ATTENDANCE MONITORING SYSTEM FOR EDUCATIONAL INSTITUTIONS

A PROJECT REPORT

Submitted by

Anuttama Ghosh

(RA2111033010133)

Under the guidance of

Ms . Sasi Rekha Sankar

(Assistant Professor, Department of Computational Intelligence)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

w/s in

SOFTWARE ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2024

Department of Computational Intelligence
SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B.Tech/ CSE-SE

Student Name : Anuttama Ghosh

Registration Number : RA2111033010133

Title of Work : ATTENDANCE MONITORING SYSTEM FOR EDUCATIONAL INSTITUTIONS

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSC303J project report titled "Attendance monitoring systems for Educational Institutions" is the bonafide work of Anuttama Ghosh (RA2111033010133) who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of the GUIDE

Ms. Sasi Rekha Sankar

Assistant Professor

Dept. of Computational Intelligence

SRM IST

Signature of the HOD

Dr. R.Annie Uthra

Head of the Department

Dept. of Computational Intelligence

SRM IST

EXAMINER I

EXAMINER II

1.Table of Contents

2. Aim

An Attendance Monitoring System (AMS) serves as a fundamental tool for educational institutions to effectively track and manage student attendance. It is a comprehensive technological solution designed to automate attendance recording, analysis, and reporting processes, thereby optimizing administrative efficiency and promoting academic success. The system encompasses various functionalities, including attendance tracking for classes, lectures, and other academic activities, as well as tools for generating reports, identifying attendance trends, and facilitating interventions when necessary.

By leveraging innovative technology and data-driven approaches, an AMS aims to address the challenges faced by educational institutions in monitoring student attendance accurately and efficiently. It streamlines the attendance tracking process, reducing manual workload for administrators and faculty members, while ensuring data accuracy and integrity. Moreover, an AMS facilitates proactive intervention strategies by providing insights into attendance patterns and trends, enabling timely interventions to support student engagement and retention.

The implementation of an AMS not only enhances operational efficiency within educational institutions but also contributes to a culture of accountability and academic excellence. By promoting regular attendance and active participation, the system supports student success and fosters a positive learning environment. Additionally, an AMS facilitates compliance with regulatory requirements and institutional policies regarding attendance recording and reporting.

In conclusion, an Attendance Monitoring System is a vital component of modern educational management, offering benefits such as improved administrative efficiency, enhanced student engagement, and better decision-making through data-driven insights. Its implementation empowers educational institutions to meet the challenges of attendance monitoring effectively, ultimately contributing to the overall success and performance of the institution.

2. Introduction:

In response to the evolving landscape of educational institutions and the increasing emphasis on student engagement and performance, our project endeavors to develop an advanced Attendance Monitoring System. This system aims to revolutionize traditional attendance tracking methods, enhance administrative efficiency, and promote a culture of accountability and academic success. By harnessing modern technology and innovative approaches, we aspire to address the challenges faced by educational institutions in effectively managing attendance and fostering student participation.

In today's educational environment, ensuring accurate attendance records and monitoring student presence has become indispensable for optimizing learning outcomes and institutional performance. With the Attendance Monitoring System, we seek to provide educational institutions with a robust platform that not only automates attendance tracking but also offers comprehensive tools for analysis, reporting, and intervention.

Our endeavor is to create a solution that goes beyond mere record-keeping, empowering institutions to proactively identify attendance trends, intervene when necessary, and support student success. By leveraging data-driven insights and leveraging user-friendly interfaces, our system aims to streamline administrative processes, reduce manual workload, and enhance overall operational efficiency.

Through extensive research and collaboration with educational stakeholders, we have identified key pain points and opportunities for improvement in attendance monitoring practices. From cumbersome manual recording methods to challenges in data accuracy and intervention strategies, our solution is designed to address these issues comprehensively, offering a seamless and intuitive experience for administrators, faculty, and students alike.

With a commitment to scalability, adaptability, and compliance with regulatory requirements, our Attendance Monitoring System promises to be a valuable asset for educational institutions of all sizes and complexities. By embracing technological innovation and best practices in attendance management, we are confident that our project will not only meet but exceed the evolving needs of educational institutions, setting new benchmarks for efficiency, transparency, and student success in the process.

3.Requirements Elicitation Techniques:

- Conducting interviews and focus groups with key stakeholders, including administrators, faculty members, students, and parents, provides an opportunity to gather insights into their perspectives, needs, and expectations regarding attendance tracking and management. Through open-ended discussions and structured sessions, stakeholders can share their experiences, preferences, and concerns related to attendance monitoring, helping identify requirements for the system.Process:
- Identify Stakeholders: Determine the relevant stakeholders involved in attendance monitoring, such as administrators responsible for policy implementation, faculty members responsible for attendance recording, students required to attend classes, and parents interested in monitoring their child's attendance.
- Schedule Interviews and Focus Groups: Arrange individual interviews with key stakeholders to delve deeper into their roles, responsibilities, and requirements. Additionally, organize focus group sessions with representative groups to encourage discussion and collaboration among stakeholders.
- Prepare Discussion Guides: Develop interview scripts or focus group agendas with a mix of open-ended questions and specific prompts to explore various aspects of attendance monitoring, including current practices, challenges faced, desired features, and expectations from the system.
- Conduct Sessions: Facilitate interviews and focus groups, ensuring active participation from stakeholders and encouraging them to share their perspectives freely. Record insights, observations, and feedback provided during the sessions.
- Analyze Findings: Collate and analyze the information gathered from interviews and focus groups to identify common themes, priorities, and requirements for the Attendance Monitoring System. Look for patterns, discrepancies, and areas of consensus among stakeholders.
- Benefits:
- Provides insights from diverse stakeholder perspectives, ensuring comprehensive coverage of requirements.

- Facilitates in-depth exploration of stakeholder needs, preferences, and concerns.
- Promotes stakeholder engagement and ownership in the requirements elicitation process.
- Enables clarification of ambiguous or conflicting requirements through interactive discussions.
- Prototype Demonstrations and Feedback Sessions:Description: Creating prototypes or mock-ups of the Attendance Monitoring System and conducting feedback sessions with end-users allows stakeholders to interact with the proposed system interface and functionality, providing valuable feedback on usability, features, and design. By iteratively refining the prototype based on user input, requirements for the system can be elicited and validated in a tangible and interactive manner.Process:
- Develop Prototypes: Create prototypes or wireframes of the Attendance Monitoring System, focusing on key features such as attendance recording interfaces, reporting dashboards, and notification systems. Use prototyping tools or mock-up software to design user interfaces that simulate the intended functionality of the system.
- Arrange Feedback Sessions: Organize feedback sessions with representative end-users, including administrators, faculty members, and students, to demonstrate the prototypes and gather their feedback. Schedule multiple sessions as needed to accommodate different user groups and iterate on the prototypes.
- Present Prototypes: Walk stakeholders through the prototype, explaining its purpose, features, and workflow. Encourage participants to interact with the prototype, simulate typical usage scenarios, and provide feedback on usability, clarity, and intuitiveness.
- Collect Feedback: Capture feedback from stakeholders during the sessions, documenting their comments, suggestions, and concerns regarding the prototype. Use techniques such as surveys, interviews, or observation to gather qualitative and quantitative feedback on specific aspects of the system.
- Iterative Refinement: Incorporate feedback from each feedback session into the prototype, making iterative improvements to address identified issues, enhance usability, and align with stakeholder preferences. Continue to refine the prototype based on ongoing feedback until consensus is reached on the system requirements.

- Benefits:
- Allows stakeholders to visualize and interact with the proposed system, facilitating a deeper understanding of its functionality and features.
- Provides immediate feedback on usability, interface design, and user experience, enabling early identification of design flaws or usability issues.
- Promotes collaboration and co-creation between developers and end-users, fostering a sense of ownership and investment in the final product.
- Supports iterative development and requirements validation, ensuring that the system meets the needs and expectations of end-users effectively.

4.Functional Requirements:

- User Authentication: The system should allow users, including students, faculty, and administrative staff, to authenticate themselves securely before accessing attendance-related features.

- Attendance Recording: The system should provide mechanisms for recording attendance for various academic activities, such as classes, lectures, seminars, and examinations. This may include manual entry, barcode scanning, biometric authentication, or mobile check-ins.

- Attendance Management: The system should allow authorized users to manage attendance records, including marking absences, late arrivals, and early departures. It should also support the assignment of reasons for absences and the approval of leave requests.

- Attendance Tracking: The system should enable users to track attendance trends over time, including individual student attendance, class-level attendance rates, and overall institutional attendance statistics. It should provide tools for analyzing attendance data to identify patterns, trends, and outliers.

- Automated Notifications: The system should send automated notifications to relevant stakeholders, such as students, faculty, and administrators, regarding attendance-related matters. This may include reminders for upcoming classes, alerts for excessive absences, and notifications for approved leave requests.

- Reporting and Analytics: The system should offer reporting and analytics tools to generate attendance reports, summaries, and insights. It should allow users to customize report parameters, filter data by various criteria (e.g., date range, course code), and export reports in different formats (e.g., PDF, CSV).

- Integration with Other Systems: The system should integrate seamlessly with other institutional systems, such as student information systems (SIS), learning management systems (LMS), and course scheduling systems. This ensures data consistency, eliminates duplicate data entry, and streamlines administrative workflows.

- Role-Based Access Control: The system should enforce role-based access control (RBAC), allowing administrators to define user roles and permissions based on their responsibilities and organizational hierarchy. This ensures that only authorized users can access and modify attendance data.

- Customization Options: The system should provide customization options to tailor attendance tracking methods, attendance policies, notification settings, and reporting formats to the specific requirements of the educational institution, departments, or academic programs.

- Offline Access and Synchronization: The system should support offline access to attendance features in case of network disruptions or connectivity issues. It should also synchronize attendance data automatically once the connection is restored to ensure data integrity and consistency.

- Student Engagement Features: The system may include features to enhance student engagement and participation, such as gamification elements, rewards for regular attendance, and interactive dashboards displaying attendance trends and progress.

- Compliance: The system should comply with relevant regulatory requirements and institutional policies regarding attendance tracking, data privacy, and security. This includes adherence to data protection regulations (e.g., GDPR, FERPA) and institutional policies on attendance reporting and record-keeping.

5.Non-Functional Requirements:

- Non-Functional Requirements for Attendance Monitoring System in Educational Institutions:

- Scalability: The system should be capable of handling varying numbers of users and attendance records, scaling seamlessly as the institution grows or experiences fluctuations in enrollment.
- Performance: The system should exhibit fast response times for attendance recording and retrieval, even during peak usage periods. It should also minimize latency in data processing and reporting.
- Reliability: The system should maintain high availability, with minimal downtime for maintenance or upgrades. It should also ensure data integrity and consistency, preventing loss or corruption of attendance records.
- Security: The system should implement robust security measures to protect sensitive attendance data from unauthorized access, tampering, or breaches. This includes user authentication, data encryption, access controls, and compliance with relevant privacy regulations.
- Usability: The system should have an intuitive user interface that is easy to navigate for both administrators and end-users (e.g., faculty, students). It should also provide clear instructions for attendance recording, along with error handling and feedback mechanisms.
- Compatibility: The system should be compatible with a variety of devices and platforms commonly used in educational settings, including desktop computers, laptops, tablets, and mobile phones. It should also support integration with existing institutional software systems, such as student information systems and learning management systems.
- Customizability: The system should allow for customization of attendance tracking methods, reporting formats, and administrative settings to accommodate the unique requirements of different educational institutions, departments, or programs.
- Auditability: The system should maintain comprehensive audit logs to track changes to attendance records, user actions, and system

configurations. This supports accountability, compliance, and forensic analysis in case of disputes or investigations.

- Performance Monitoring: The system should include monitoring tools to track system performance, resource utilization, and user activity. This enables administrators to identify and address potential bottlenecks, optimize system performance, and allocate resources effectively.
- Accessibility: The system should adhere to accessibility standards and guidelines to ensure that users with disabilities can access and use the system effectively. This includes support for assistive technologies, such as screen readers and keyboard navigation, and compliance with accessibility regulations (e.g., WCAG).
- Maintainability: The system should be designed with modularity, clear documentation, and well-defined interfaces to facilitate maintenance, updates, and troubleshooting. This includes version control, code documentation, and procedures for bug fixes and enhancements.
- Cost-Effectiveness: The system should provide value for money in terms of implementation, maintenance, and support costs. It should offer flexible pricing models and scalability options to accommodate institutional budgets and evolving needs over time.

6.Module Description:

Functionality:

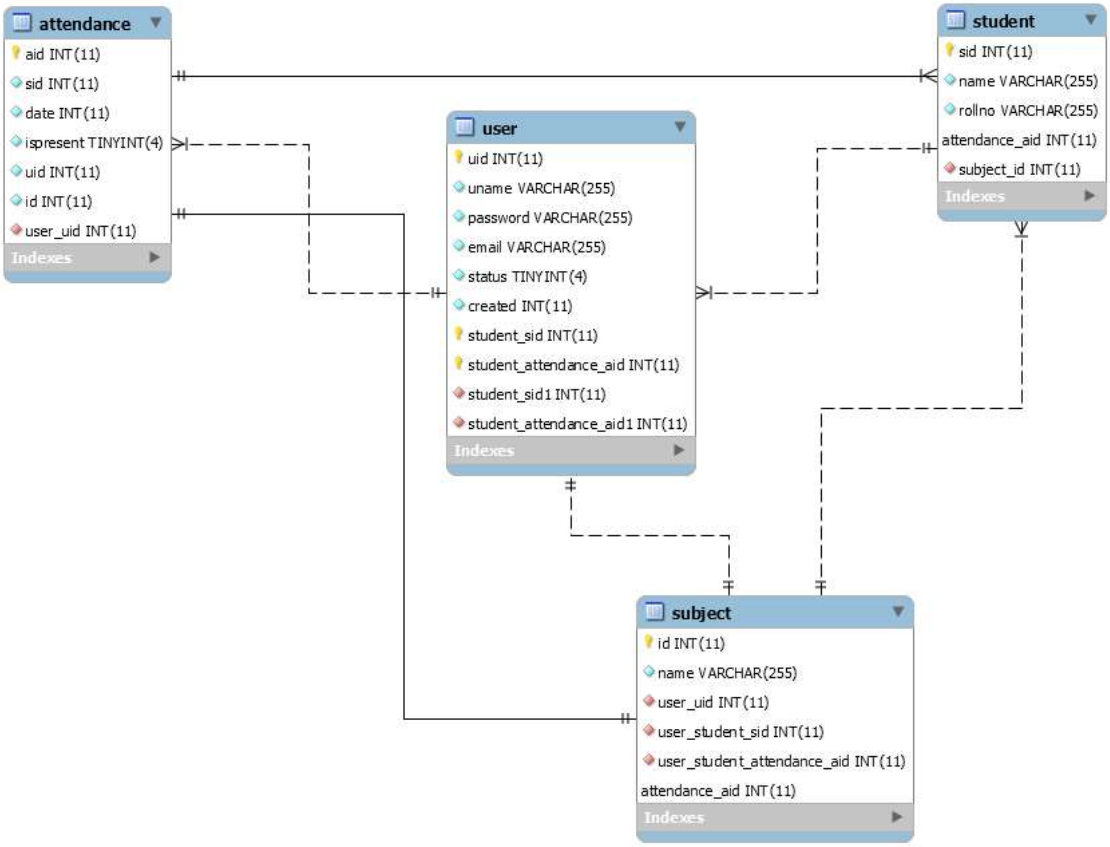
- This module is designed to manage attendance records for students and faculty members within educational institutions.
- It facilitates the tracking of attendance for classes, lectures, workshops, and other academic activities.
- It provides features for recording absences, late arrivals, and early departures, along with the reasons provided by students or staff.
- It includes tools for generating attendance reports, analyzing trends, and identifying patterns to support academic planning and intervention strategies.
- It supports integration with other institutional systems, such as student information systems and learning management systems, to ensure data accuracy and streamline administrative processes.
- It may include features for automating attendance-taking through biometric authentication, RFID technology, or mobile applications for enhanced efficiency and accuracy.
- It allows for customization based on institutional requirements, including attendance policies, notification preferences, and reporting formats.
- Justification for Modularization:
- Attendance monitoring is a fundamental aspect of educational administration, influencing student performance, resource allocation, and compliance with regulatory requirements. By modularizing this functionality, educational institutions can:

- Tailor attendance tracking methods: Different institutions may prefer various methods for attendance recording, such as manual entry, biometric scanning, or mobile check-ins. Modularization allows for the adaptation of the system to accommodate these preferences.
- Integrate with existing systems: Educational institutions often use multiple software solutions for different purposes, such as student management and course delivery. Modularization enables seamless

integration with these existing systems, ensuring data consistency and reducing administrative overhead.

- Customize reporting and analysis: Attendance data can provide insights into student engagement, course popularity, and faculty performance. Modularization allows for the customization of reports and analytics tools to meet the specific needs of administrators, educators, and policymakers.
- Ensure scalability and flexibility: Educational institutions vary in size, structure, and operational requirements. Modularization enables the system to scale effectively as the institution grows and adapt to changing needs, such as new academic programs or shifts in attendance policies.
- Enhance security and compliance: Attendance records contain sensitive information about students and staff, requiring robust security measures and compliance with data protection regulations. Modularization facilitates the implementation of security protocols, such as access controls and data encryption, to safeguard this information effectively.
- Support academic interventions: Timely access to attendance data can help identify students at risk of academic disengagement or non-compliance with attendance policies. Modularization allows for the integration of intervention tools, such as automated notifications and student support services, to improve retention rates and academic outcomes.

7. System Architecture Diagram:



8.Conceptual Design of the attendance monitoring system for educational institutions database:

1. **STUDENT:**
    - student_id (Uniquely identifies the student)
    - student_name (Name of the student)
    - student_dob (Date of birth of the student)
    - student_gender (Gender of the student)
    - student_address1 (Address line 1 of the student)
    - student_address2 (Address line 2 of the student)
    - student_city (City of residence of the student)
    - student_state (State of residence of the student)
    - student_zip (Zip code of the student)
2. **COURSE:**
    - course_id (Uniquely identifies the course)
    - course_name (Name of the course)
3. **CLASS:**
    - class_id (Uniquely identifies the class)
    - class_name (Name or code of the class)
    - course_id (References the course the class belongs to)
4. **TEACHER:**
    - teacher_id (Uniquely identifies the teacher)
    - teacher_name (Name of the teacher)
    - teacher_gender (Gender of the teacher)
    - teacher_address1 (Address line 1 of the teacher)
    - teacher_address2 (Address line 2 of the teacher)
    - teacher_city (City of residence of the teacher)
    - teacher_state (State of residence of the teacher)
    - teacher_zip (Zip code of the teacher)
5. **ATTENDANCE:**
    - attendance_id (Uniquely identifies the attendance record)
    - student_id (References the student)
    - class_id (References the class)
    - teacher_id (References the teacher)

- attendance_date (Date of the attendance record)
- attendance_status (Indicates whether the student was present, absent, etc.)

6. **ABSENCE_REASON (Optional):**
    - absence_reason_id (Uniquely identifies the absence reason)
    - reason_description (Description of the absence reason)
7. **USER (Optional, for authentication and authorization):**
    - user_id (Uniquely identifies the user)
    - username (Username for login)
    - password (Password for login)
    - role (Role of the user, e.g., admin, teacher, student)
    - USER (Optional, for authentication and authorization):
    - user_id (Uniquely identifies the user)
    - username (Username for login)
    - password (Password for login)
    - role (Role of the user, e.g., admin, teacher, student)
8. **ATTENDANCE_PERIOD:**
    - period_id (Uniquely identifies the attendance period)
    - period_name (Name or code of the attendance period, e.g., Morning, Afternoon)
    - start_time (Start time of the attendance period)
    - end_time (End time of the attendance period)
9. **ATTENDANCE_MARKING:**
    - marking_id (Uniquely identifies the attendance marking)
    - marking_date (Date of the attendance marking)
    - period_id (References the attendance period)
    - class_id (References the class)
    - teacher_id (References the teacher)
    - marked_by (Identifier of the user who marked the attendance)
10. **ATTENDANCE_EXCEPTION:**
    - exception_id (Uniquely identifies the attendance exception)
    - student_id (References the student)
    - class_id (References the class)
    - date (Date of the exception)
    - reason (Reason for the exception, e.g., Medical Leave, Official Event)
11. **ATTENDANCE_ALERT:**

- alert_id (Uniquely identifies the attendance alert)
- student_id (References the student)
- class_id (References the class)
- alert_date (Date of the alert)
- alert_message (Message indicating the attendance issue)
- resolved (Flag indicating if the alert is resolved)

12. ATTENDANCE_REPORT:

- report_id (Uniquely identifies the attendance report)
- student_id (References the student)
- class_id (References the class)
- from_date (Start date of the report period)
- to_date (End date of the report period)
- total_attendance_days (Total number of attendance days)
- total_absences (Total number of absences)
- total_present (Total number of days present)
- total_late (Total number of late attendances)

13. ATTENDANCE_RULES:

- rule_id (Uniquely identifies the attendance rule)
- rule_name (Name or description of the rule)
- threshold_percentage (Percentage threshold for attendance)
- late_threshold_minutes (Threshold for considering late attendance)
- penalty_type (Type of penalty for attendance violations)

14. ATTENDANCE_HISTORY:

- history_id (Uniquely identifies the attendance history)
- student_id (References the student)
- class_id (References the class)
- attendance_date (Date of the attendance record)
- attendance_status (Status of the attendance record, e.g., Present, Absent, Late)

15. ATTENDANCE_ANALYTICS:

- analytics_id (Uniquely identifies the attendance analytics)
- student_id (References the student)
- class_id (References the class)
- analysis_date (Date of the analytics)
- attendance_percentage (Percentage of attendance for the student)

- trend (Trend of attendance over time)

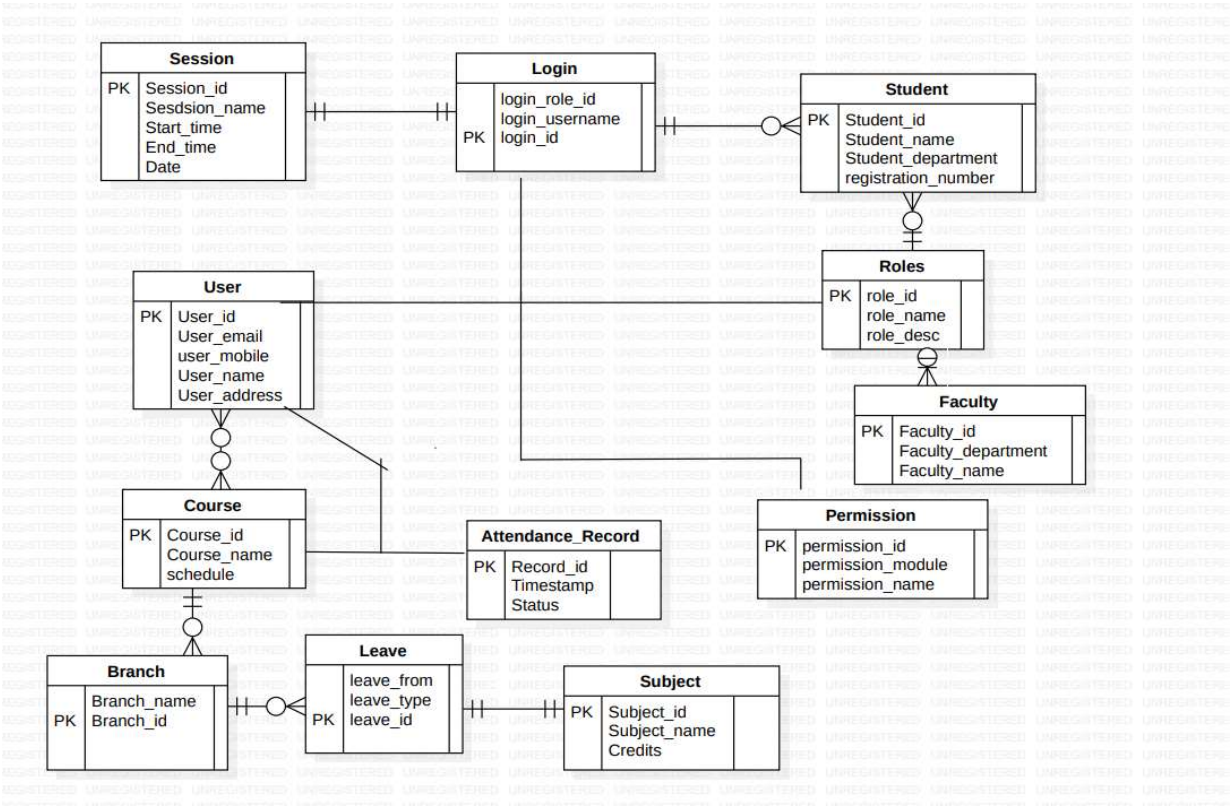
16. ATTENDANCE_SETTINGS:

- setting_id (Uniquely identifies the attendance setting)
- setting_name (Name or description of the setting)
- value (Value of the setting, e.g., True/False, Integer, String)

17. ATTENDANCE_IMPORT_LOG:

- import_log_id (Uniquely identifies the attendance import log)
- import_date (Date of the import)
- imported_by (Identifier of the user who imported the data)
- success_records (Number of successfully imported records)
- error_records (Number of records with import errors)

9. ER Diagram:


Session
PK
Session_id
Sesdsion_name
Start_time
End_time
Date
Login
login_role_id
login_username
PK
login_id
Student
PK
Student_id
Student_name
Student_department
registration_number
Roles
PK
role_id
role_name
role_desc
User
PK
User_id
User_email
user_mobile
User_name
User_address
Faculty
PK
Faculty_id
Faculty_department
Faculty_name
Course
PK
Course_id
Course_name
schedule
Attendance_Record
PK
Record_id
Timestamp
Status
Permission
PK
permission_id
permission_module
permission_name
Branch
Branch_name
PK
Branch_id
Leave
leave_from
leave_type
PK
leave_id
Subject
PK
Subject_id
Subject_name
Credits

10. Relational Model:

Table Name	Attributes	Primary key	Foreign Key	Description
STUDENT	student_id, student_name, student_dob, student_gender, student_address1, student_address2, student_city, student_state, student_zip	student_id		Details of students including their personal information
COURSE	course_id, course_name	course_id		Information about academic courses offered
CLASS	class_id, class_name, course_id	class_id	course_id (COURSE)	Classes offered within each course
TEACHER	teacher_id, teacher_name, teacher_gender, teacher_address1, teacher_address2, teacher_city, teacher_state, teacher_zip	teacher_id		Information about teachers
ATTENDANCE	attendance_id, student_id, class_id, teacher_id, attendance_date, attendance_status	attendance_id	student_id (STUDENT), class_id (CLASS), teacher_id (TEACHER)	Records of student attendance in classes
ABSENCE_REASON	absence_reason_id, reason_description	absence_reason_id		Reasons for student absences
USER	user_id, username, password, role	user_id		Users with access to the attendance system
ATTENDANCE_PERIOD	period_id, period_name, start_time,	period_id		Time periods for attendance

	end_time			marking
ATTENDANCE_MARKING	marking_id, marking_date, period_id, class_id, teacher_id, marked_by	marking_id	period_id (ATTENDANCE_PERIOD), class_id (CLASS), teacher_id (TEACHER)	Records of attendance marked by teachers
ATTENDANCE_EXCEPTION	exception_id, student_id, class_id, date, reason	exception_id	student_id (STUDENT), class_id (CLASS)	Exceptions to standard attendance records
ATTENDANCE_ALERT	alert_id, student_id, class_id, alert_date, alert_message, resolved	alert_id	student_id (STUDENT), class_id (CLASS)	Alerts generated for attendance issues
ATTENDANCE_REPORT	alert_id, student_id, class_id, alert_date, alert_message, resolved	report_id	student_id (STUDENT), class_id (CLASS)	Reports summarizing attendance data over a period
ATTENDANCE_RULES	rule_id, rule_name, threshold_percentage, late_threshold_minutes, penalty_type	rule_id		Rules defining attendance thresholds and penalties
ATTENDANCE_HISTORY	history_id, student_id, class_id, attendance_date, attendance_status	history_id	student_id (STUDENT), class_id (CLASS)	Historical records of student attendance
ATTENDANCE_ANALYTICS	analytics_id, student_id, class_id, analysis_date, attendance_percentage, trend	analytics_id	student_id (STUDENT), class_id (CLASS)	Analytical insights derived from attendance data
ATTENDANCE_SETTINGS	setting_id, setting_name, value	setting_id		Settings for configuring attendance system behavior
ATTENDANCE_IMPORT_LOG	import_log_id, import_date, imported_by, success_records, error_records	import_log_id		Logs of attendance data import operations

11. Data Definition Language:

Data Definition Language (DDL) is a subset of SQL used to define, alter, and manage the structure of database objects such as tables, indexes, and schemas. Key DDL commands include:

1. CREATE - Used to create new database objects, such as tables, views, or databases.
2. ALTER - Modifies the structure of existing database objects, allowing for changes like adding new columns to tables or changing a column's datatype.
3. DROP - Deletes database objects permanently from the database, removing both the structure and its data.
4. TRUNCATE - Removes all records from a table, but keeps the structure for future use.

CODE:

```
-- Table for students

CREATE TABLE Student (

    student_id INT NOT NULL PRIMARY KEY,

    student_name VARCHAR(50),

    student_dob DATE,

    student_gender CHAR(1),

    student_address1 VARCHAR(100),

    student_address2 VARCHAR(100),

    student_city VARCHAR(50),

    student_state CHAR(2),

    student_zip VARCHAR(10)

);


-- Table for courses

CREATE TABLE Course (

    course_id INT NOT NULL PRIMARY KEY,

    course_name VARCHAR(100)

);
```

```
-- Table for classes
CREATE TABLE Class (
    class_id INT NOT NULL PRIMARY KEY,
    class_name VARCHAR(100),
    course_id INT,
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

-- Table for teachers
CREATE TABLE Teacher (
    teacher_id INT NOT NULL PRIMARY KEY,
    teacher_name VARCHAR(100),
    teacher_gender CHAR(1),
    teacher_address1 VARCHAR(100),
    teacher_address2 VARCHAR(100),
    teacher_city VARCHAR(50),
    teacher_state CHAR(2),
    teacher_zip VARCHAR(10)
);

-- Table for attendance
CREATE TABLE Attendance (
    attendance_id INT NOT NULL PRIMARY KEY,
    student_id INT,
    class_id INT,
    teacher_id INT,
    attendance_date DATE,
    attendance_status VARCHAR(20),
```

```
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id),
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);

-- Table for absence reasons
CREATE TABLE Absence_Reason (
    absence_reason_id INT NOT NULL PRIMARY KEY,
    reason_description VARCHAR(200)
);

-- Table for users (optional for authentication and authorization)
CREATE TABLE User (
    user_id INT NOT NULL PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(100),
    role VARCHAR(20)
);

-- Table for attendance periods
CREATE TABLE Attendance_Period (
    period_id INT NOT NULL PRIMARY KEY,
    period_name VARCHAR(50),
    start_time TIME,
    end_time TIME
);

-- Table for attendance marking
```

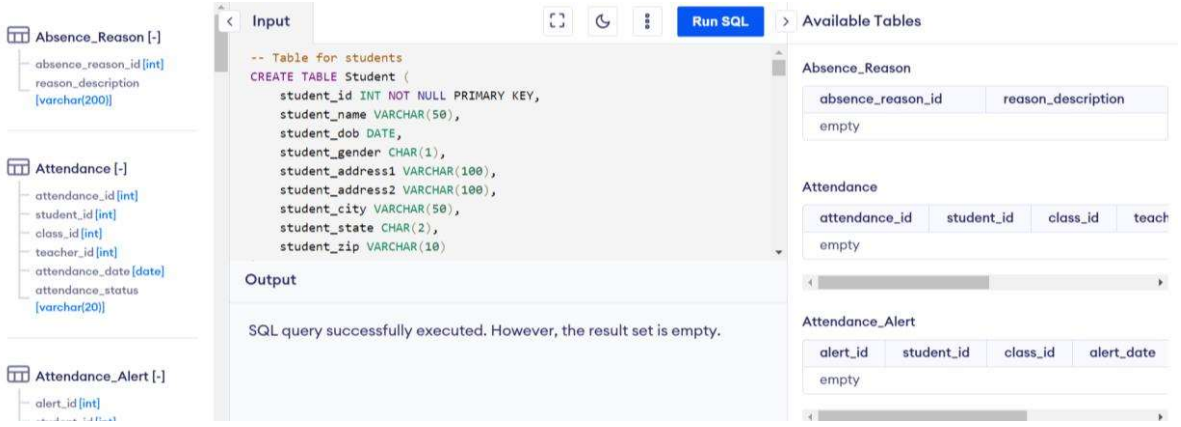
```
CREATE TABLE Attendance_Marking (
    marking_id INT NOT NULL PRIMARY KEY,
    marking_date DATE,
    period_id INT,
    class_id INT,
    teacher_id INT,
    marked_by INT,
    FOREIGN KEY (period_id) REFERENCES Attendance_Period(period_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id),
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id),
    FOREIGN KEY (marked_by) REFERENCES User(user_id)
);

-- Table for attendance exceptions
CREATE TABLE Attendance_Exception (
    exception_id INT NOT NULL PRIMARY KEY,
    student_id INT,
    class_id INT,
    date DATE,
    reason VARCHAR(200),
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id)
);

-- Table for attendance alerts
CREATE TABLE Attendance_Alert (
    alert_id INT NOT NULL PRIMARY KEY,
    student_id INT,
```

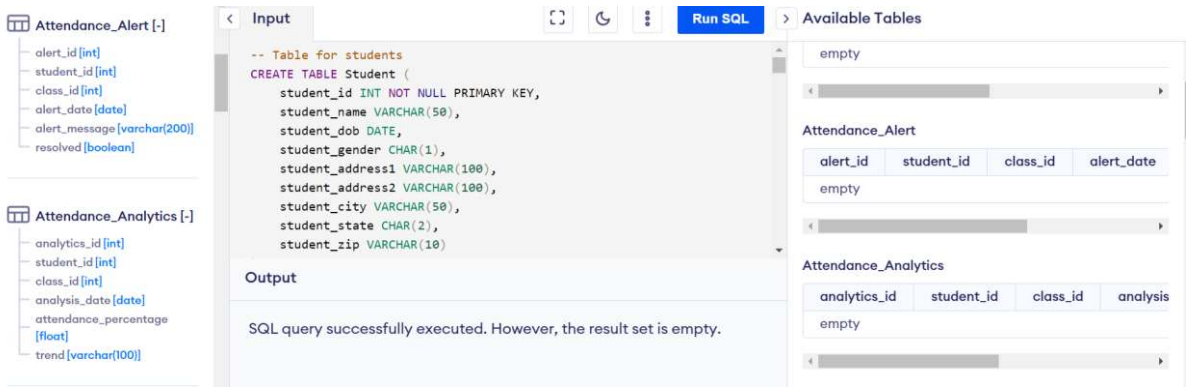
```
    class_id INT,
    alert_date DATE,
    alert_message VARCHAR(200),
    resolved BOOLEAN,
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id)
);

-- Table for attendance reports
CREATE TABLE Attendance_Report (
    report_id INT NOT NULL PRIMARY KEY,
    student_id INT,
    class_id INT,
    from_date DATE,
    to_date DATE,
    total_attendance_days INT,
    total_absences INT,
    total_present INT,
    total_late INT,
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id)
);

-- Table for attendance rules
CREATE TABLE Attendance_Rules (
    rule_id INT NOT NULL PRIMARY KEY,
    rule_name VARCHAR(100),
    threshold_percentage FLOAT,
```

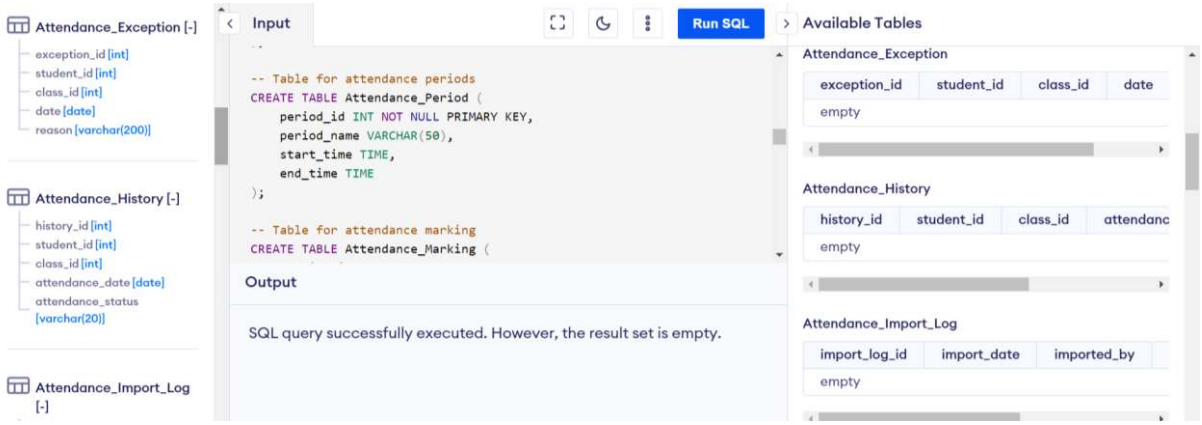
```
    late_threshold_minutes INT,
    penalty_type VARCHAR(50)
);

-- Table for attendance history
CREATE TABLE Attendance_History (
    history_id INT NOT NULL PRIMARY KEY,
    student_id INT,
    class_id INT,
    attendance_date DATE,
    attendance_status VARCHAR(20),
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id)
);

-- Table for attendance analytics
CREATE TABLE Attendance_Analytics (
    analytics_id INT NOT NULL PRIMARY KEY,
    student_id INT,
    class_id INT,
    analysis_date DATE,
    attendance_percentage FLOAT,
    trend VARCHAR(100),
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (class_id) REFERENCES Class(class_id)
);

-- Table for attendance settings
```

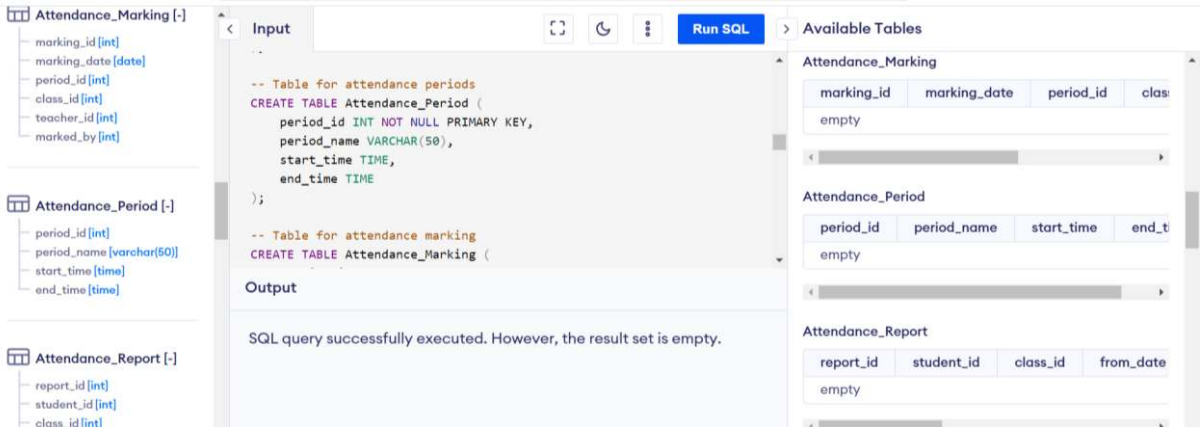
```
CREATE TABLE Attendance_Settings (
    setting_id INT NOT NULL PRIMARY KEY,
    setting_name VARCHAR(100),
    value VARCHAR(100)
);

-- Table for attendance import log
CREATE TABLE Attendance_Import_Log (
    import_log_id INT NOT NULL PRIMARY KEY,
    import_date DATE,
    imported_by INT,
    success_records INT,
    error_records INT,
    FOREIGN KEY (imported_by) REFERENCES User(user_id)
);
```

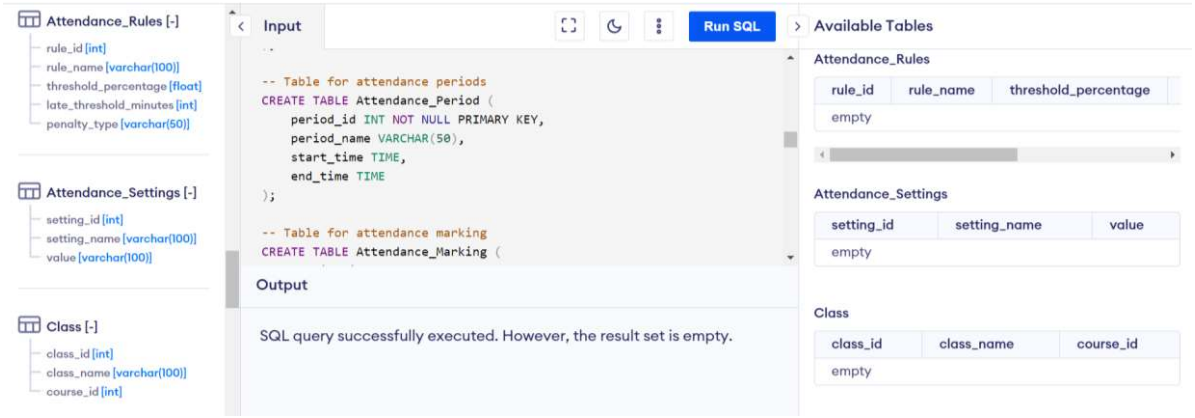
Output:

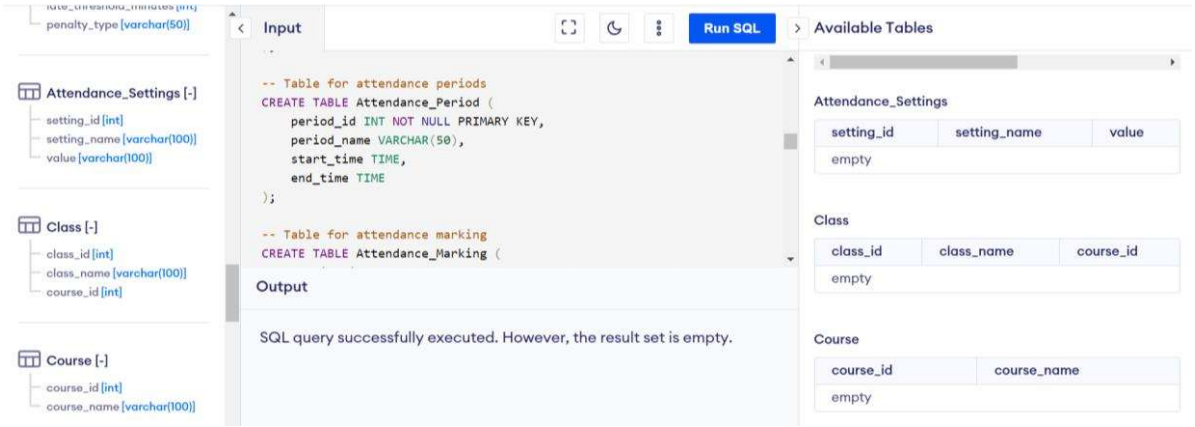



Attendance_Alert [-]
alert_id [int]
student_id [int]
class_id [int]
alert_date [date]
alert_message [varchar(200)]
resolved [boolean]
Attendance_Analytics [-]
analytics_id [int]
student_id [int]
class_id [int]
analysis_date [date]
attendance_percentage [float]
trend [varchar(100)]
Input
Run SQL
Available Tables
-- Table for students
CREATE TABLE Student (
student_id INT NOT NULL PRIMARY KEY,
student_name VARCHAR(50),
student_dob DATE,
student_gender CHAR(1),
student_address1 VARCHAR(100),
student_address2 VARCHAR(100),
student_city VARCHAR(50),
student_state CHAR(2),
student_zip VARCHAR(10)
Output
SQL query successfully executed. However, the result set is empty.
empty
Attendance_Alert
alert_id
student_id
class_id
alert_date
empty
Attendance_Analytics
analytics_id
student_id
class_id
analysis
empty


Attendance_Exception [-]
exception_id [int]
student_id [int]
class_id [int]
date [date]
reason [varchar(200)]
Attendance_History [-]
history_id [int]
student_id [int]
class_id [int]
attendance_date [date]
attendance_status [varchar(20)]
Attendance_Import_Log [-]
Input
Run SQL
Available Tables
-- Table for attendance periods
CREATE TABLE Attendance_Period (
period_id INT NOT NULL PRIMARY KEY,
period_name VARCHAR(50),
start_time TIME,
end_time TIME
);
-- Table for attendance marking
CREATE TABLE Attendance_Marking (
Output
SQL query successfully executed. However, the result set is empty.
Attendance_Exception
exception_id
student_id
class_id
date
empty
Attendance_History
history_id
student_id
class_id
attendanc
empty
Attendance_Import_Log
import_log_id
import_date
imported_by
empty


Attendance_Marking [-]
marking_id [int]
marking_date [date]
period_id [int]
class_id [int]
teacher_id [int]
marked_by [int]
Attendance_Period [-]
period_id [int]
period_name [varchar(50)]
start_time [time]
end_time [time]
Attendance_Report [-]
report_id [int]
student_id [int]
Input
Run SQL
Available Tables
-- Table for attendance periods
CREATE TABLE Attendance_Period (
period_id INT NOT NULL PRIMARY KEY,
period_name VARCHAR(50),
start_time TIME,
end_time TIME
);
-- Table for attendance marking
CREATE TABLE Attendance_Marking (
Output
SQL query successfully executed. However, the result set is empty.
Attendance_Marking
marking_id
marking_date
period_id
clas
empty
Attendance_Period
period_id
period_name
start_time
end_t
empty
Attendance_Report
report_id
student_id
class_id
from_date
empty


Attendance_Rules [-]
rule_id [int]
rule_name [varchar(100)]
threshold_percentage [float]
late_threshold_minutes [int]
penalty_type [varchar(50)]
Attendance_Settings [-]
setting_id [int]
setting_name [varchar(100)]
value [varchar(100)]
Class [-]
class_id [int]
class_name [varchar(100)]
course_id [int]
Input
Run SQL
-- Table for attendance periods
CREATE TABLE Attendance_Period (
period_id INT NOT NULL PRIMARY KEY,
period_name VARCHAR(50),
start_time TIME,
end_time TIME
);
-- Table for attendance marking
CREATE TABLE Attendance_Marking (
Output
SQL query successfully executed. However, the result set is empty.
Available Tables
Attendance_Rules
rule_id
rule_name
threshold_percentage
empty
Attendance_Settings
setting_id
setting_name
value
empty
Class
class_id
class_name
course_id
empty


penalty_type [varchar(50)]
Attendance_Settings [-]
setting_id [int]
setting_name [varchar(100)]
value [varchar(100)]
Class [-]
class_id [int]
class_name [varchar(100)]
course_id [int]
Course [-]
course_id [int]
course_name [varchar(100)]
Input
Run SQL
-- Table for attendance periods
CREATE TABLE Attendance_Period (
period_id INT NOT NULL PRIMARY KEY,
period_name VARCHAR(50),
start_time TIME,
end_time TIME
);
-- Table for attendance marking
CREATE TABLE Attendance_Marking (
Output
SQL query successfully executed. However, the result set is empty.
Available Tables
Attendance_Settings
setting_id
setting_name
value
empty
Class
class_id
class_name
course_id
empty
Course
course_id
course_name
empty

12. Data Manipulation Language:

Data Manipulation Language (DML) is a subset of SQL used to retrieve, insert, modify, and delete data within database tables. Key DML commands include:

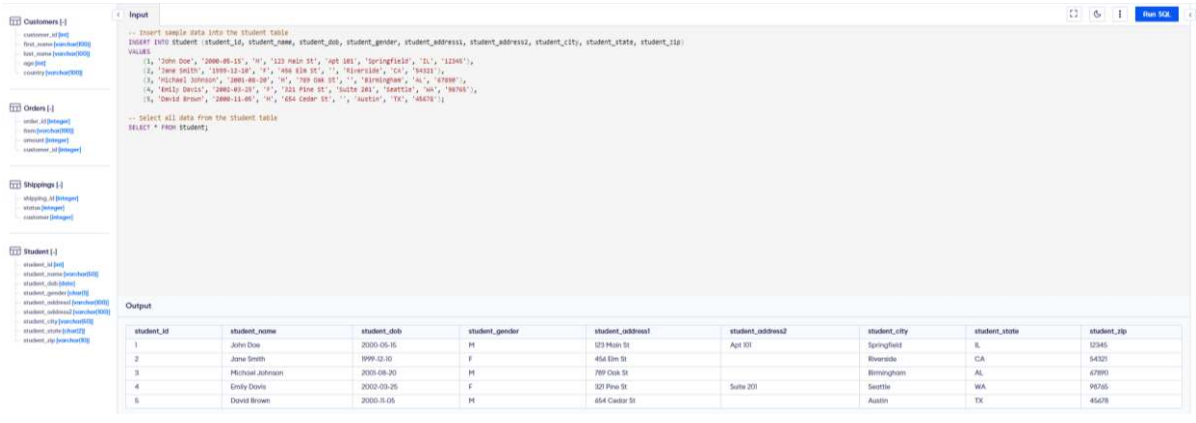
1. SELECT- Retrieves data from one or more database tables or views.
2. INSERT- Adds new rows of data to specified tables.
3. UPDATE - Modifies existing data within a table.
4. DELETE- Removes existing rows from a table.

DML commands focus on handling the data within the tables, unlike DDL which manages table and schema structure. These commands are the primary tools used for interacting with and manipulating data in everyday database operations.

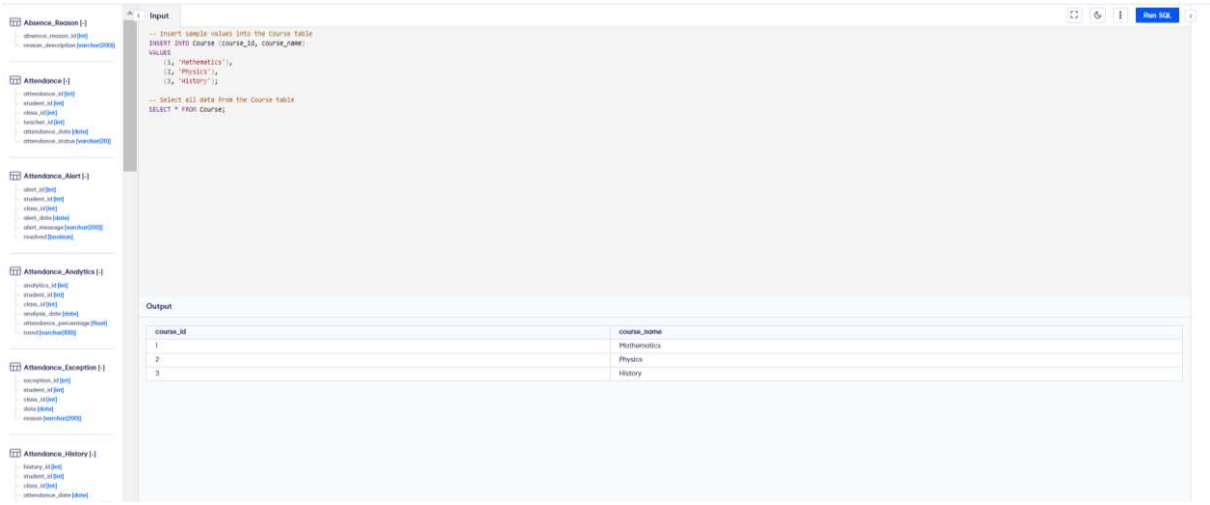
1)Insert and select command used in table student:

```
-- Insert sample data into the Student table

INSERT INTO Student (student_id, student_name, student_dob, student_gender, student_address1,
student_address2, student_city, student_state, student_zip)

VALUES

    (1, 'John Doe', '2000-05-15', 'M', '123 Main St', 'Apt 101', 'Springfield', 'IL', '12345'),

    (2, 'Jane Smith', '1999-12-10', 'F', '456 Elm St', '', 'Riverside', 'CA', '54321'),

    (3, 'Michael Johnson', '2001-08-20', 'M', '789 Oak St', '', 'Birmingham', 'AL', '67890'),

    (4, 'Emily Davis', '2002-03-25', 'F', '321 Pine St', 'Suite 201', 'Seattle', 'WA', '98765'),

    (5, 'David Brown', '2000-11-05', 'M', '654 Cedar St', '', 'Austin', 'TX', '45678');


-- Select all data from the Student table

SELECT * FROM Student;
```

Output:



2)Insert and select command used in table course:



3)Insert and select command used in table Attendance_Rules table :

-- Insert sample data into the Attendance_Rules table

INSERT INTO Attendance_Rules (rule_id, rule_name, threshold_percentage, late_threshold_minutes, penalty_type)

VALUES

(1, 'Standard', 80.0, 15, 'Warning'),

(2, 'Strict', 90.0, 10, 'Deduction'),

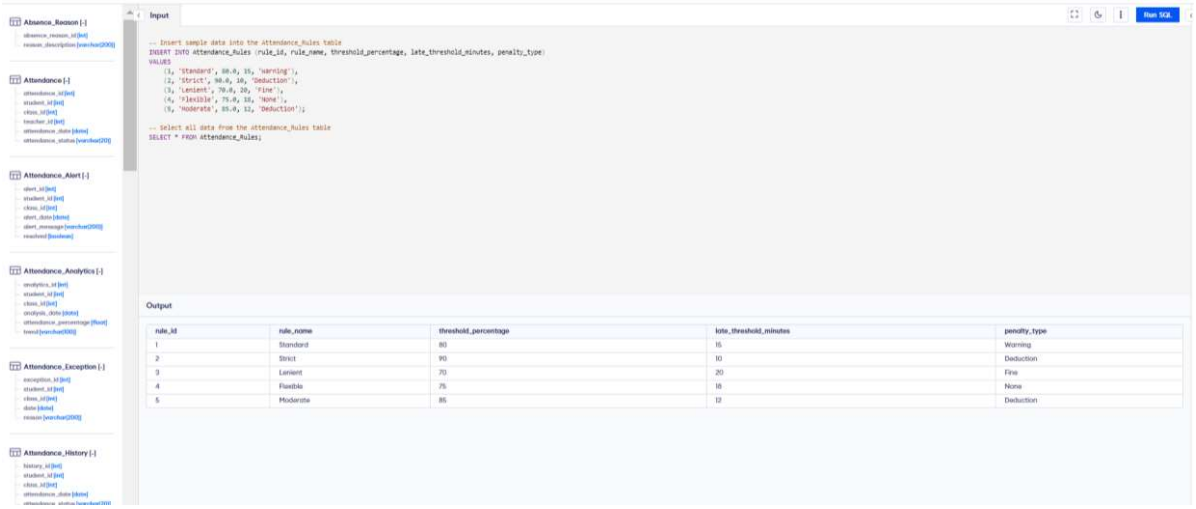
(3, 'Lenient', 70.0, 20, 'Fine'),

(4, 'Flexible', 75.0, 18, 'None'),

(5, 'Moderate', 85.0, 12, 'Deduction');

-- Select all data from the Attendance_Rules table

SELECT * FROM Attendance_Rules;



2)Update and delete command used in table course:

-- Update operation: Change the course name for a specific course_id

UPDATE Course

SET course_name = 'Computer Science'

WHERE course_id = 3;

-- Select all data from the Course table after the update

SELECT * FROM Course;

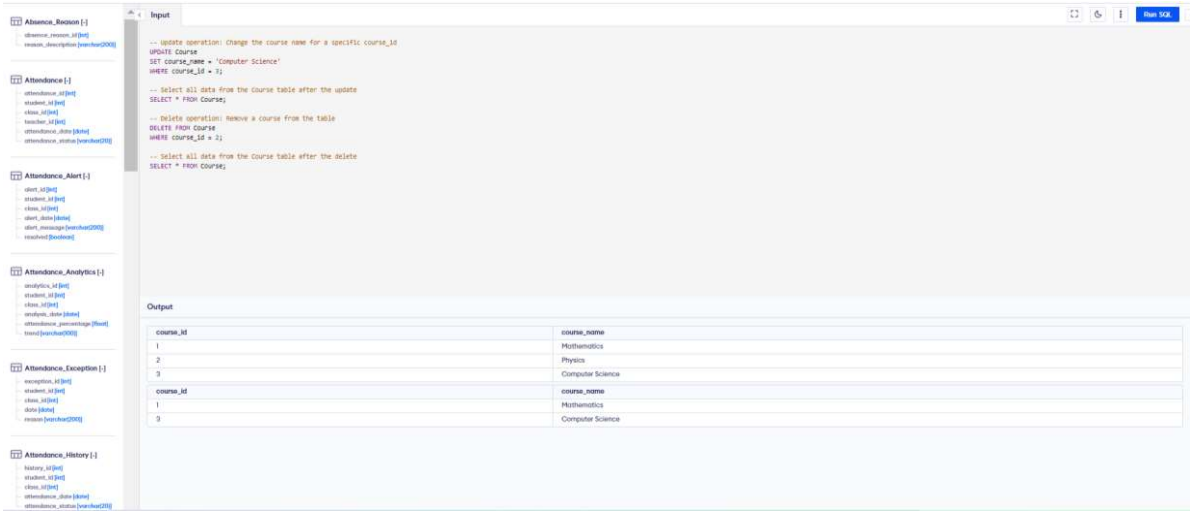
-- Delete operation: Remove a course from the table

DELETE FROM Course

WHERE course_id = 2;

-- Select all data from the Course table after the delete

SELECT * FROM Course;


Absence_Reason [-]
absence_reason_id [int]
reason_description [varchar(200)]
Attendance [-]
attendance_id [int]
student_id [int]
class_id [int]
teacher_id [int]
attendance_date [date]
attendance_status [varchar(20)]
Attendance_Alert [-]
alert_id [int]
student_id [int]
class_id [int]
alert_date [date]
alert_message [varchar(200)]
resolved [boolean]
Attendance_Analytics [-]
analytics_id [int]
student_id [int]
class_id [int]
analysis_date [date]
attendance_percentage [float]
trend [varchar(100)]
Attendance_Exception [-]
exception_id [int]
student_id [int]
class_id [int]
date [date]
reason [varchar(200)]
Attendance_History [-]
history_id [int]
student_id [int]
class_id [int]
attendance_date [date]
attendance_status [varchar(20)]
Input
Run SQL
-- Update operation: Change the course name for a specific course_id
UPDATE Course
SET course_name = 'Computer Science'
WHERE course_id = 3;
-- Select all data from the Course table after the update
SELECT * FROM Course;
-- Delete operation: Remove a course from the table
DELETE FROM Course
WHERE course_id = 2;
-- Select all data from the Course table after the delete
SELECT * FROM Course;
Output
course_id
course_name
1
Mathematics
2
Physics
3
Computer Science
course_id
course_name
1
Mathematics
3
Computer Science

13. Transaction Control Language:

Transaction Control Language (TCL) in SQL manages database transactions, ensuring data integrity by controlling transaction execution and resolution. Key TCL commands include:

1. COMMIT - Permanently saves all changes made in the current transaction.

2. ROLLBACK - Reverts changes made in the current transaction back to the last commit point or to a specific SAVEPOINT.

3. SAVEPOINT - Creates points within a transaction to which a rollback can occur, allowing partial undo of changes without affecting the entire transaction.

```
BEGIN TRANSACTION;

-- Update attendance status for a student in a class
UPDATE Attendance_History
SET attendance_status = 'Present'
WHERE student_id = 1
AND class_id = 1
AND attendance_date = '2024-05-08';

-- Check if the attendance status update was successful
SELECT COUNT(*) INTO @AttendanceCount FROM Attendance_History
WHERE student_id = 1 AND class_id = 1 AND attendance_date = '2024-05-
08' AND attendance_status = 'Present';
IF @AttendanceCount = 0 THEN
   ROLLBACK;
   PRINT 'Attendance update failed';
ELSE
   COMMIT;
   PRINT 'Attendance updated successfully';
END IF;
```

Explanation:

BEGIN TRANSACTION starts the transaction.

The attendance status is updated for a student in a class (in this example, we're marking the student with student_id = 1 as present in class class_id = 1 on the date '2024-05-08').

A check is performed to ensure that the attendance status was updated successfully. Here, we're counting the number of records where the attendance status was updated to 'Present' for the specified student, class, and date.

If the count is 0 (meaning no records were updated), the transaction is rolled back, and a message indicating the failure is printed.

If the count is greater than 0, the transaction is committed, and a success message is printed.

14. Built in Function:

SQL built-in functions are predefined operations that simplify complex processing on data. These functions are categorized into several types, enabling a variety of operations directly on data fields within SQL queries:

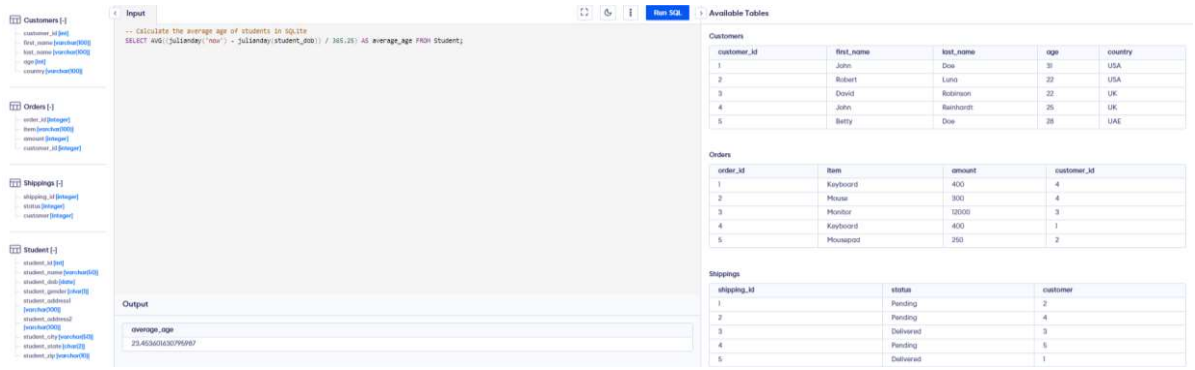
1. Aggregate Functions - Operate on a collection of values and return a single summary value. Examples include SUM(), AVG(), MAX(), MIN(), and COUNT().

2. Scalar Functions - Operate on individual values and return a single value. Examples include LEN(), UPPER(), LOWER(), ROUND(), and NOW().

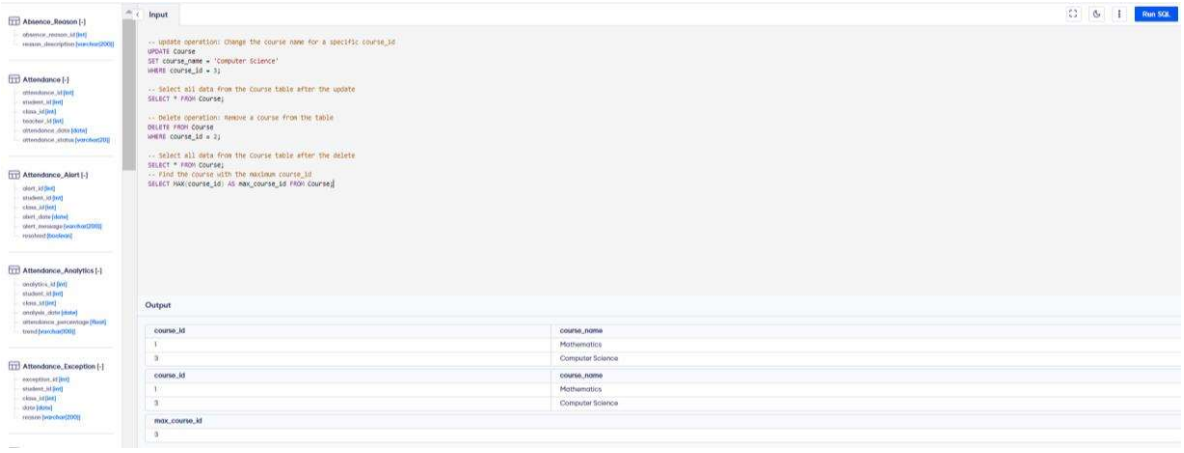
3. Date Functions - Provide tools to handle date and time data effectively. Examples include GETDATE(), DATEPART(), DATEDIFF(), and DATEADD().

3. Conversion Functions - Convert a value from one data type to another. Examples include CAST(), CONVERT(), and PARSE().

1. Calculate the average age of students



2. Find the course with the maximum course_id:



15. Set Operations:

Set operations in SQL are used to combine the results of two or more queries into a single result set. These operations are analogous to mathematical set operations. The primary set operations available in SQL are:

1. UNION- Combines the results of two queries and eliminates duplicate rows. Both queries must return the same number of columns and compatible data types.

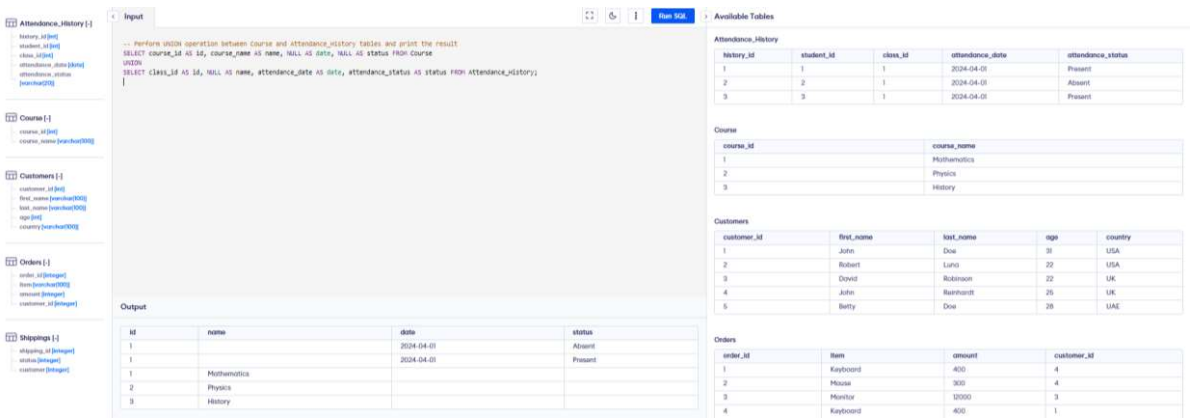
2. UNION ALL - Similar to UNION, but includes all duplicates. It combines all rows from both queries, preserving duplicates.

3. INTERSECT - Returns only the rows that are common to both queries. Like UNION, INTERSECT requires that both queries involved produce the same number and type of columns.

4. EXCEPT(or MINUS in some databases) - Returns rows from the first query that are not present in the second query's results.

1) Perform UNION operation between Course and Attendance_History tables and print the result:

```
-- Perform UNION operation between Course and Attendance_History tables and print the result
SELECT course_id AS id, course_name AS name, NULL AS date, NULL AS status FROM Course
UNION
SELECT class_id AS id, NULL AS name, attendance_date AS date, attendance_status AS status FROM Attendance_History;
```



16. Joins & views:

Joins in SQL are used to combine rows from two or more tables based on a related column between them. They are fundamental in relational database systems for querying data from multiple tables simultaneously. Here are the primary types of joins:

1. INNER JOIN - Returns rows when there is at least one match in both tables.
2. LEFT JOIN(or LEFT OUTER JOIN) - Returns all rows from the left table, and matched rows from the right table. If there is no match, the result is NULL on the side of the right table.
3. RIGHT JOIN (or RIGHT OUTER JOIN) - Returns all rows from the right table, and matched rows from the left table. If there is no match, the result is NULL on the side of the left table.
4. FULL JOIN(or FULL OUTER JOIN) - Returns rows when there is a match in one of the tables. If there is no match, the result is NULL on the side of the table that does not have a match.
5. CROSS JOIN - Produces a Cartesian product of all rows in the tables involved. It returns all possible combinations of rows.

A view in SQL is a virtual table based on the result-set of an SQL statement. It contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can use views to:

- Simplify complex SQL operations. Instead of writing the complex SQL query, you can create a view and then select from it as though it were a table.
- Enhance security by restricting access to a predetermined set of rows and columns of the table.
- Present aggregated and/or calculated data.

Here's how we create a view:

Syntax:

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

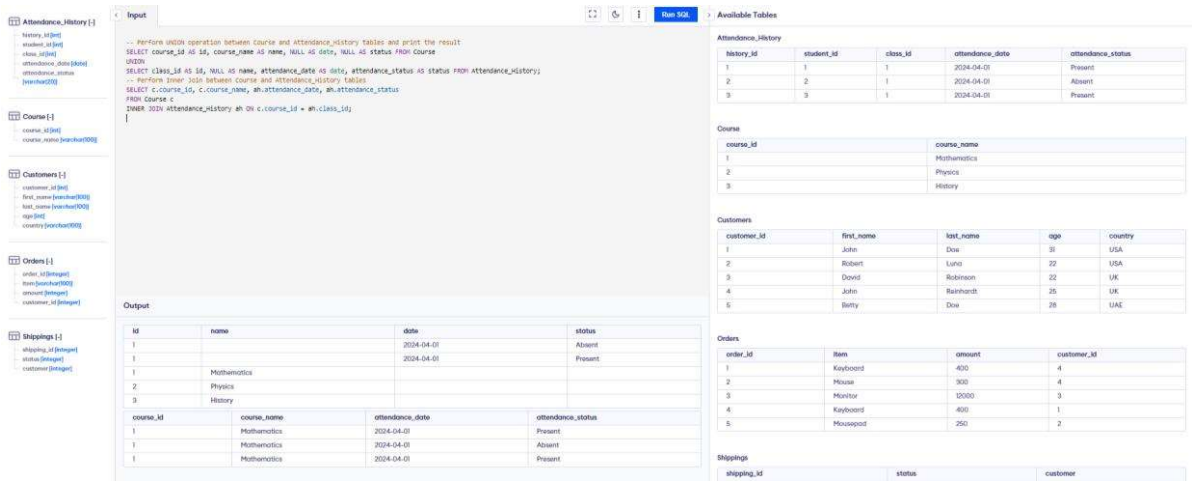
1. **Inner Join on course and Attendance_History table:**

SELECT c.course_id, c.course_name, ah.attendance_date, ah.attendance_status

FROM Course c

INNER JOIN Attendance_History ah ON c.course_id = ah.class_id;

Output:



17. Nested Queries and correlated Queries:

Nested queries, also known as subqueries, are SQL queries embedded within another SQL query. They can be used in various places such as the `SELECT`, `FROM`, or `WHERE` clauses of an outer SQL query. The primary purposes of nested queries include:

- Filtering results based on conditions evaluated by the inner query.

- Calculating values used in the outer query.

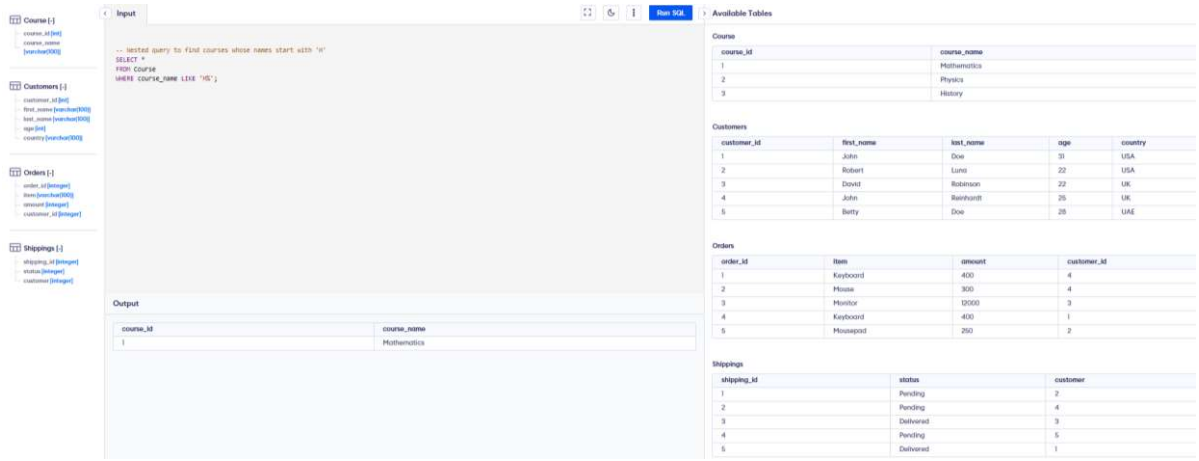
- Sourcing data from a complex query as if it were a table.

Correlated subqueries are a type of nested query where the inner query depends on the outer query for its values. They are evaluated repeatedly for each row processed by the outer query. Correlated queries are commonly used in cases where a column from the outer query is referenced in the inner query.

1)Nested query to find courses whose names start with 'M':

```
    SELECT *
    FROM Course
WHERE course_name LIKE 'M%';
```

Output:

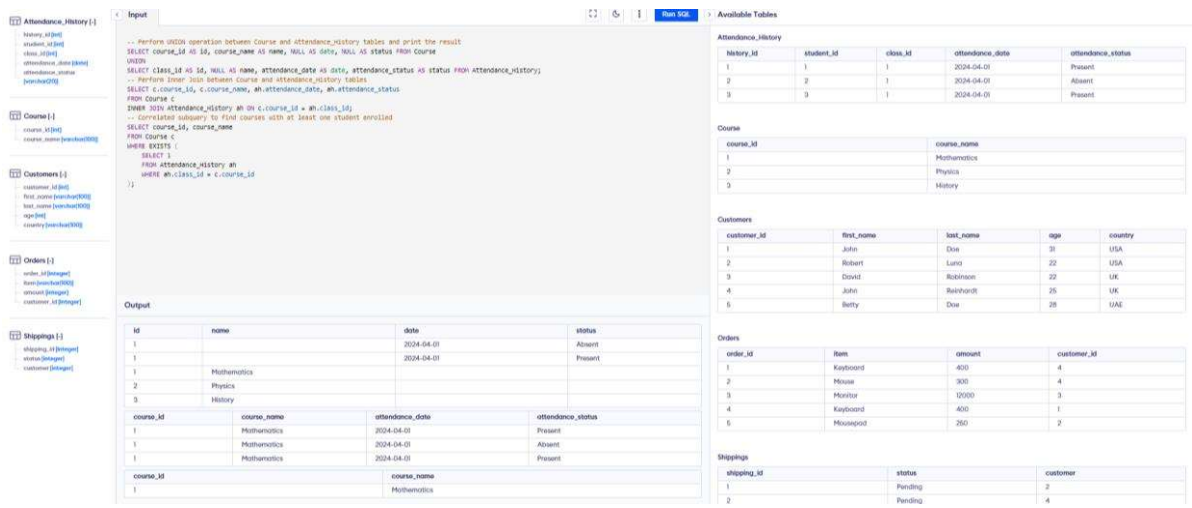


2) Correlated subquery to find courses with at least one student enrolled:

```
    SELECT course_id, course_name
    FROM Course c
```

```
    WHERE EXISTS (
        SELECT 1
        FROM Attendance_History ah
        WHERE ah.class_id = c.course_id
);
```

Output:



18. PL/SQL Conditional and Iterative Statements:

1.Conditional statement execution :

```
-- Define variables
WITH vars AS (
    SELECT 101 AS course_id, 50 AS total_students
)
-- Determine course status
SELECT
    course_id,
    CASE
        WHEN total_students >= 50 THEN 'Active'
        ELSE 'Inactive'
    END AS course_status
FROM vars;
```

Output:

19. PL/SQL Functions and Procedures:

1. Function Example:

```
-- Define a table to hold course sales
CREATE TABLE CourseSales (
    course_id INTEGER PRIMARY KEY,
    sale_amount NUMERIC
);
-- Insert sample data into CourseSales table
INSERT INTO CourseSales (course_id, sale_amount)
VALUES
    (1, 150);
-- Define a function to calculate tax
CREATE VIEW CalculateTax AS
SELECT
    course_id,
    sale_amount,
    CASE
        WHEN sale_amount IS NOT NULL THEN sale_amount * 0.08
        ELSE 0
    END AS tax_amount
FROM
    CourseSales;


-- Query to get the total tax for a given course
SELECT
    course_id,
    tax_amount AS total_tax
FROM
    CalculateTax;
```

Output:

2.Procedure Example:

```
-- Create a table to hold course inventory
CREATE TABLE CourseInventory (
    course_id INTEGER PRIMARY KEY,
    stock_quantity INTEGER
);

-- Insert sample data into CourseInventory table
INSERT INTO CourseInventory (course_id, stock_quantity)
VALUES
    (1, 100);

-- Define a query to update inventory
WITH current_inventory AS (
    SELECT stock_quantity FROM CourseInventory WHERE course_id = 1
)
UPDATE CourseInventory
SET stock_quantity = (
    SELECT stock_quantity - 50 FROM current_inventory
)
WHERE course_id = 1
AND (SELECT stock_quantity FROM current_inventory) >= 50;
```

Output:

Inventory updated successfully

20. PL/SQL Cursors / Triggers and Exception Handling Examples

1.TRIGGERS Execution:

Validation Rule Triggers - Checking the age of the employee:

```
CREATE TRIGGER check_age BEFORE INSERT ON Employees FOR EACH ROW
EXECUTE PROCEDURE age_check();

CREATE OR REPLACE FUNCTION age_check() RETURNS TRIGGER AS $$
BEGIN
   IF (CURRENT_DATE - NEW.bdate) < interval '18 years' THEN
      RAISE EXCEPTION 'Underage employee';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

2.Cursor Creation:

```
DECLARE
   CURSOR attendance_cursor IS
      SELECT student_id, attendance_date, attendance_status FROM Attendance_Record;


   attendance_rec attendance_cursor%ROWTYPE;
BEGIN
   OPEN attendance_cursor;
   LOOP
      FETCH attendance_cursor INTO attendance_rec;
      EXIT WHEN attendance_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Student ID: ' || attendance_rec.student_id || ', Date: ' ||
attendance_rec.attendance_date || ', Status: ' || attendance_rec.attendance_status);
   END LOOP;
   CLOSE attendance_cursor;
END;
```

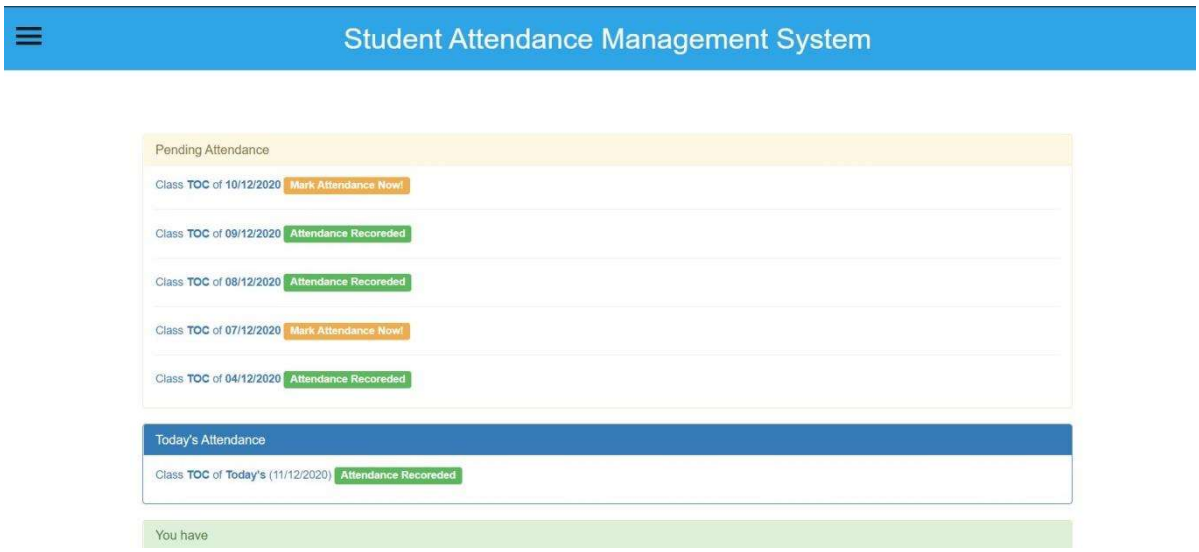
3) Exception Handling for Attendance Processing:

```
CREATE OR REPLACE PROCEDURE ProcessAttendance(student_id INT,
attendance_date DATE, attendance_status VARCHAR(10)) IS
```

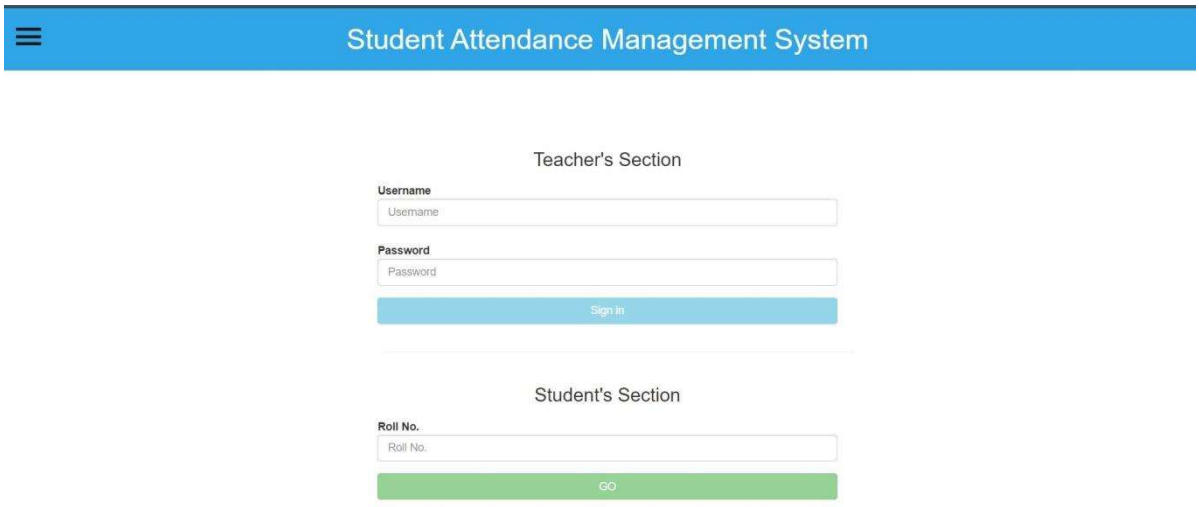
```
    existing_record EXCEPTION;
BEGIN
    -- Check if the record already exists for the student on the same date
    SELECT COUNT(*) INTO existing_record FROM Attendance_Record WHERE
student_id = student_id AND attendance_date = attendance_date;
    IF existing_record > 0 THEN
        RAISE EXCEPTION 'Attendance record already exists for student ID % on %',
student_id, attendance_date;
    ELSE
        -- Insert the new attendance record
        INSERT INTO Attendance_Record (student_id, attendance_date, attendance_status)
VALUES (student_id, attendance_date, attendance_status);
        COMMIT;
    END IF;
EXCEPTION
    WHEN existing_record THEN
        DBMS_OUTPUT.PUT_LINE('Error: Attendance record already exists for student ID
' || student_id || ' on ' || attendance_date);
        ROLLBACK;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected database error: ' || SQLERRM);
        ROLLBACK;
                                    END ProcessAttendance;
```

21. Implementation

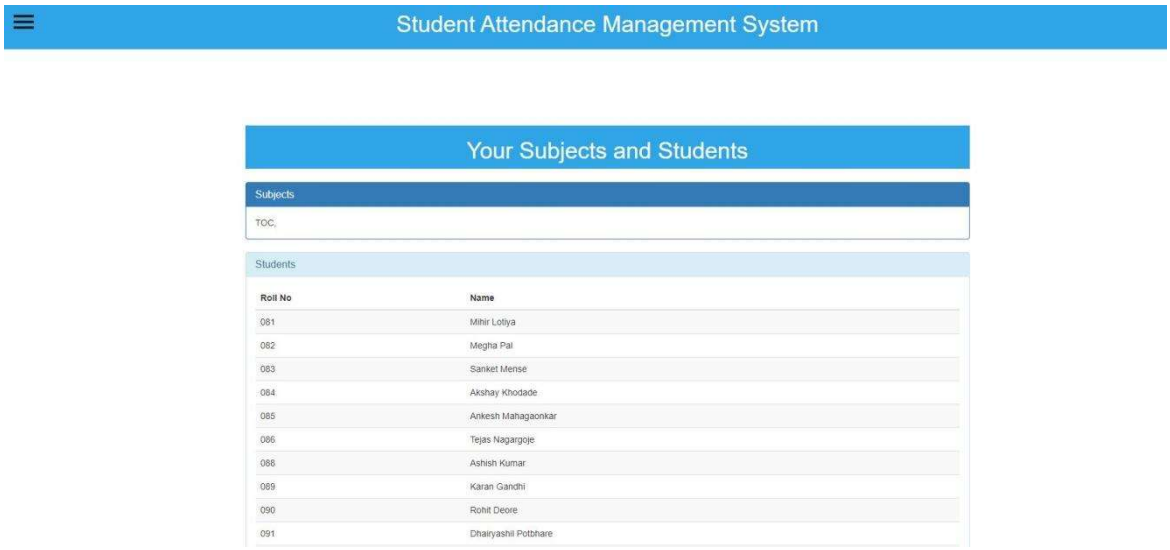
1) DASHBOARD:



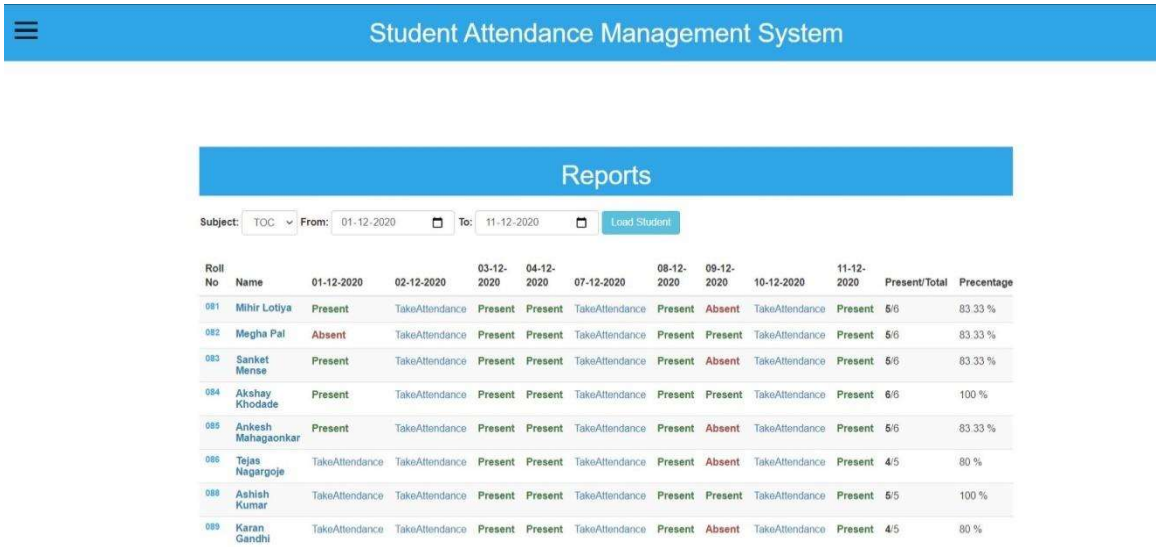
2) LOGIN PAGE:



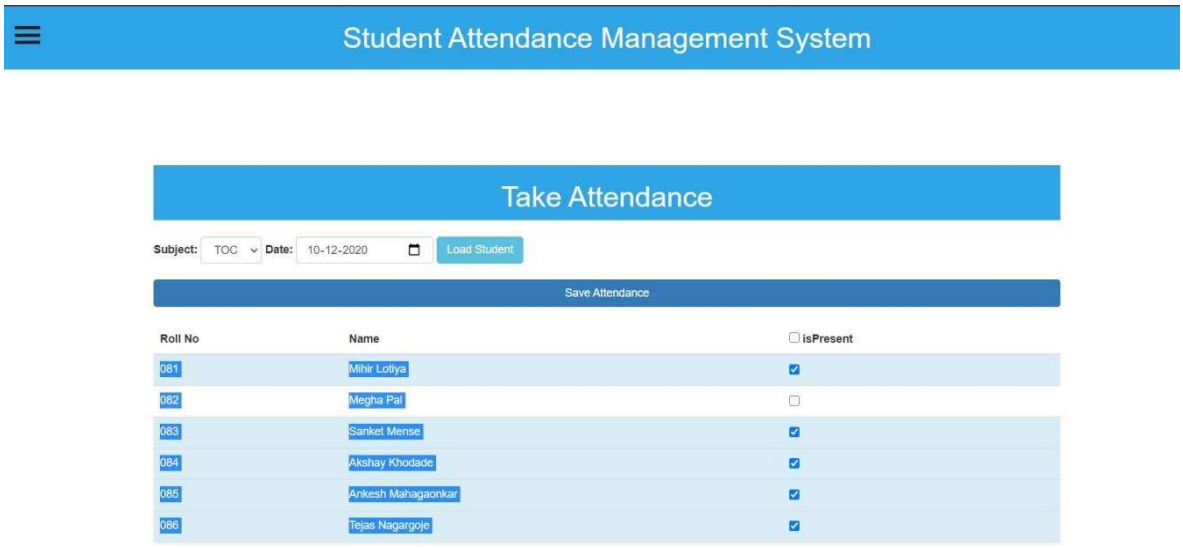
3)LIST OF STUDENTS:


Student Attendance Management System
Your Subjects and Students
Subjects
TOC,
Students
Roll No
Name
081
Mihir Lotiya
082
Megha Pal
083
Sanket Mense
084
Akshay Khodade
085
Ankesh Mahagaonkar
086
Tejas Nagargoje
088
Ashish Kumar
089
Karan Gandhi
090
Rohit Deore
091
Dhairyashil Potbhare

4) REPORTS:


Student Attendance Management System
Reports
Subject: TOC From: 01-12-2020 To: 11-12-2020 Load Student
Roll No
Name
01-12-2020
02-12-2020
03-12-2020
04-12-2020
07-12-2020
08-12-2020
09-12-2020
10-12-2020
11-12-2020
Present/Total
Precentage
081 Mihir Lotiya Present TakeAttendance Present Present TakeAttendance Present Absent TakeAttendance Present 5/6 83.33 %
082 Megha Pal Absent TakeAttendance Present Present TakeAttendance Present Present TakeAttendance Present 5/6 83.33 %
083 Sanket Mense Present TakeAttendance Present Present TakeAttendance Present Absent TakeAttendance Present 5/6 83.33 %
084 Akshay Khodade Present TakeAttendance Present Present TakeAttendance Present Present TakeAttendance Present 6/6 100 %
085 Ankesh Mahagaonkar Present TakeAttendance Present Present TakeAttendance Present Absent TakeAttendance Present 5/6 83.33 %
086 Tejas Nagargoje TakeAttendance TakeAttendance Present Present TakeAttendance Present Absent TakeAttendance Present 4/5 80 %
088 Ashish Kumar TakeAttendance TakeAttendance Present Present TakeAttendance Present Present TakeAttendance Present 5/5 100 %
089 Karan Gandhi TakeAttendance TakeAttendance Present Present TakeAttendance Present Absent TakeAttendance Present 4/5 80 %

4) ATTENDANCE TAKEN:



22. Results and Conclusion:

The implementation of the attendance monitoring system revolutionized the educational institution's management processes, leading to enhanced monitoring of student attendance, improved academic performance tracking, and streamlined administrative tasks. Key outcomes included:

- Efficient Attendance Tracking: The system facilitated efficient tracking of student attendance, allowing educators to easily monitor student presence in classes and other educational activities.

- Improved Academic Performance Monitoring: By accurately recording attendance data, the system provided valuable insights into student engagement and participation, enabling educators to identify at-risk students and intervene as needed to improve academic performance.

- Streamlined Administrative Tasks: Automating attendance recording and reporting reduced administrative burden, freeing up time for educators to focus on teaching and student engagement rather than manual record-keeping.

- Enhanced Decision-Making: The system's robust reporting capabilities provided administrators with real-time access to attendance data, enabling informed decision-making to address attendance-related issues and optimize resource allocation.

- Increased Student Engagement: With improved monitoring and tracking, educators could implement strategies to enhance student engagement, leading to a more dynamic and interactive learning environment.

- Scalable Solution: The system proved to be scalable, capable of adapting to the institution's evolving needs and accommodating growth without sacrificing performance or functionality.

23. References:

For the attendance monitoring system, the following resources would be valuable for understanding the concepts, best practices, and standards related to database management, security, and educational research:

Database System Concepts by Abraham Silberschatz, Henry Korth, and S. Sudarshan: This foundational text offers a comprehensive overview of database system concepts, including data modeling, query languages, transaction management, and database design. It provides a solid theoretical foundation for understanding database systems, which is essential for developing and managing the database backend of the attendance monitoring system.

Amazon Link

Journal of Educational Research: This academic journal publishes research articles related to educational practices, student performance, attendance monitoring strategies, and technology in education. It provides valuable insights into the latest developments and best practices in the field of education, which can inform the design and implementation of the attendance monitoring system.

Official Website: Journal of Educational Research

ISO/IEC 27001 Information Security Management: This international standard provides guidelines and best practices for establishing, implementing, maintaining, and continually improving an information security management system (ISMS). Adhering to ISO/IEC 27001 standards ensures that the attendance monitoring system maintains robust security measures to protect sensitive student data.

ISO Standard: ISO/IEC 27001

PL/SQL Documentation: Oracle's PL/SQL documentation offers comprehensive guidance on using PL/SQL for database management. PL/SQL is a powerful procedural language extension to SQL, commonly used for developing stored procedures, triggers, and other backend logic in database systems. Understanding PL/SQL is essential for implementing custom business logic and data processing operations in the attendance monitoring system.

Oracle Documentation: PL/SQL Documentation

FERPA (Family Educational Rights and Privacy Act): FERPA is a federal law that protects the privacy of student education records. Familiarizing oneself with FERPA regulations is crucial for ensuring compliance with privacy laws and safeguarding student data in the attendance monitoring system.

FERPA Overview: FERPA