PREMIER UNIVERSITY CHATTOGRAM

DEPARTMENT OF **COMPUTER SCIENCE AND ENGINEERING**

COMPLEX ENGINEERING PROBLEM

COURSE NAME	Database Management System
COURSE CODE	CSE 2221
ASSIGNMENT TOPIC	Designing a Database for the School Result Management System.
DATE OF ASSIGNMENT	6-04-2024
DATE OF SUBMISSION	7-09-2024
SUBMITTED TO	
ASIF SAAD **LECTURER** **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**	

REMARKS	**SUBMITTED BY**	
	NAME	Rimjhim Dey
	ID	0222220005101039
	SEMESTER	4th
	BATCH	42
	SESSION	Spring 2024
	SECTION	A

Introduction:

In this assignment, our goal was to implement a robust database schema for a School Result Management System. This system is designed to streamline the management of student information, academic results, and administrative tasks within an educational institution. By leveraging relational database principles, we aimed to address the diverse needs of stakeholders such as students, teachers, administrators, and support staff involved in academic operations.

Throughout this report, we present a detailed structure of the database schema, focusing on essential tables and their attributes. The design emphasizes efficient data management, relationship integrity, and scalability to support ongoing academic activities and administrative processes effectively.

Objectives:

➢ Analyze Requirements

The key stakeholders of the system include students, teachers, administrators, and parents. Each stakeholder has specific requirements:

- Students need access to their academic records and results.
- Teachers require functionalities to manage courses, assessments, and student results.
- Administrators need to oversee the entire system, manage users, and generate reports.
- Parents require access to their children's academic performance and communication with teachers.

- **Develop a Normalized Database Schema**

The database schema is designed to be normalized to ensure data integrity, reduce redundancy, and improve query performance. The schema includes tables for users, roles, classes, students, teachers, subjects, exams, and exam results.

Evaluation:

A relational database was chosen for the School Result Management System (SRMS) due to several key advantages:

- **Efficient Management of Relationships:** Relational databases effectively manage complex relationships among students, classes, subjects, and exams. This structure ensures organized data storage and streamlined access.
- **Enhanced Data Integrity:** By enforcing referential integrity through foreign keys, relational databases maintain data consistency and accuracy, minimizing errors and redundancies.
- **Powerful Query Capabilities:** Structured Query Language (SQL) supports advanced querying and data manipulation, enabling efficient retrieval of specific information and generation of reports crucial for academic analysis.
- **Scalability and Performance:** Relational databases are scalable and optimized for performance, handling increasing data volumes and user demands without compromising speed or efficiency.

- **Advantages Over File Systems or Spreadsheets:**

- **Data Integrity:** Ensures consistent and reliable data management compared to file systems.
- **Concurrent Access:** Supports simultaneous access by multiple users, facilitating collaboration and real-time updates.

- **Advanced Reporting:** Enables complex reporting and data analysis, aiding informed decision-making and strategic planning.
- **Data Security:** Provides robust security measures to protect sensitive student information and maintain compliance with privacy regulations.

Investigation:

➢ Features Implemented

- **User Roles and Permissions:** Different access levels are provided for administrators, teachers, students, and parents to ensure data security and appropriate access control.

- **Student Management:** Store and manage student data securely, including personal information, academic history, performance, and attendance records.

- **Teacher Management:** Maintain detailed records of teacher information, including their qualifications, department, and assigned classes and subjects.

- **Class and Subject Management:** Efficiently organize classes and subjects, assign teachers to classes, and manage subject-related information.

- **Exam Management:** Schedule exams, track exam details, manage question papers, and record exam results.

- **Reporting and Analytic:** Generate comprehensive reports on student performance, class averages, subject performance, and other relevant metrics. Use this data to identify areas for improvement and provide personalized feedback to students.

- **Parent Portal:** Provide parents with secure access to their child's academic performance, attendance records, and other relevant information. Allow parents to communicate with teachers and administrators.

- **Student Portal:** Enable students to securely access their academic results, exam schedules, attendance records, and communicate with teachers.

- **Administrative Dashboard:** Allow administrators to monitor overall school performance, manage users, classes, subjects, exams, and generate various reports. Facilitate efficient management of school operations and decision-making.

Design:

++ Relationship Descriptions

➢ One-to-One Relationships

Users and Students:

- Each user in the users table can be associated with one student in the students table, and each student can be associated with one user.
- **Foreign Key:** students.user_id references users.user_id.

Users and Teachers:

- Each user in the users table can be associated with one teacher in the teachers table, and each teacher can be associated with one user.
- **Foreign Key:** teachers.user_id references users.user_id.

Users and Administrators:

- Each user in the users table can be associated with one administrator in the administrators table, and each administrator can be associated with one user.
- **Foreign Key:** administrators.user_id references users.user_id.

➢ One-to-Many Relationships

Roles and Users:

- Each role in the roles table can be associated with multiple users, but each user can have only one role.
- **Foreign Key:** Not explicitly defined but conceptually each user will have a role.

Classes and Students:

- Each class in the classes table can have multiple students, but each student belongs to only one class.
- **Foreign Key:** students.class_id references classes.class_id.

Teachers and Classes:

- Each teacher in the teachers table can be a class teacher for multiple classes, but each class can have only one class teacher.
- **Foreign Key:** classes.class_teacher_id references teachers.teacher_id.

Teachers and Subjects:

- Each teacher in the teachers table can teach multiple subjects, but each subject is taught by only one teacher.
- **Foreign Key:** subjects.teacher_id references teachers.teacher_id.

Subjects and Exams:

- Each subject in the subjects table can have multiple exams, but each exam is associated with only one subject.
- **Foreign Key:** exams.subject_id references subjects.subject_id.

Exams and Exam Results:

- Each exam in the exams table can have multiple exam results, but each exam result is associated with only one exam.
- **Foreign Key:** exam_results.exam_id references exams.exam_id.

Students and Exam Results:

- Each student in the students table can have multiple exam results, but each exam result is associated with only one student.
- **Foreign Key:** exam_results.student_id references students.student_id.

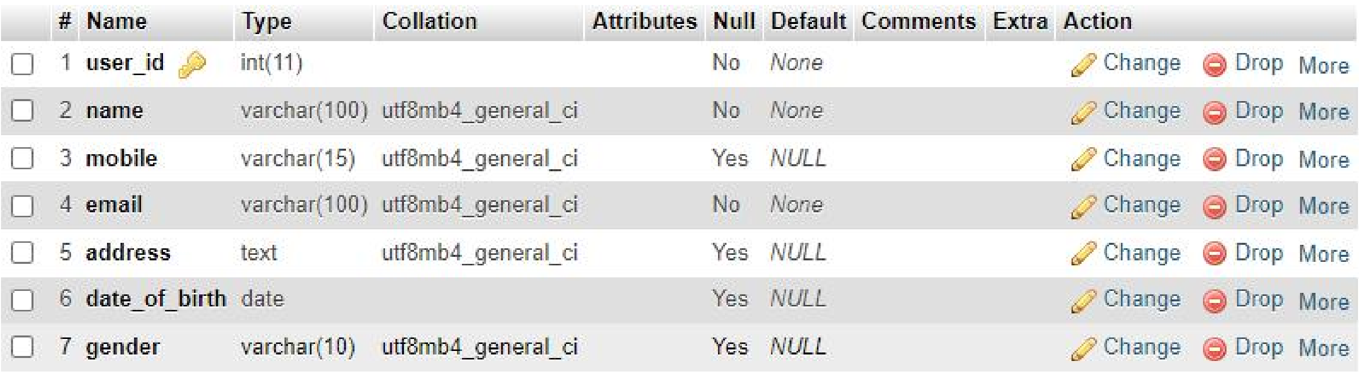
➢ Tables, Attributes & Database Schema

- ## User Table

➢ Query for creating the User table

```
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    mobile VARCHAR(15),
    email VARCHAR(100) NOT NULL,
    address TEXT,
    date_of_birth DATE,
    gender VARCHAR(10)
);
```

➢ Table Structure



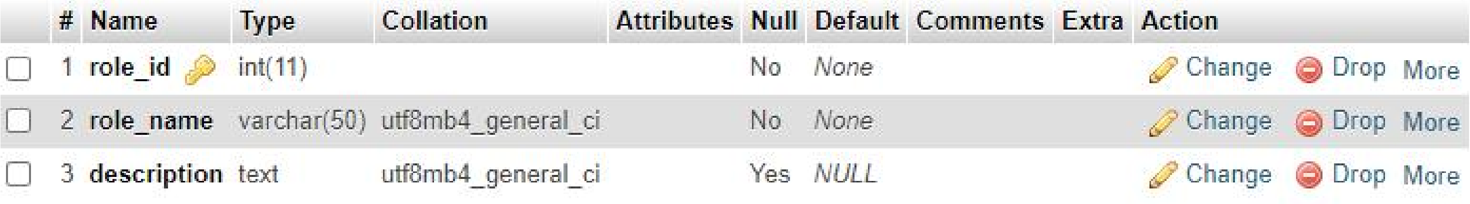
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**user_id** 🔑	int(11)			No	*None*			Change	Drop	More
□	2	**name**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	3	**mobile**	varchar(15)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	4	**email**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	5	**address**	text	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	6	**date_of_birth**	date			Yes	*NULL*			Change	Drop	More
□	7	**gender**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More

- **Roles Table**

➢ Query for creating the Roles table

```
CREATE TABLE roles (
    role_id INTEGER PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL,
       description TEXT
);
```

➢ Table Structure



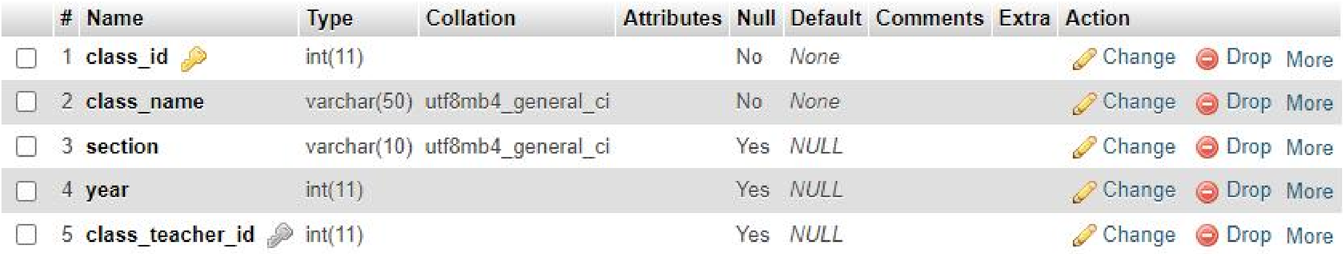
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**role_id** 🔑	int(11)			No	*None*			Change Drop More
☐	2	**role_name**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**description**	text	utf8mb4_general_ci		Yes	*NULL*			Change Drop More

- **Classes Table**

➢ Query for creating the Classes table

```
CREATE TABLE classes (
    class_id INTEGER PRIMARY KEY,
    class_name VARCHAR(50) NOT NULL,
    section VARCHAR(10),
    year INTEGER,
    class_teacher_id INTEGER,
    FOREIGN KEY (class_teacher_id) REFERENCES teachers(teacher_id)
);
```

➢ Table Structure



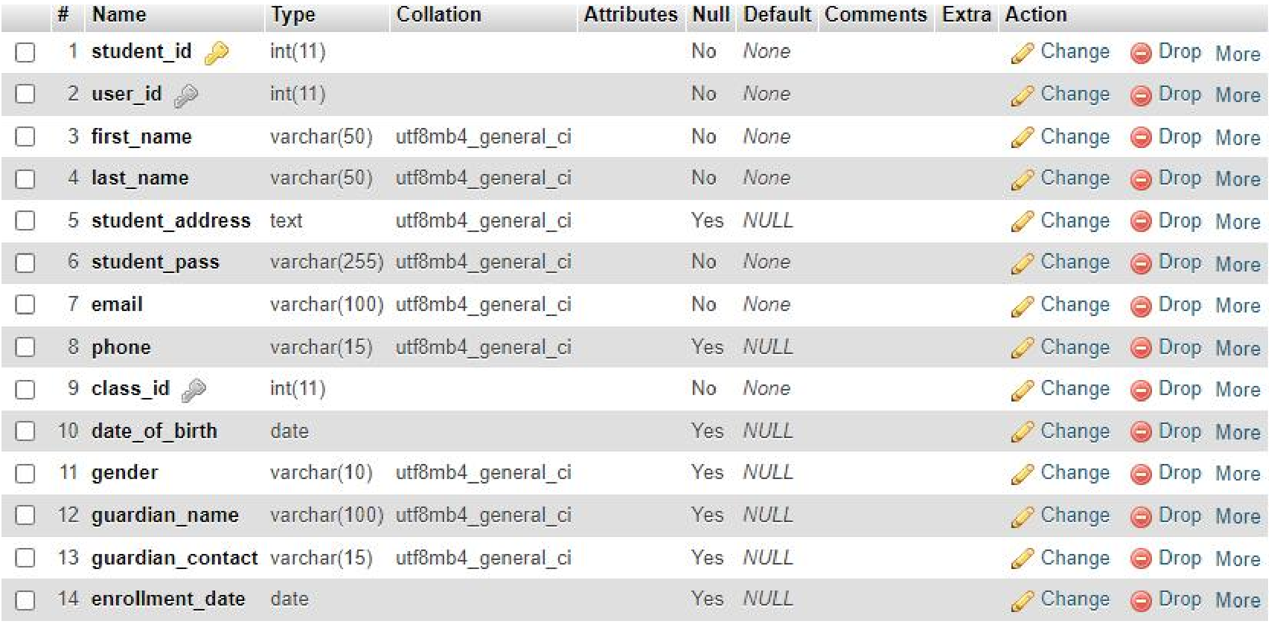
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**class_id**	int(11)			No	*None*			Change Drop More
☐	2	**class_name**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**section**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
☐	4	**year**	int(11)			Yes	*NULL*			Change Drop More
☐	5	**class_teacher_id**	int(11)			Yes	*NULL*			Change Drop More

- ## Student Table

➢ Query for creating the Student table

```
CREATE TABLE students (
    student_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    student_address TEXT,
    student_pass VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15),
    class_id INTEGER NOT NULL,
    date_of_birth DATE,
    gender VARCHAR(10),
    guardian_name VARCHAR(100),
    guardian_contact VARCHAR(15),
    enrollment_date DATE,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (class_id) REFERENCES classes(class_id)
);
```

➢ Table Structure



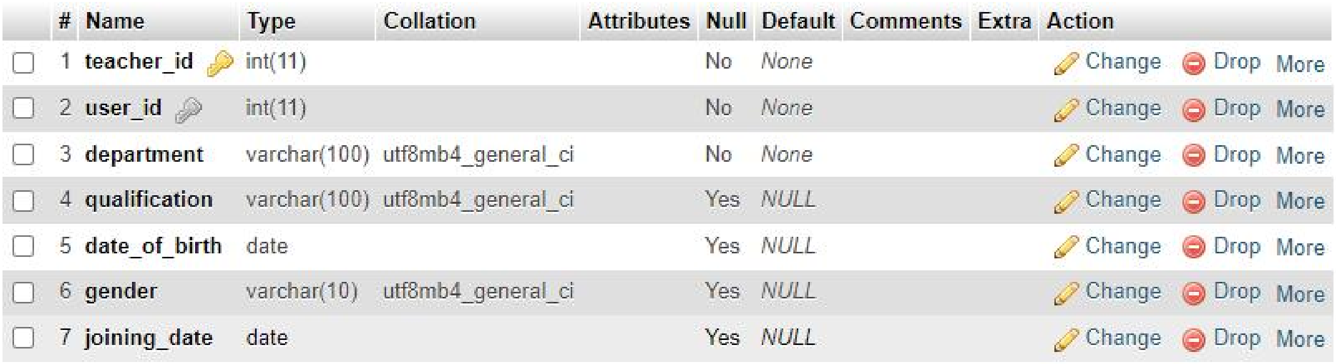
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**student_id** 🔑	int(11)			No	*None*			Change	Drop	More
□	2	**user_id** 🔑	int(11)			No	*None*			Change	Drop	More
□	3	**first_name**	varchar(50)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	4	**last_name**	varchar(50)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	5	**student_address**	text	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	6	**student_pass**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	7	**email**	varchar(100)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	8	**phone**	varchar(15)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	9	**class_id** 🔑	int(11)			No	*None*			Change	Drop	More
□	10	**date_of_birth**	date			Yes	*NULL*			Change	Drop	More
□	11	**gender**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	12	**guardian_name**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	13	**guardian_contact**	varchar(15)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
□	14	**enrollment_date**	date			Yes	*NULL*			Change	Drop	More

- ## Teachers Table

➢ Query for creating the Teachers table

```
CREATE TABLE teachers (
    teacher_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    department VARCHAR(100) NOT NULL,
    qualification VARCHAR(100),
    date_of_birth DATE,
    gender VARCHAR(10),
    joining_date DATE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

➢ Table Structure



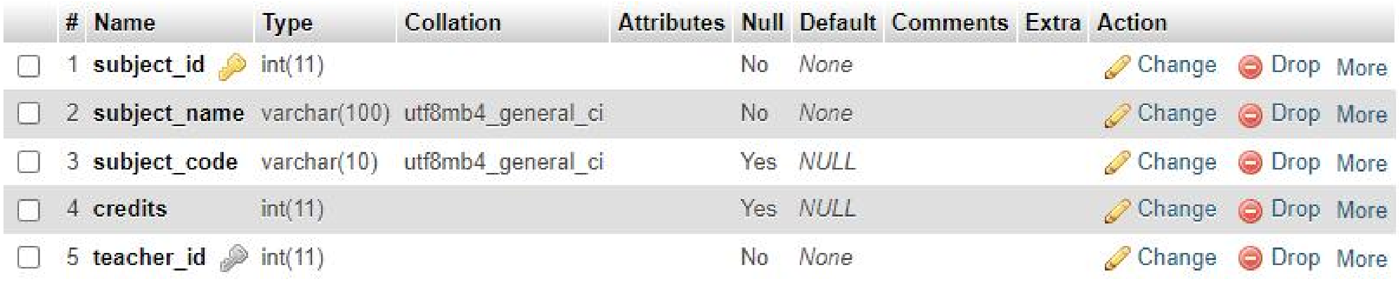
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**teacher_id**	int(11)			No	*None*			Change Drop More
□	2	**user_id**	int(11)			No	*None*			Change Drop More
□	3	**department**	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
□	4	**qualification**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	5	**date_of_birth**	date			Yes	*NULL*			Change Drop More
□	6	**gender**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	7	**joining_date**	date			Yes	*NULL*			Change Drop More

• Subjects Table

➢ Query for creating the Subject table

```
CREATE TABLE subjects (
    subject_id INTEGER PRIMARY KEY,
    subject_name VARCHAR(100) NOT NULL,
    subject_code VARCHAR(10),
    credits INTEGER,
    teacher_id INTEGER NOT NULL,
    FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)
);
```

➢ Table Structure



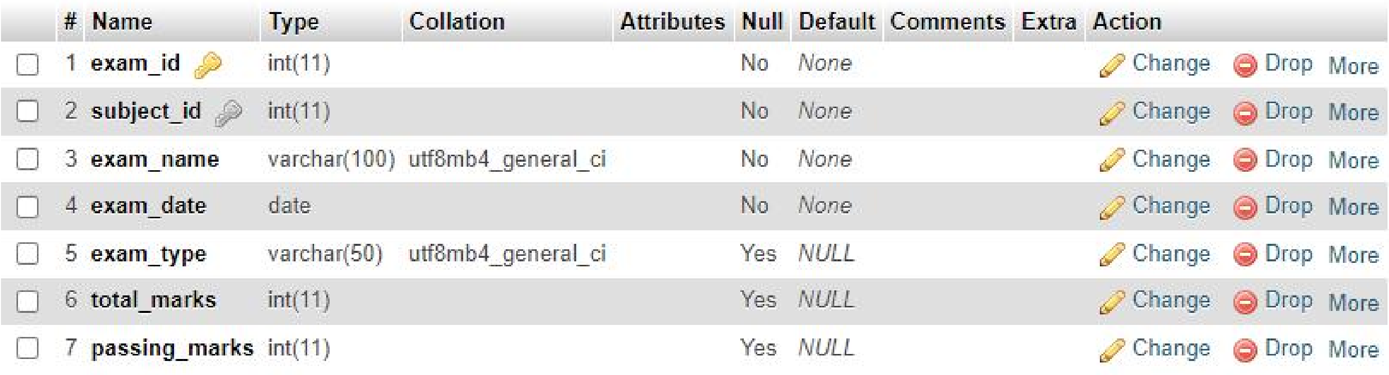
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**subject_id**	int(11)			No	*None*			Change Drop More
□	2	**subject_name**	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
□	3	**subject_code**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	4	**credits**	int(11)			Yes	*NULL*			Change Drop More
□	5	**teacher_id**	int(11)			No	*None*			Change Drop More

Exams Table

Query for creating the Exam table

```
CREATE TABLE exams (
    exam_id INTEGER PRIMARY KEY,
    subject_id INTEGER NOT NULL,
    exam_name VARCHAR(100) NOT NULL,
    exam_date DATE NOT NULL,
    exam_type VARCHAR(50),
    total_marks INTEGER,
    passing_marks INTEGER,
    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id)
);
```

Table Structure



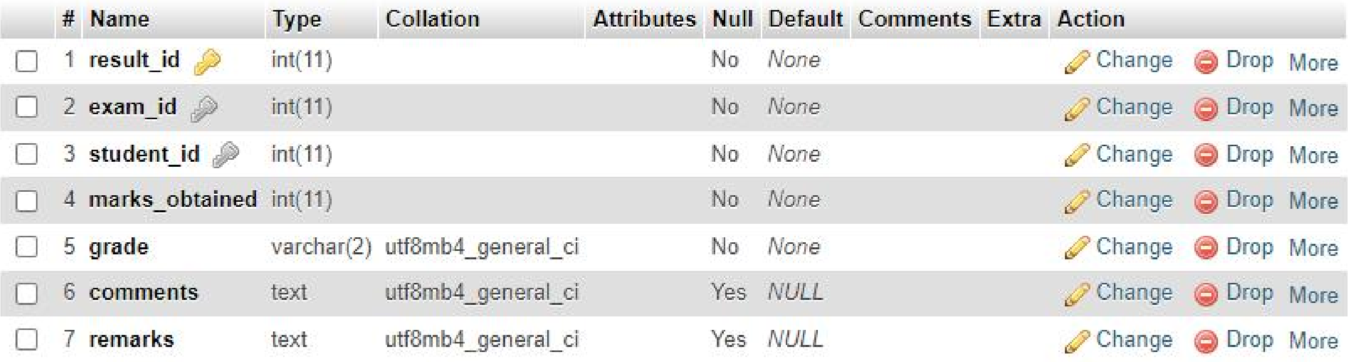
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**exam_id** 🔑	int(11)			No	*None*			Change Drop More
□	2	**subject_id** 🔑	int(11)			No	*None*			Change Drop More
□	3	**exam_name**	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
□	4	**exam_date**	date			No	*None*			Change Drop More
□	5	**exam_type**	varchar(50)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	6	**total_marks**	int(11)			Yes	*NULL*			Change Drop More
□	7	**passing_marks**	int(11)			Yes	*NULL*			Change Drop More

- **Exam_Results Table**

➢ Query for creating the Exam_Result table

```
CREATE TABLE exam_results (
    result_id INTEGER PRIMARY KEY,
    exam_id INTEGER NOT NULL,
    student_id INTEGER NOT NULL,
    marks_obtained INTEGER NOT NULL,
    grade VARCHAR(2) NOT NULL,
    comments TEXT,
    remarks TEXT,
    FOREIGN KEY (exam_id) REFERENCES exams(exam_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);
```

➢ Table Structure



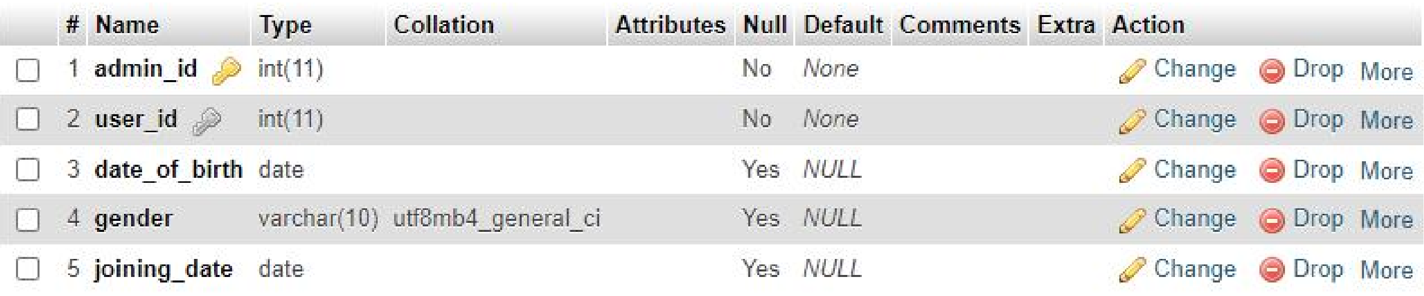
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**result_id**	int(11)			No	*None*			Change Drop More
□	2	**exam_id**	int(11)			No	*None*			Change Drop More
□	3	**student_id**	int(11)			No	*None*			Change Drop More
□	4	**marks_obtained**	int(11)			No	*None*			Change Drop More
□	5	**grade**	varchar(2)	utf8mb4_general_ci		No	*None*			Change Drop More
□	6	**comments**	text	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	7	**remarks**	text	utf8mb4_general_ci		Yes	*NULL*			Change Drop More

- **Administrators Table**

➢ Query for creating the Administrators table

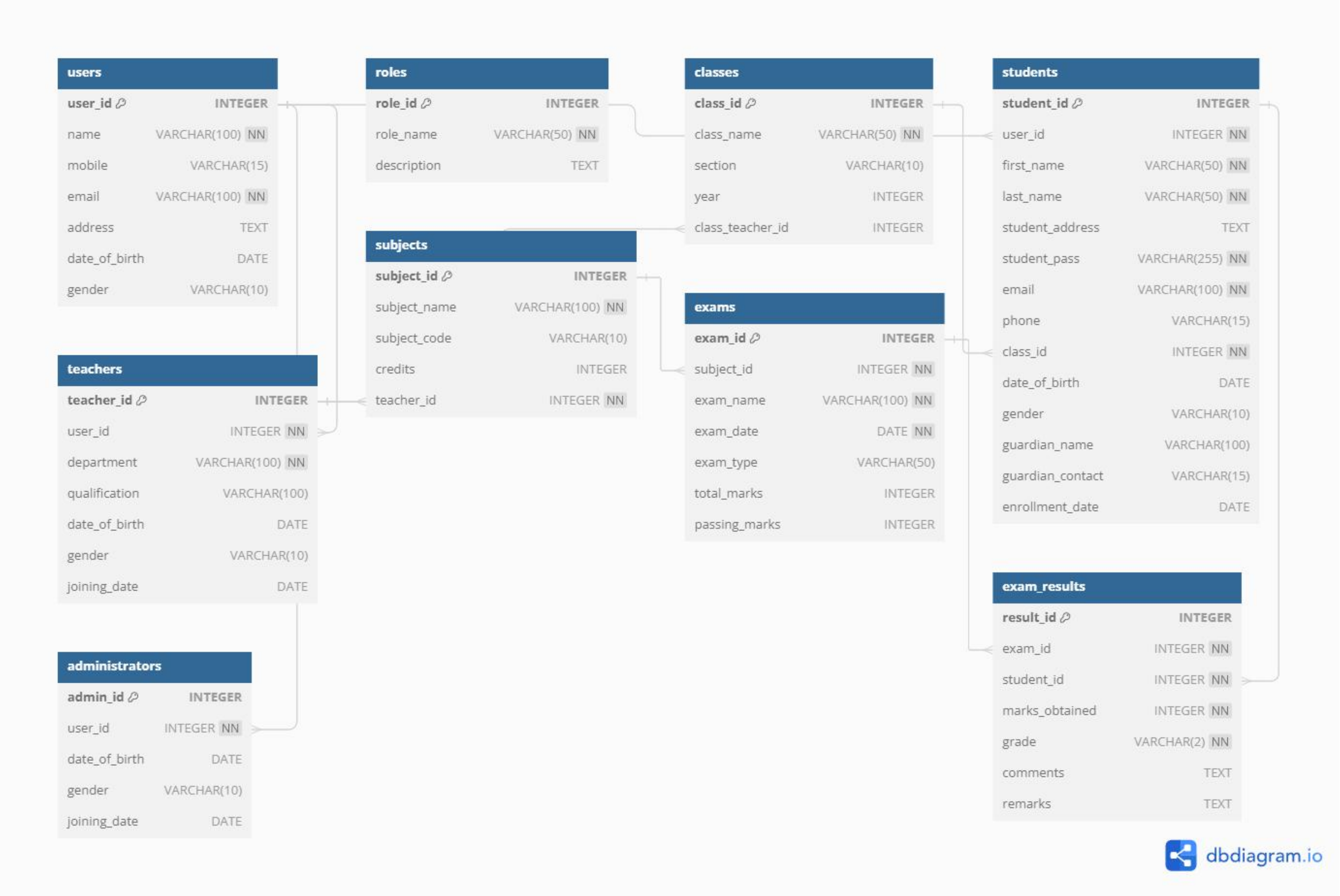
```
CREATE TABLE administrators (
    admin_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    date_of_birth DATE,
    gender VARCHAR(10),
    joining_date DATE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

➢ Table Structure



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**admin_id**	int(11)			No	*None*			Change Drop More
□	2	**user_id**	int(11)			No	*None*			Change Drop More
□	3	**date_of_birth**	date			Yes	*NULL*			Change Drop More
□	4	**gender**	varchar(10)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
□	5	**joining_date**	date			Yes	*NULL*			Change Drop More

➢ Relationship Diagram Between the Tables



users		
user_id	INTEGER	
name	VARCHAR(100)	NN
mobile	VARCHAR(15)	
email	VARCHAR(100)	NN
address	TEXT	
date_of_birth	DATE	
gender	VARCHAR(10)	

roles		
role_id	INTEGER	
role_name	VARCHAR(50)	NN
description	TEXT	

classes		
class_id	INTEGER	
class_name	VARCHAR(50)	NN
section	VARCHAR(10)	
year	INTEGER	
class_teacher_id	INTEGER	

students		
student_id	INTEGER	
user_id	INTEGER	NN
first_name	VARCHAR(50)	NN
last_name	VARCHAR(50)	NN
student_address	TEXT	
student_pass	VARCHAR(255)	NN
email	VARCHAR(100)	NN
phone	VARCHAR(15)	
class_id	INTEGER	NN
date_of_birth	DATE	
gender	VARCHAR(10)	
guardian_name	VARCHAR(100)	
guardian_contact	VARCHAR(15)	
enrollment_date	DATE	

subjects		
subject_id	INTEGER	
subject_name	VARCHAR(100)	NN
subject_code	VARCHAR(10)	
credits	INTEGER	
teacher_id	INTEGER	NN

exams		
exam_id	INTEGER	
subject_id	INTEGER	NN
exam_name	VARCHAR(100)	NN
exam_date	DATE	NN
exam_type	VARCHAR(50)	
total_marks	INTEGER	
passing_marks	INTEGER	

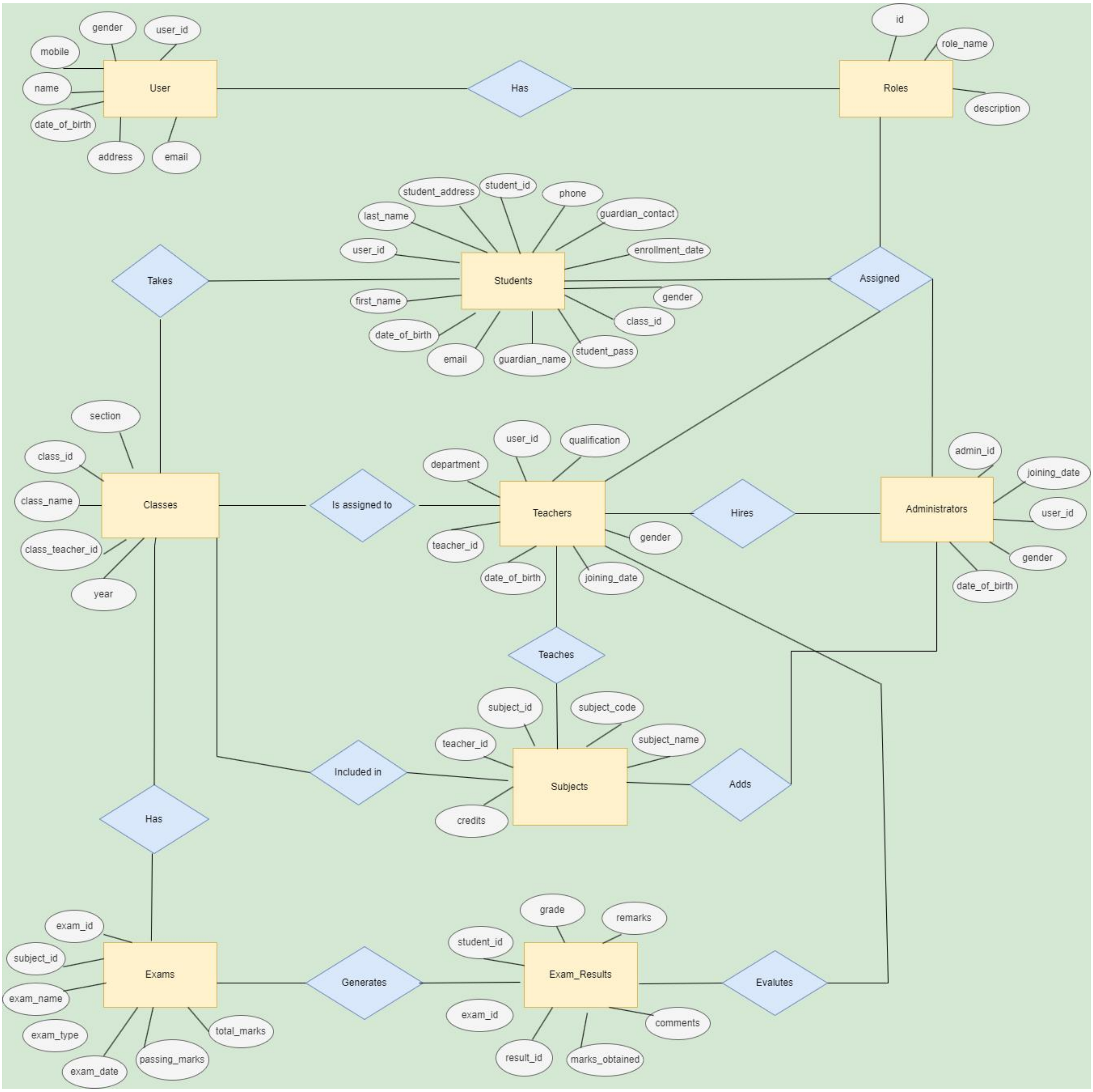
teachers		
teacher_id	INTEGER	
user_id	INTEGER	NN
department	VARCHAR(100)	NN
qualification	VARCHAR(100)	
date_of_birth	DATE	
gender	VARCHAR(10)	
joining_date	DATE	

exam_results		
result_id	INTEGER	
exam_id	INTEGER	NN
student_id	INTEGER	NN
marks_obtained	INTEGER	NN
grade	VARCHAR(2)	NN
comments	TEXT	
remarks	TEXT	

administrators		
admin_id	INTEGER	
user_id	INTEGER	NN
date_of_birth	DATE	
gender	VARCHAR(10)	
joining_date	DATE	

- **ER Diagram**

The ER Diagram for the School Result Management System visually represents the database structure, showcasing the entities, their attributes, and the relationships among them.



ER Diagram Explanation

The Entity-Relationship Diagram (ERD) provides a visual representation of the database schema for the School Result Management System. It illustrates the various entities, their attributes, and the relationships between them. The ERD helps in understanding how data is structured and interconnected, ensuring a well-organized and efficient database design. Here's an explanation of the ERD for the School Result Management System:

- **Entities and Attributes**

Users:

- **Attributes**: user_id, name, mobile, email, address, date_of_birth, gender
- **Description:** Stores general user information applicable to students, teachers, and administrators.

Roles:

- **Attributes:** role_id, role_name, description
- **Description:** Defines different roles within the system, such as student, teacher, and administrator.

Classes:

- **Attributes:** class_id, class_name, section, year, class_teacher_id
- **Description:** Contains information about the various classes in the school, including the class teacher.

Students:

- **Attributes:** student_id, user_id, first_name, last_name, student_address, student_pass, email, phone, class_id, date_of_birth, gender, guardian_name, guardian_contact, enrollment_date

- **Description:** Holds detailed student information and links to the corresponding user and class.

Teachers:

- **Attributes:** teacher_id, user_id, department, qualification, date_of_birth, gender, joining_date

- **Description:** Captures information about teachers, including their qualifications and the department they belong to.

Subjects:

- Attributes: subject_id, subject_name, subject_code, credits, teacher_id
- Description: Stores details about subjects taught in the school, including the assigned teacher.

Exams:

- **Attributes:** exam_id, subject_id, exam_name, exam_date, exam_type, total_marks, passing_marks

- **Description:** Contains information about exams conducted for various subjects.

Exam Results:

- **Attributes:** result_id, exam_id, student_id, marks_obtained, grade, comments, remarks

- **Description:** Records the results of students for different exams, including grades and remarks.

Administrators:

- **Attributes:** admin_id, user_id, date_of_birth, gender, joining_date

- **Description:** Stores administrator-specific information and links to the corresponding user.

Conclusion

- **Comprehensive Solution:** The School Result Management System (SRMS) provides a robust and efficient database solution for managing diverse educational institution needs.

- **Secure and Scalable:** By utilizing a relational database model, the SRMS ensures secure, scalable, and structured data management, facilitating easy access and retrieval of information.

- **User Roles and Permissions:** The system's design includes different user roles and permissions, allowing seamless interaction for administrators, teachers, students, and parents.

- **Essential Functions:** The SRMS supports critical functions such as student enrollment, exam scheduling, result processing, and reporting through detailed tables and well-defined relationships.

- **Engineering Knowledge:** Implementing this solution required in-depth engineering knowledge, particularly in database design and SQL.

- **Wide-Ranging Issues:** The solution addressed various technical, engineering, and user-related issues, necessitating significant abstract thinking and analysis.

- **Adherence to Standards:** The design adhered to database standards and best practices, ensuring reliability and efficiency.

- **Balancing Requirements:** The solution effectively balanced conflicting stakeholder requirements, ensuring a user-friendly experience for all involved.

- **Interdependent Components**: The SRMS involves interdependent components, such as students and classes, exams and results, creating a cohesive and efficient system.

Overall, the SRMS enhances operational efficiency, accuracy, and accessibility of school data, contributing to improved educational outcomes and administrative processes.