**SVKM’s NMIMS**

**School of Technology Management & Engineering**

A.Y. 2023-24

**Course: Database Management Systems**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program | BTECH CE | |
| Semester | SEM-IV | |
| Name of the Project: | School Management System | |
|  | | |
| Details of Project Members: | | |
| Batch | Roll No. | Name |
| B1 | A-179 | Harshita Sai Mogalapu |
| B1 | A-181 | Purva Sakharle |
| Date of Submission: 01/04/2024 | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| A-179 | Harshita Sai Mogalapu | Schema Table, running queries, problem statement, normalization |
| A-181 | Purva Sakharle | ER Diagram, creating tables, adding data, components, learning |

**Github link of your project:**

<https://github.com/harshi1308/School_management_System_dbms/tree/main>

**Project Report**

**School Management System**

**By**

**Harshita Sai M, Roll number: A-179**

**Purva Sakharle, Roll number: A-181**

**Course: DBMS**

**AY:2023-24**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline | 4 |
| **2** | Components of Database Design | 5 |
| **3** | Entity Relationship Diagram | 9 |
| **4** | Relational Model | 10 |
| **5** | Normalization | 11 |
| **6** | SQL Queries | 13 |
| **8** | Project Demonstration | 28 |
| **9** | Self-learning beyond classroom | 31 |
| **10** | Learning from the project | 33 |
| **11** | Challenges faced | 34 |
| **12** | Conclusion | 35 |

**I. Storyline**

In today’s world, schools and universities are facing a data explosion. There's more information than ever before about students, teachers, classes, and all the behind-the-scenes stuff that keeps a school running. This can be overwhelming, but it's also an opportunity. The right database system can make things much easier and even improve learning.

Imagine a system that can handle all this data smoothly, like a well-organized filing cabinet. That's what this new database aims to be. It will store student grades, attendance, and course information, making it a breeze for teachers and administrators to find what they need. But it's not just about filing.

Security is a top priority. This database will have fort-knocks worth of protection to keep everyone's information safe. Only authorized people, like teachers and guidance counsellors, will be able to access certain details. We can think of it like a high-security building with ID card checks.

This database won't just hold information; it will also analyse it. Imagine reports that show how students are doing in class, where a particular program might need improvement, or even how best to allocate resources. This will be like having a crystal ball for school administrators, helping them make the best decisions for students.

The best part? This database can connect with other cool tech tools, like online learning platforms or financial management software. This will make everything run smoother, from online classes to keeping track of the school budget.

In short, this new database will make things more efficient, keep information safe, and even help improve how students learn. It might be a big step towards a more digital and effective future for education.

**II. Components of Database Design**

**Entities:**

Table: Login

Attribute: Login\_ID(PK), Username, Role\_ID, Password

Table: Roles

Attributes: Role\_ID(PK), Role\_Name, Role\_Desc

Table: Student

Attributes: Stud\_ID(PK), Stud\_Name, Stud\_Addr, Stud\_Mobile, Stud\_mail

Table: Employee

Attributes: aadhar (PK), emp\_Name, salary, emp\_mobile, emp\_Mail, Working hours

Table: Teaching

Attributes: Teach\_ID(PK), Subject, no of classes, designation

Table: Non-Teaching

Attributes: ID(PK), designation

Table: Report Card

Attributes: Stud\_ID(PK), marks, phy\_fitness, class\_participation, grade, course\_credits

Table: Attendance

Attributes: Stud\_ID(PK), no\_of\_lecture, no\_of\_lecture\_present, no\_of\_lecture\_absent

Table: Extra\_course

Attributes: course\_id (PK), course\_name, course\_credits

**Relationships, cardinality, participation:**

1. Roles and Login

Relationship: A role is assigned to a login.

Cardinality: One-to-Many (1:M)

Participation:

Roles: Every role must be assigned to at least one login.

Login: A login can have only one role assigned to it.

2. Employee and TeachingStaff

Relationship: An employee can be a teaching staff member.

Cardinality: One-to-One (1:1)

Participation:

Employee: Every employee can be a teaching staff member, but not necessarily.

TeachingStaff: A teaching staff member must be an employee.

3. Employee and NonTeachingStaff

Relationship: An employee can be a non-teaching staff member.

Cardinality: One-to-One (1:1)

Participation:

Employee: Every employee can be a non-teaching staff member, but not necessarily.

NonTeachingStaff: A non-teaching staff member must be an employee.

4. Student and Attendance

Relationship: A student has attendance records.

Cardinality: One-to-Many (1:M)

Participation:

Student: Every student must have attendance records.

Attendance: An attendance record must be associated with a student.

5. Student and ReportCard

Relationship: A student has a report card.

Cardinality: One-to-One (1:1)

Participation:

Student: Every student must have a report card.

ReportCard: A report card must be associated with a student.

6. Student and ExtraCourse

Relationship: A student can enroll in extra courses.

Cardinality: Many-to-Many (M:N)

Participation:

Student: A student can enroll in multiple extra courses.

ExtraCourse: An extra course can have multiple students enrolled.

7. TeachingStaff and ExtraCourse

Relationship: A teaching staff member can teach extra courses.

Cardinality: Many-to-Many (M:N)

Participation:

TeachingStaff: A teaching staff member can teach multiple extra courses.

ExtraCourse: An extra course can be taught by multiple teaching staff members.

8. NonTeachingStaff and ExtraCourse

Relationship: A non-teaching staff member can be involved in extra courses.

Cardinality: Many-to-Many (M:N)

Participation:

NonTeachingStaff: A non-teaching staff member can be involved in multiple extra courses.

ExtraCourse: An extra course can involve multiple non-teaching staff members.

9. Employee and Login

Relationship: An employee can have a login.

Cardinality: One-to-One (1:1)

Participation:

Employee: Every employee can have a login, but not necessarily.

Login: A login must be associated with an employee.

10. Student and Login

Relationship: A student can have a login.

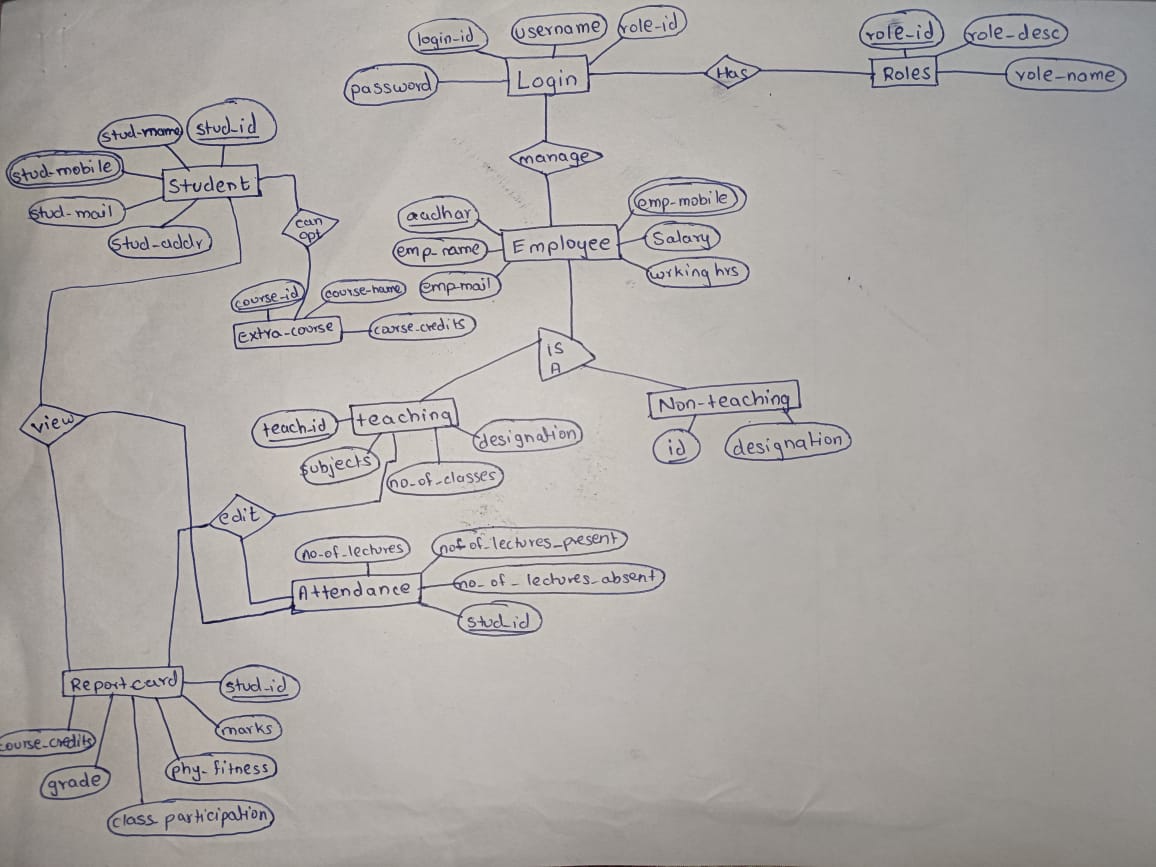
Cardinality: One-to-One (1:1)

Participation:

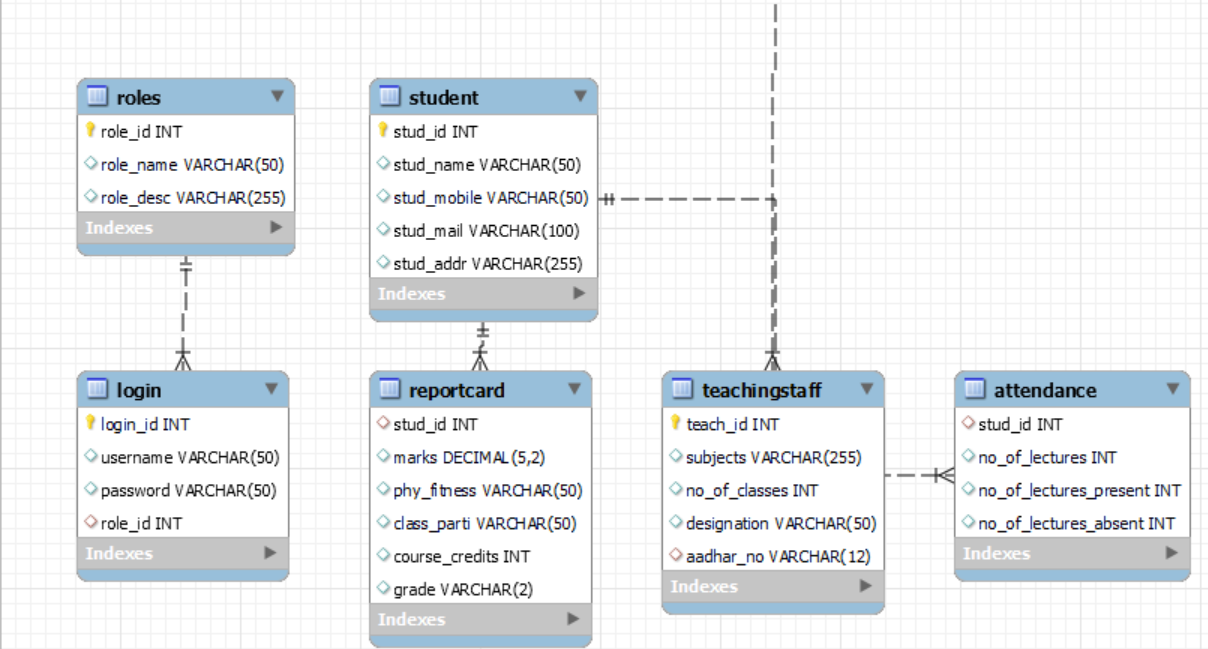
Student: Every student can have a login, but not necessarily.

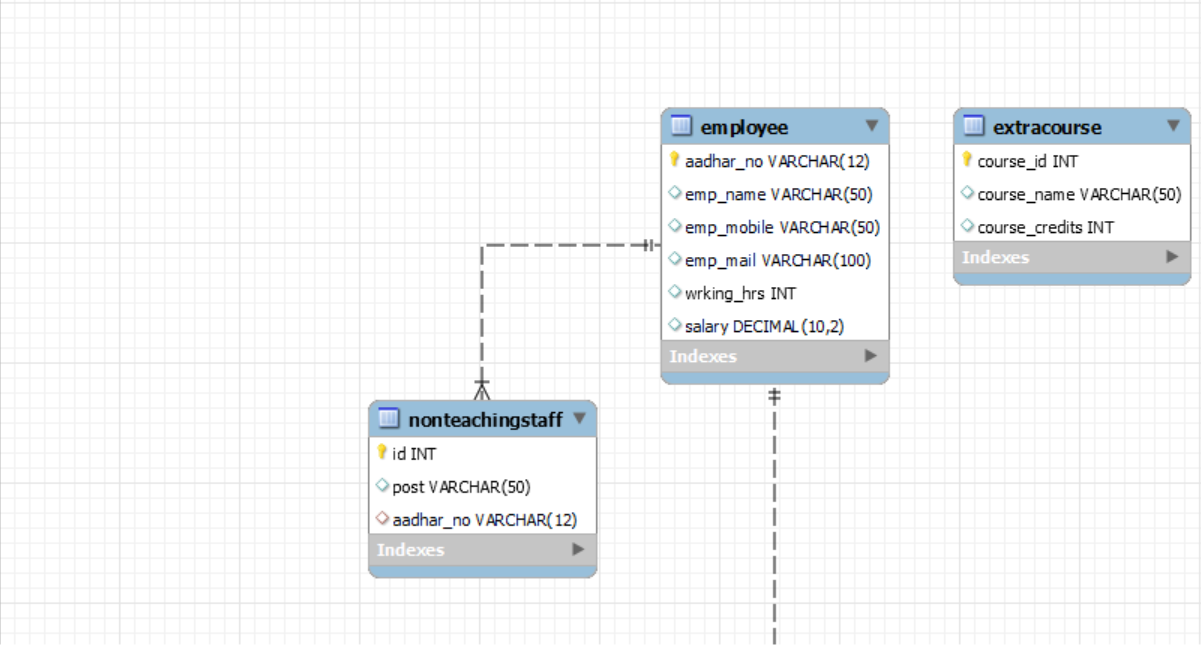
Login: A login must be associated with a student.

**III. Entity Relation Diagram**



**IV. Relational Model**





**V. Normalization**

To determine if the database is in 1NF, 2NF, 3NF, and BCNF, we have to check the following conditions:

1NF (First Normal Form): A table is in 1NF if it contains no repeating groups of data. This means that each column in the table should contain atomic values, and each row should be unique.

2NF (Second Normal Form): A table is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the primary key.

3NF (Third Normal Form): A table is in 3NF if it is in 2NF and there are no transitive dependencies of non-prime attributes on the primary key.

BCNF (Boyce-Codd Normal Form): A table is in BCNF if it is in 3NF and for every one of its dependencies X → Y, X is a superkey.

1NF Check

Roles: Each column contains atomic values, and each row is unique.

Login: Each column contains atomic values, and each row is unique.

Student: Each column contains atomic values, and each row is unique.

Employee: Each column contains atomic values, and each row is unique.

TeachingStaff: Each column contains atomic values, and each row is unique.

NonTeachingStaff: Each column contains atomic values, and each row is unique.

Attendance: Each column contains atomic values, and each row is unique.

ReportCard: Each column contains atomic values, and each row is unique.

ExtraCourse: Each column contains atomic values, and each row is unique.

2NF Check

Roles: The primary key is role\_id, and all other columns (role\_name, role\_desc) are fully functionally dependent on it.

Login: The primary key is login\_id, and all other columns (username, password, role\_id) are fully functionally dependent on it.

Student: The primary key is stud\_id, and all other columns (stud\_name, stud\_mobile, stud\_mail, stud\_addr) are fully functionally dependent on it.

Employee: The primary key is aadhar\_no, and all other columns (emp\_name, emp\_mobile, emp\_mail, wrking\_hrs, salary) are fully functionally dependent on it.

TeachingStaff: The primary key is teach\_id, and all other columns (subjects, no\_of\_classes, designation, aadhar\_no) are fully functionally dependent on it.

NonTeachingStaff: The primary key is id, and all other columns (post, aadhar\_no) are fully functionally dependent on it.

Attendance: The primary key is stud\_id, and all other columns (no\_of\_lectures, no\_of\_lectures\_present, no\_of\_lectures\_absent) are fully functionally dependent on it.

ReportCard: The primary key is stud\_id, and all other columns (marks, phy\_fitness, class\_parti, course\_credits, grade) are fully functionally dependent on it.

ExtraCourse: The primary key is course\_id, and all other columns (course\_name, course\_credits) are fully functionally dependent on it.

Therefore, the database schema is in 1NF and 2NF. Each table meets the criteria for the normal forms, ensuring that the database is well-structured and avoids common anomalies associated with denormalized data. If we further normalize these tables they will no longer be functional.

**VI. SQL Queries**

Query to create tables and insert data into them:

create database project\_dbms;

use project\_dbms;

CREATE TABLE Roles (

role\_id INT PRIMARY KEY,

role\_name VARCHAR(50),

role\_desc VARCHAR(255)

);

CREATE TABLE Login (

login\_id INT PRIMARY KEY,

username VARCHAR(50),

password VARCHAR(50),

role\_id INT,

FOREIGN KEY (role\_id) REFERENCES Roles(role\_id)

);

CREATE TABLE Student (

stud\_id INT PRIMARY KEY,

stud\_name VARCHAR(50),

stud\_mobile VARCHAR(50),

stud\_mail VARCHAR(100),

stud\_addr VARCHAR(255)

);

CREATE TABLE Employee (

aadhar\_no VARCHAR(12) PRIMARY KEY,

emp\_name VARCHAR(50),

emp\_mobile VARCHAR(50),

emp\_mail VARCHAR(100),

wrking\_hrs INT,

salary DECIMAL(10,2)

);

CREATE TABLE TeachingStaff (

teach\_id INT PRIMARY KEY,

subjects VARCHAR(255),

no\_of\_classes INT,

designation VARCHAR(50),

aadhar\_no VARCHAR(12),

FOREIGN KEY (aadhar\_no) REFERENCES Employee(aadhar\_no)

);

CREATE TABLE NonTeachingStaff (

id INT PRIMARY KEY,

post VARCHAR(50),

aadhar\_no VARCHAR(12),

FOREIGN KEY (aadhar\_no) REFERENCES Employee(aadhar\_no)

);

CREATE TABLE Attendance (

stud\_id INT,

no\_of\_lectures INT,

no\_of\_lectures\_present INT,

no\_of\_lectures\_absent INT,

FOREIGN KEY (stud\_id) REFERENCES Student(stud\_id)

);

CREATE TABLE ReportCard (

stud\_id INT,

marks DECIMAL(5,2),

phy\_fitness VARCHAR(50),

class\_parti VARCHAR(50),

course\_credits INT,

grade VARCHAR(2),

FOREIGN KEY (stud\_id) REFERENCES Student(stud\_id)

);

CREATE TABLE ExtraCourse (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50),

course\_credits INT

);

INSERT INTO Roles (role\_id, role\_name, role\_desc) VALUES

(1, 'Student', 'A student in the school'),

(2, 'Teacher', 'A teacher in the school'),

(3, 'Admin', 'An administrator of the school'),

(4, 'Principal', 'The principal of the school'),

(5, 'Vice Principal', 'The vice principal of the school'),

(6, 'Counselor', 'A school counselor'),

(7, 'Librarian', 'A school librarian'),

(8, 'Janitor', 'A school janitor'),

(9, 'Nurse', 'A school nurse'),

(10, 'Cafeteria Staff', 'A cafeteria staff member');

select\* from Roles;

INSERT INTO Student (stud\_id, stud\_name, stud\_mobile, stud\_mail, stud\_addr) VALUES

(101, 'John Doe', '1234567890', 'john.doe@example.com', '123 Main St'),

(102, 'Jane Smith', '0987654321', 'jane.smith@example.com', '456 Elm St'),

(103, 'Emily Davis', '3210987654', 'emily.davis@example.com', '789 Oak St'),

(104, 'Michael Brown', '4567890123', 'michael.brown@example.com', '321 Pine St'),

(105, 'Sarah White', '5678901234', 'sarah.white@example.com', '654 Maple St'),

(106, 'William Green', '6789012345', 'william.green@example.com', '987 Elm St'),

(107, 'Jennifer Black', '7890123456', 'jennifer.black@example.com', '456 Birch St'),

(108, 'Robert Red', '8901234567', 'robert.red@example.com', '789 Cedar St'),

(109, 'Elizabeth Blue', '9012345678', 'elizabeth.blue@example.com', '321 Ash St'),

(110, 'James Yellow', '0123456789', 'james.yellow@example.com', '654 Poplar St');

select\* from Student;

INSERT INTO Employee (aadhar\_no, emp\_name, emp\_mobile, emp\_mail, wrking\_hrs, salary) VALUES

('123456789012', 'Mr. Smith', '1234567890', 'smith@example.com', 40, 50000.00),

('234567890123', 'Ms. Johnson', '2345678901', 'johnson@example.com', 35, 45000.00),

('321098765412', 'Ms. Lee', '3210987654', 'lee@example.com', 38, 55000.00),

('456789012312', 'Mr. Kim', '4567890123', 'kim@example.com', 32, 52000.00),

('567890123412', 'Ms. Park', '5678901234', 'park@example.com', 35, 53000.00),

('678901234512', 'Mr. Choi', '6789012345', 'choi@example.com', 40, 54000.00),

('789012345612', 'Ms. Han', '7890123456', 'han@example.com', 37, 51000.00),

('890123456712', 'Mr. Lee', '8901234567', 'lee2@example.com', 39, 56000.00),

('901234567812', 'Ms. Kim', '9012345678', 'kim2@example.com', 33, 57000.00),

('412345678912', 'Mr. Park', '0123456789', 'park2@example.com', 36, 58000.00),

('123956789012', 'Ms. Choi', '1234567890', 'choi2@example.com', 41, 59000.00),

('234567590123', 'Mr. Han', '2345678901', 'han2@example.com', 34, 60000.00);

select\* from Employee;

INSERT INTO TeachingStaff (teach\_id, subjects, no\_of\_classes, designation, aadhar\_no) VALUES

(201, 'Math, Science', 30, 'Professor', '123456789012'),

(202, 'English, History', 25, 'Assistant Professor', '234567890123'),

(203, 'Chemistry, Biology', 25, 'Associate Professor', '321098765412'),

(204, 'Physics, Astronomy', 20, 'Assistant Professor', '456789012312'),

(205, 'Mathematics, Geometry', 22, 'Professor', '567890123412');

select\* from TeachingStaff;

INSERT INTO NonTeachingStaff (id, post, aadhar\_no) VALUES

(301, 'Cleaner', '678901234512'),

(302, 'Security', '789012345612'),

(303, 'Maintenance', '890123456712'),

(304, 'Security', '901234567812'),

(305, 'Cafeteria Manager', '412345678912');

select\* from NonTeachingStaff;

INSERT INTO Attendance (stud\_id, no\_of\_lectures, no\_of\_lectures\_present, no\_of\_lectures\_absent) VALUES

(101, 30, 28, 2),

(102, 25, 23, 2),

(103, 30, 28, 2),

(104, 25, 23, 2),

(105, 35, 33, 2),

(106, 20, 18, 2),

(107, 22, 20, 2),

(108, 27, 25, 2),

(109, 32, 30, 2),

(110, 28, 26, 2);

select\* from Attendance;

INSERT INTO ReportCard (stud\_id, marks, phy\_fitness, class\_parti, course\_credits, grade) VALUES

(101, 85.5, 'Good', 'Active', 10, 'A'),

(102, 80.0, 'Average', 'Passive', 10, 'B'),

(103, 85.5, 'Good', 'Active', 10, 'A'),

(104, 80.0, 'Average', 'Passive', 10, 'B'),

(105, 90.5, 'Excellent', 'Active', 10, 'A+'),

(106, 75.0, 'Good', 'Passive', 10, 'A-'),

(107, 82.5, 'Average', 'Active', 10, 'B+'),

(108, 77.5, 'Good', 'Passive', 10, 'B-'),

(109, 92.0, 'Excellent', 'Active', 10, 'A+'),

(110, 87.5, 'Good', 'Passive', 10, 'A-');

select\* from ReportCard;

INSERT INTO ExtraCourse (course\_id, course\_name, course\_credits) VALUES

(11, 'Swimming', 5),

(12, 'Reading', 5),

(13, 'Writing', 5),

(14, 'Skating', 5),

(15, 'Art History', 5),

(16, 'Music Theory', 5),

(17, 'Dance', 5),

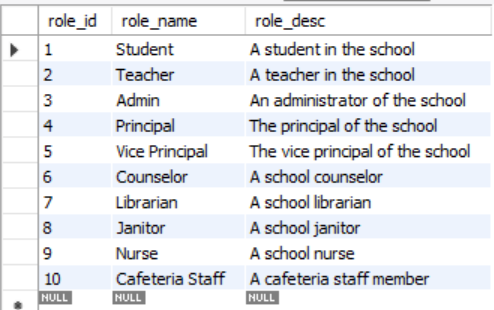
(18, 'Theater', 5),

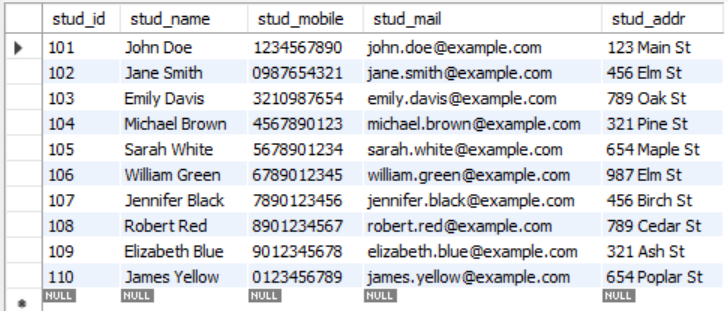
(19, 'Photography', 5),

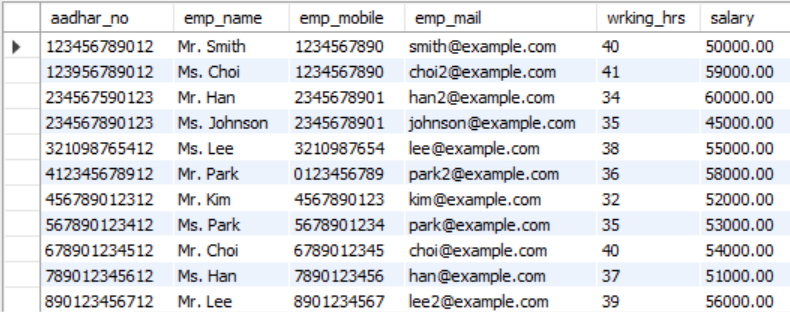
(20, 'Film Studies', 5);

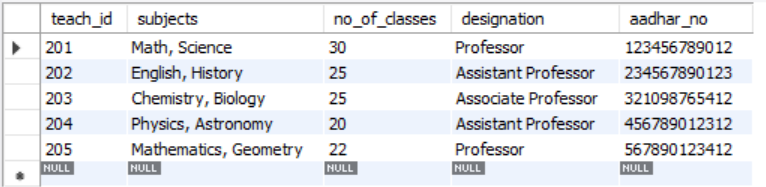
select\* from ExtraCourse;

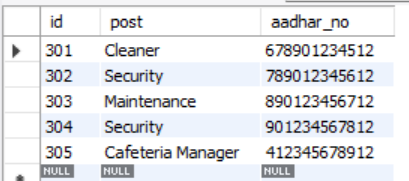
Output of tables:

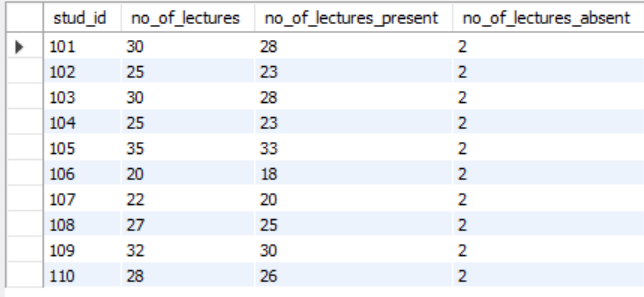


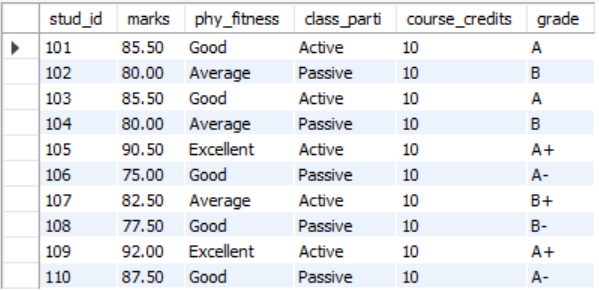


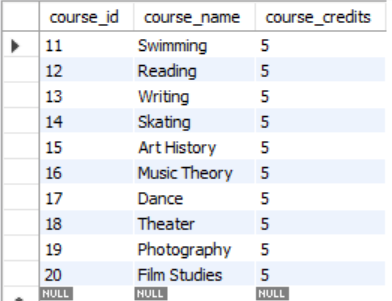












Queries executed on the database with their purpose and output:

1. Count the number of students in the database

SELECT COUNT(\*) FROM Student;



1. Find the average salary of employees

SELECT AVG(salary) FROM Employee;

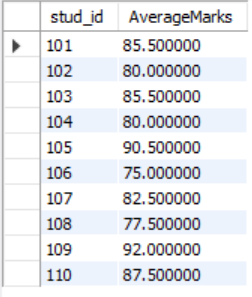


1. List all students with their average marks

SELECT stud\_id, AVG(marks) as AverageMarks

FROM ReportCard

GROUP BY stud\_id;

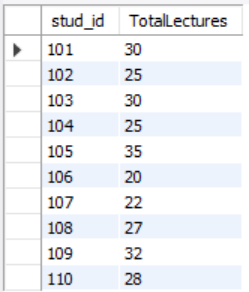


1. Find the total number of lectures attended by each student

SELECT stud\_id, SUM(no\_of\_lectures) as TotalLectures

FROM Attendance

GROUP BY stud\_id;

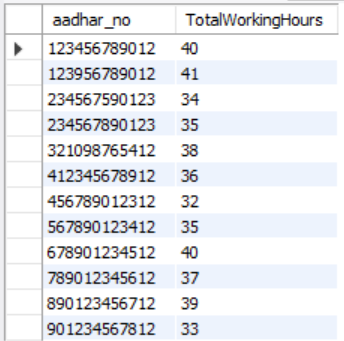


1. List all employees with their total working hours

SELECT aadhar\_no, SUM(wrking\_hrs) as TotalWorkingHours

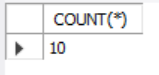
FROM Employee

GROUP BY aadhar\_no;



1. Find the total number of courses offered

SELECT COUNT(\*) FROM ExtraCourse;



1. Update the salary of an employee

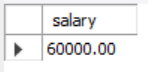
UPDATE Employee

SET salary = 60000.00

WHERE aadhar\_no = '123456789012';

select salary from Employee

WHERE aadhar\_no = '123456789012';



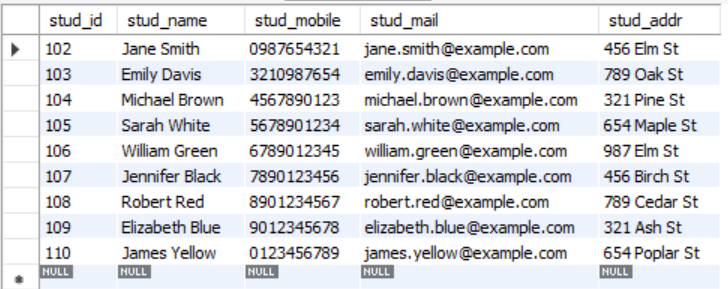
1. Delete a student from the database

DELETE FROM attendance WHERE stud\_id = 101;

DELETE FROM reportcard WHERE stud\_id = 101;

DELETE FROM Student WHERE stud\_id = 101;

select\* from Student;



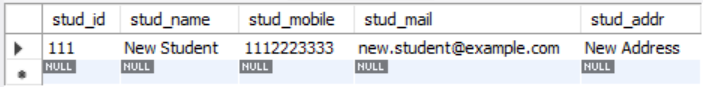
1. Insert a new student into the database

INSERT INTO Student (stud\_id, stud\_name, stud\_mobile, stud\_mail, stud\_addr)

VALUES (111, 'New Student', '1112223333', 'new.student@example.com', 'New Address');

select\* from Student

WHERE stud\_id = 111;



1. Create a new table for student grades

CREATE TABLE StudentGrades (

stud\_id INT,

course\_id INT,

grade VARCHAR(2),

FOREIGN KEY (stud\_id) REFERENCES Student(stud\_id),

FOREIGN KEY (course\_id) REFERENCES ExtraCourse(course\_id)

);

select\* from StudentGrades;

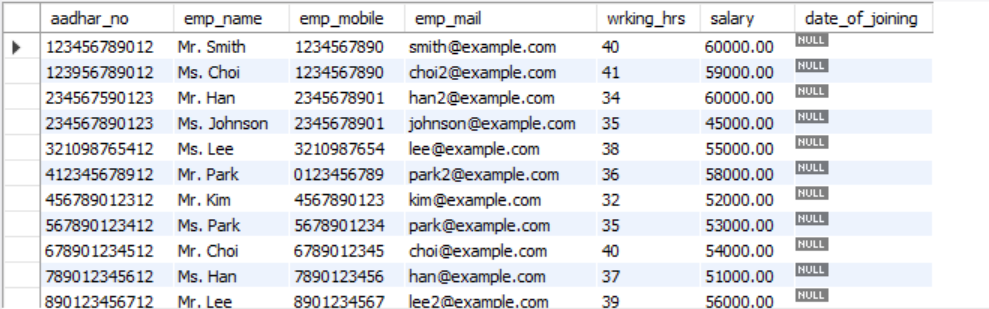


1. Add a new column to the Employee table for date of joining

ALTER TABLE Employee

ADD COLUMN date\_of\_joining DATE;

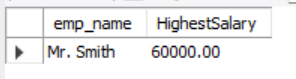
select\* from employee;



1. Find the highest paid employee

SELECT emp\_name, MAX(salary) as HighestSalary

FROM Employee;



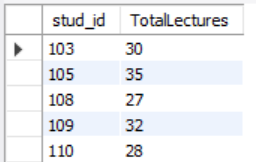
1. List all students who have attended more than 25 lectures

SELECT stud\_id, SUM(no\_of\_lectures) as TotalLectures

FROM Attendance

GROUP BY stud\_id

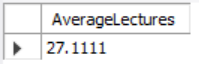
HAVING SUM(no\_of\_lectures) > 25;



1. Find the average number of lectures attended per student

SELECT AVG(no\_of\_lectures) as AverageLectures

FROM Attendance;

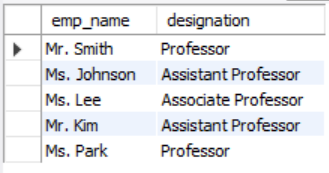


1. List all employees with their designation

SELECT emp\_name, designation

FROM Employee

JOIN TeachingStaff ON Employee.aadhar\_no = TeachingStaff.aadhar\_no;



1. Update the mobile number of a student

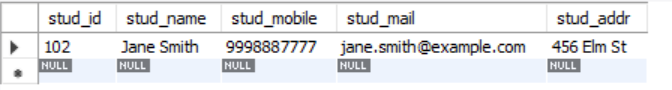
UPDATE Student

SET stud\_mobile = '9998887777'

WHERE stud\_id = 102;

select\* from Student

WHERE stud\_id = 102;

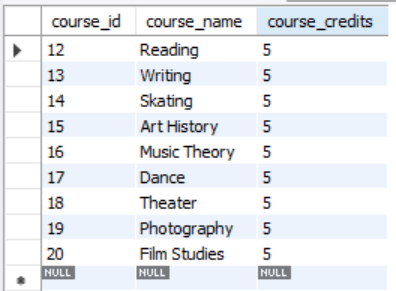


1. Delete a course from the database

DELETE FROM ExtraCourse

WHERE course\_id = 11;

select\* from extracourse;

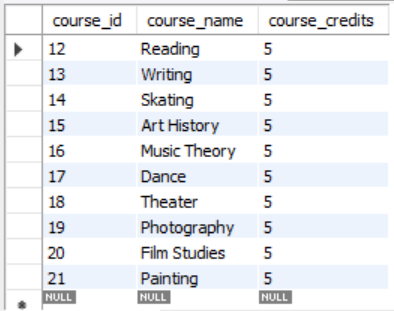


1. Insert a new course into the database

INSERT INTO ExtraCourse (course\_id, course\_name, course\_credits)

VALUES (21, 'Painting', 5);

select\* from extracourse;



1. Drop the StudentGrades table

DROP TABLE StudentGrades;

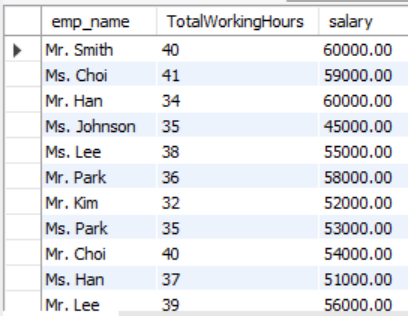


1. List all employees with their total working hours and salary

SELECT emp\_name, SUM(wrking\_hrs) as TotalWorkingHours, salary

FROM Employee

GROUP BY emp\_name, salary;

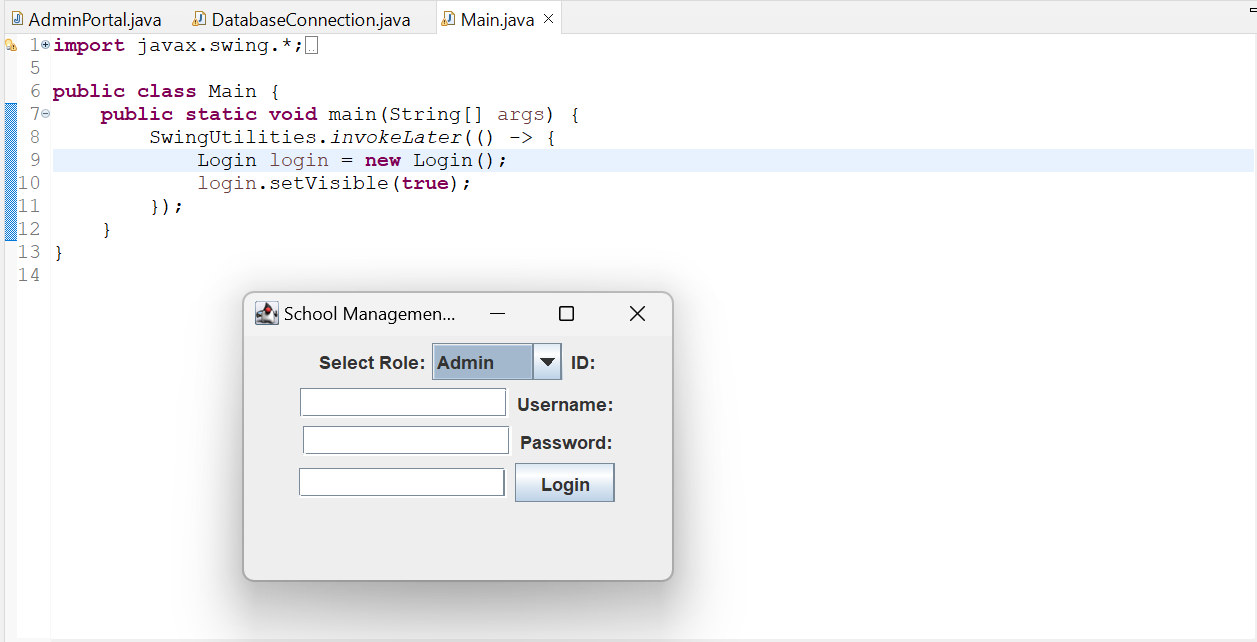


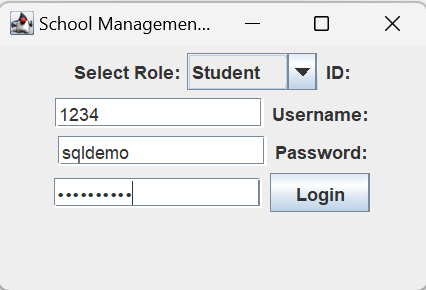
**VII. Project Demonstration**

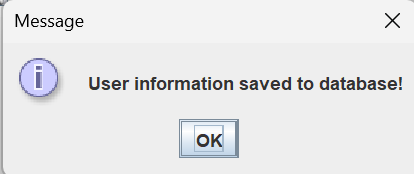
Tools/Software/libraries used:

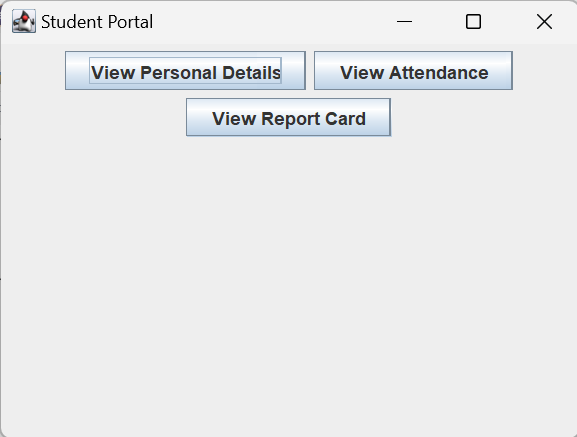
* Mysql workbench
* Eclipse

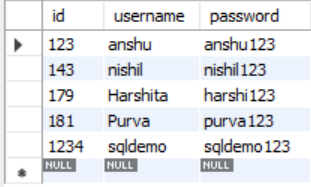
Demonstration:











Our project aims to develop a comprehensive school management system using Java. We've begun by connecting one of the tables, the "user" table, to our Java code. This table stores login credentials for the entire system. This initial step demonstrates our ability to bridge our database with a Java backend.

Moving forward, we plan to expand this connection to encompass additional school data. By building upon this foundation, our system will manage various aspects of school operations. Imagine features like student information, class schedules, attendance tracking, and even grade management – all accessible through a user-friendly interface.

This Java-based system promises to streamline school administration, improve efficiency, and enhance communication among teachers, parents, and students. We're excited to develop this project further and contribute to a more organized and effective school management experience.

**VIII. Self-Learning beyond classroom**

1. Technical Skills Development

* Database Management: Gain hands-on experience with database management systems (DBMS), including designing schemas, writing SQL queries, and optimizing database performance.
* Web Development: Learn front-end and back-end web development technologies, such as HTML, CSS, JavaScript, and server-side languages like PHP or Python.
* Software Development Life Cycle (SDLC): Understand the SDLC, including requirements gathering, design, implementation, testing, and deployment.

2. Problem-Solving Skills

* Complex Problem Solving: Develop your ability to break down complex problems into manageable parts and find effective solutions.
* Critical Thinking: Improve your critical thinking skills by analyzing different approaches to solving problems and choosing the most efficient and effective solutions.

3. Project Management

* Project Planning: Learn how to plan and manage projects, including setting timelines, allocating resources, and tracking progress.
* Teamwork and Collaboration: Enhance your teamwork and collaboration skills by working with others to achieve project goals.
* Communication: Improve your communication skills by effectively conveying ideas and information to team members and stakeholders.

4. Ethical and Professional Considerations

* Data Privacy and Security: Understand the importance of data privacy and security in educational settings and learn how to implement measures to protect sensitive information.
* Ethical Considerations: Reflect on the ethical considerations involved in developing and using educational technology, including issues of fairness, accessibility, and inclusivity.

5. Networking and Professional Development

* Networking: Connect with other professionals in the field of education technology, which can open up opportunities for collaboration, mentorship, and professional development.
* Professional Development: Consider participating in professional development programs or certifications related to education technology to enhance your skills and knowledge.

**IX. Learning from the project**

1. Technical Proficiency

* Database Management: Gain deep expertise in database management systems (DBMS), including designing schemas, writing SQL queries, and optimizing database performance.
* Web Development: Acquire skills in both front-end and back-end web development technologies, such as HTML, CSS, JavaScript, and server-side languages like PHP or Python.
* Software Development Life Cycle (SDLC): Understand the SDLC, including requirements gathering, design, implementation, testing, and deployment.

2. Problem-Solving and Critical Thinking

* Complex Problem Solving: Develop your ability to tackle complex problems by breaking them down into manageable parts and finding effective solutions.
* Critical Thinking: Enhance your critical thinking skills by analyzing different approaches to solving problems and choosing the most efficient and effective solutions.

3. Project Management Skills

* Project Planning: Learn how to plan and manage projects, including setting timelines, allocating resources, and tracking progress.
* Teamwork and Collaboration: Improve your teamwork and collaboration skills by working with others to achieve project goals.
* Communication: Enhance your communication skills by effectively conveying ideas and information to team members and stakeholders.

4. Continuous Learning and Adaptability

* Technology Trends: Stay updated with the latest trends in technology and education, as the field is rapidly evolving.
* Feedback and Improvement: Learn from feedback and continuously seek ways to improve your project and your skills.

**X. Challenges Faced**

Developing a School Management Project in DBMS (Database Management System) involves several challenges, ranging from technical to logistical and managerial. Here are some of the key challenges that might be encountered during the project:

Complexity of Requirements:

School management systems encompass a wide range of functionalities, including user management, student records, attendance tracking, grades management, and more. Managing the complexity of these requirements can be challenging, especially when ensuring that all stakeholders' needs are met.

**XI. Conclusion**

In conclusion, the School Management Project in DBMS has been a comprehensive exploration into the practical application of database management systems (DBMS) in the context of a school environment. This project has provided valuable insights into the design, implementation, and management of a school database, covering various aspects such as user management, role management, student management, employee management, class and subject management, and feedback management.

Key Takeaways from the Project:

1. Database Design and Implementation

2. User Management and Role-Based Access Control

3. Data Integrity and Referential Integrity

4. Query Optimization and Performance

5. Real-World Application of DBMS Concepts

6. Collaboration and Teamwork

In summary, the School Management Project in DBMS has been a valuable learning experience, providing a deep understanding of the principles and practices of database management. It has equipped participants with the skills and knowledge necessary to design, implement, and manage databases effectively, preparing them for real-world challenges in the field of database management.