

**Mini Project**

|  |  |
| --- | --- |
|  |  |
| **NAME** | **: HARIDHARAN K (122012012855)** |
|  | **VISHNU J (122012012894)** |
|  | **IFFAN A (123012063254)** |
| **CLASS** | **: III – B.Tech CSE – ‘A’** |
| **COURSE CODE** | **: XCS507** |
| **COURSE NAME** | **: DATABASE MANAGEMENT SYSTEM** |
|  | **LABORATORY** |
| **TOPIC** | **: STUDENTS INFORMATION SYSTEM** |
|  | **Submitted To** |
|  | **Ms.S.GOMATHI AP/CSE** |

**Students Information System Documentation**

# 1. Introduction

Student information system means the information system for maintaining and providing student information. Some of them are paper based; heavy manual work is involved in managing and maintaining information such as student personal records. However, recently, most schools maintain the record in database. In this system, the user can add, edit, search, delete and view student’s information. All the records are listed below in the system which makes the user, easy to view the information of each and every student. This system is easy to operate and understand by the user.

# 2. Entities and Attributes

* 1. **Login**

**Attributes:**

* user\_id (INT, Primary Key): Unique identifier for each user.
* username (VARCHAR(50), UNIQUE): Username for logging in.
* password (VARCHAR(255)): Encrypted password.
* user\_role (ENUM): User role or access level (e.g., 'admin' or 'user').

**SQL Definition:**

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY ,

username VARCHAR(50) UNIQUE NOT NULL ,

password VARCHAR(255) NOT NULL ,

user\_role ENUM(‘staff ’, ‘student’)

);

#### **2.2** **Student**

**Attributes:**

* student\_id (INT, Primary Key): Unique identifier for each student.
* name (VARCHAR(100)): Full name of the student.
* dob (DATE): Date of birth of the student.
* gender (ENUM): Gender of the student, options 'Male', 'Female', 'Other'.
* address (VARCHAR(255)): Address of the student.
* phone (VARCHAR(15)): Phone number of the student.
* email (VARCHAR(50)): Email address of the student.

**SQL Definition:**

CREATE TABLE Student (

student\_id INT PRIMARY KEY ,

name VARCHAR(50) ,

dob DATE ,

gender ENUM(‘Male, ‘Female’, ‘Others’) ,

address VARCHAR(255) ,

phone VARCHAR(15) ,

email VARCHAR(50)

);

* 1. **Fees**

**Attributes:**

* fee\_id (INT, Primary Key): Unique identifier for each fee record.
* student\_id (INT, Foreign Key): References Student(student\_id).
* total\_fee (DECIMAL(10, 2)): Total fee amount for Student.
* paid\_amount (DECIMAL(10, 2)): Amount paid by Student.
* due\_date (DATE): Due date for the payment.
* status (ENUM): Fee status (e.g., 'Paid', 'Unpaid').

**SQL Definition:**

CREATE TABLE Fees (

fee\_id INT PRIMARY KEY,

student\_id INT NOT NULL,

total\_fee DECIMAL(10, 2),

paid\_amount DECIMAL(10, 2),

due\_date DATE,

status ENUM('Paid', 'Unpaid') DEFAULT 'Unpaid',

FOREIGN KEY (student\_id) REFERENCES Students(student\_id)

);

* 1. **Marks**

**Attributes:**

* mark\_id (INT, Primary Key): Unique identifier for each mark entry.
* student\_id (INT, Foreign Key): References Students(student\_id).
* tamil\_marks (INT): Tamil marks obtained by the student.
* english\_marks (INT): English marks obtained by the student.
* maths\_marks (INT): Maths marks obtained by the student.
* science\_marks (INT): Science marks obtained by the student.
* social\_marks (INT): Social marks obtained by the student.
* total\_marks (INT): Total marks obtained by the student.
* grade (CHAR(1)): Grade obtained by the student in subject.

**SQL Definition:**

CREATE TABLE Marks (

mark\_id INT PRIMARY KEY,

student\_id INT NOT NULL,

tamil\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

english\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

maths\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

science\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

social\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

total\_marks INT,

grade CHAR(1) ,

FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE

);

# 3. Relationships for Student Information System

#### **Student to Fees**

* **Relationship**: Each student can have multiple fee records, but each fee record is associated with only one student.
* **Foreign Key**: Fees.student\_id references Student.student\_id

**Explanation**: This relationship allows each Student entry to be linked with multiple Fees entries, which record the fees paid, total fees, and due amounts for each student over time.

#### **3.2 Student to Marks**

* **Relationship**: Each student can have multiple marks entries for different subjects, but each marks entry is linked to only one student.
* **Foreign Key**: Marks.student\_id references Student.student\_id

**Explanation**: This relationship connects each Student with multiple Marks entries, allowing the system to record individual subject scores and grades for each student.

#### **3.3 Login for User Authentication**

#### **Relationship**: The Login table is used for managing system access and does not have direct relationships with other tables in this schema.

**Explanation**: This table manages user credentials, allowing access control to the system. user\_role defines the type of access each user has, either as an 'admin' or a 'user'.

**Summary of Relationships :**

**Table Related Table Relational Type Foreign Key**

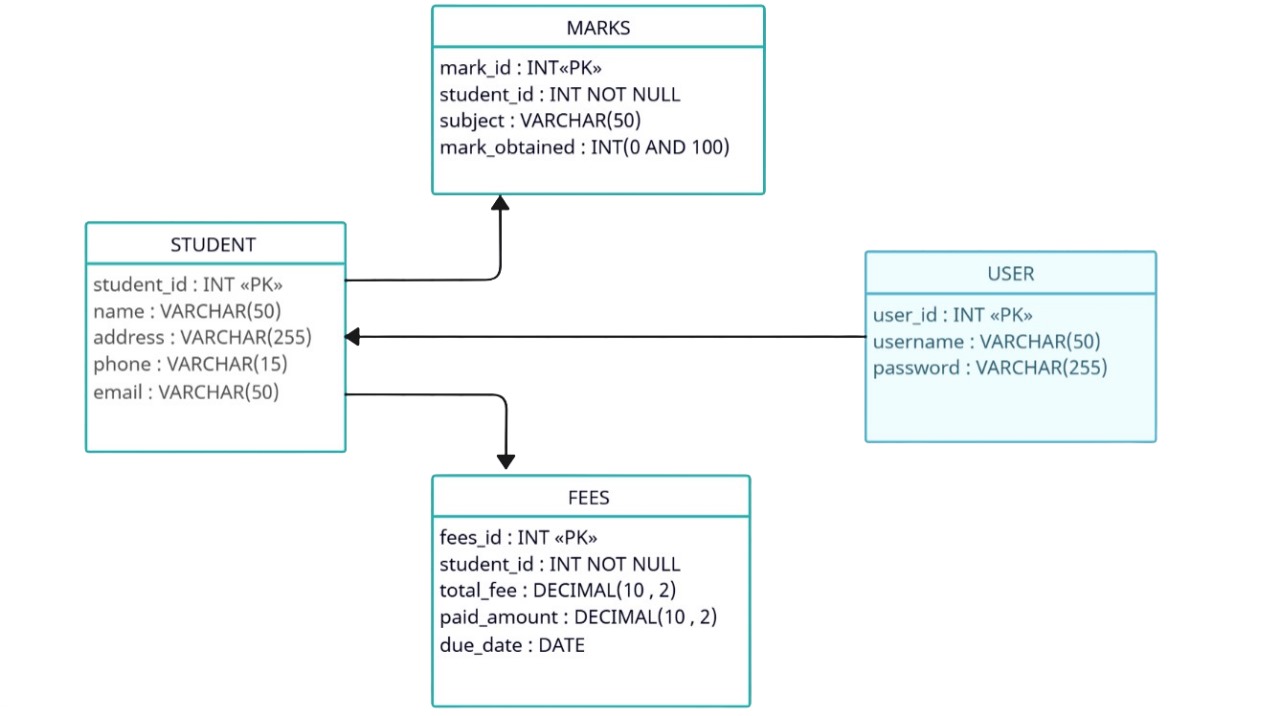
**Student** Fees One-to-Many Fees.student\_id

**Student** Marks One-to-Many marks.student\_id

**Login** -- Standalone --

**4. ER Diagram:**

The ER diagram visually represents the entities and their relationships within the Library Management System.



**Table 1 : Users**

**Query :**

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY ,

username VARCHAR(50) NOT NULL ,

password VARCHAR(255) NOT NULL ,

user\_role ENUM(‘staff ’, ‘student’)

);

INSERT INTO Users ( user\_id, username, password, user\_role )

VALUES

(001, 'john\_doe', 'password123', 'staff'),

(002, 'alice\_smith', 'alicepwd456', 'student'),

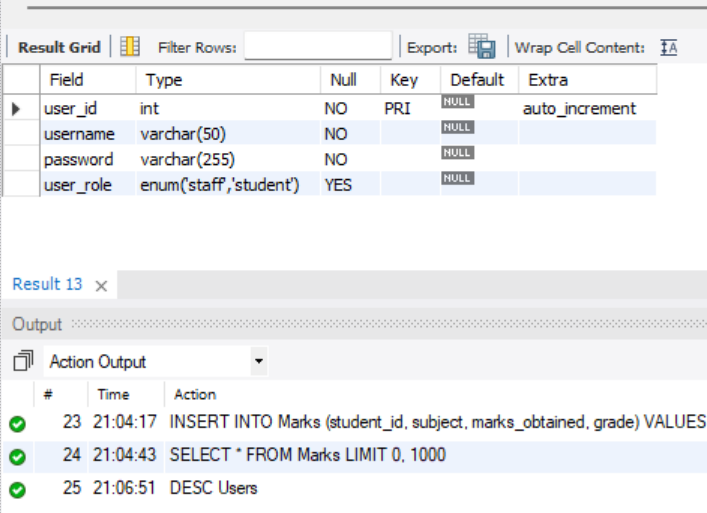
(003, 'michael\_lee', 'mikepass789', 'staff'),

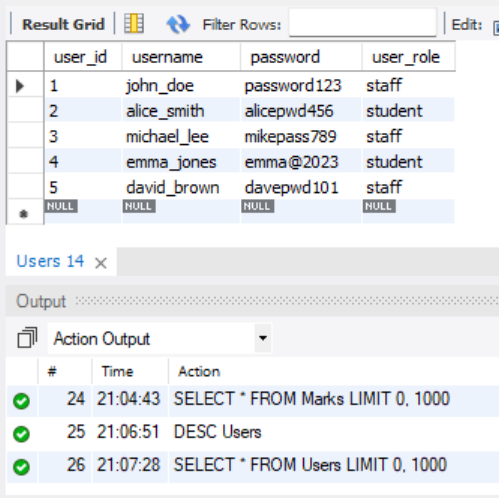
(004, 'emma\_jones', 'emma@2023', 'student'),

(005, 'david\_brown', 'davepwd101', 'staff');

SELECT \* FROM Users;

**Output :**

****

****

**Table 2 : Student**

**Query :**

CREATE TABLE Student (

student\_id INT AUTO\_INCREMENT PRIMARY KEY ,

name VARCHAR(50) ,

dob DATE ,

gender ENUM(‘Male, ‘Female’, ‘Others’) ,

address VARCHAR(255) ,

phone VARCHAR(15) ,

email VARCHAR(50)

);

INSERT INTO Student (name, dob, gender, address, phone, email)

VALUES

('John Doe', '2001-03-15', 'Male', '123 Elm Street, Cityville', '555-1234', 'johndoe@email.com'),

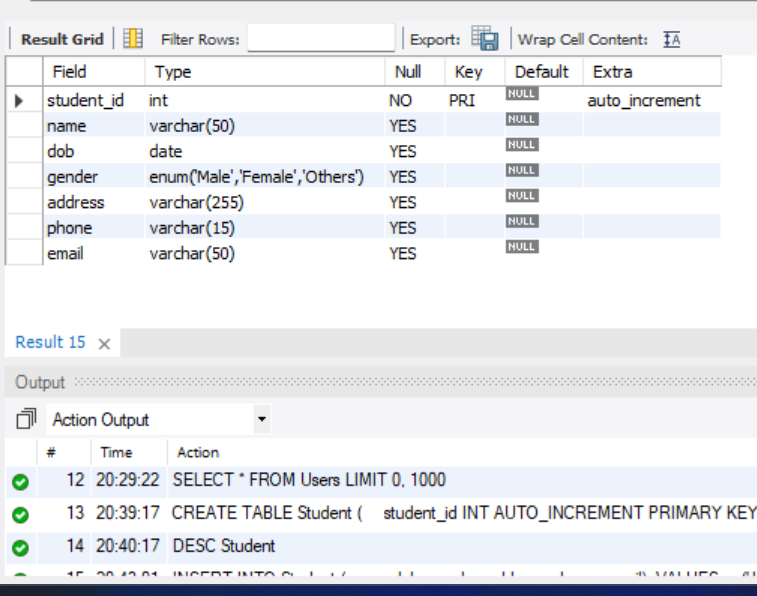
('Jane Smith', '2000-06-22', 'Female', '456 Oak Avenue, Townsville', '555-5678', 'janesmith@email.com'),

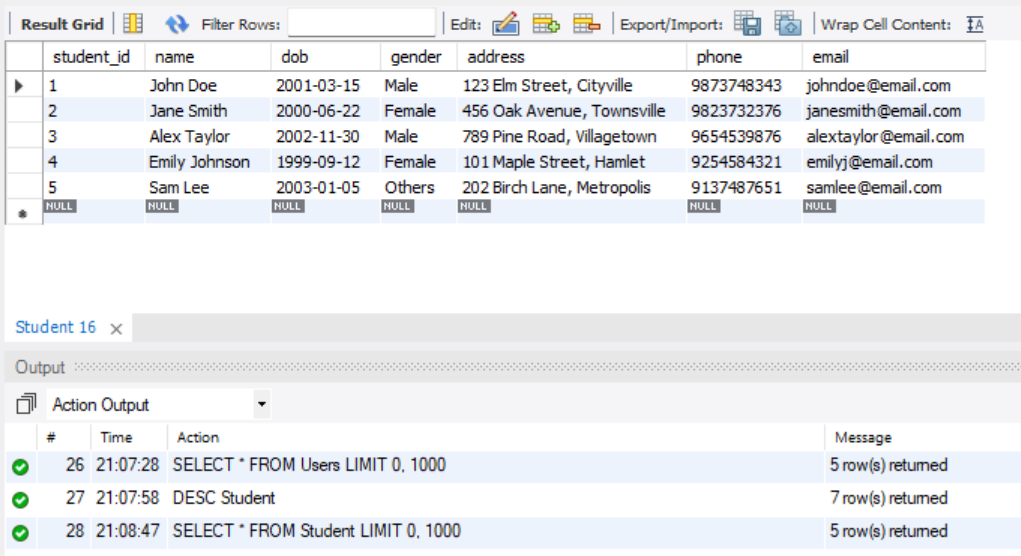
('Alex Taylor', '2002-11-30', 'Male', '789 Pine Road, Villagetown', '555-9876', 'alextaylor@email.com'),

('Emily Johnson', '1999-09-12', 'Female', '101 Maple Street, Hamlet', '555-4321', 'emilyj@email.com'),

('Sam Lee', '2003-01-05', 'Others', '202 Birch Lane, Metropolis', '555-8765', 'samlee@email.com');

**Output :**

****

****

**Table 3 : Fees**

**Query :**

CREATE TABLE Fees (

fee\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT NOT NULL,

total\_fee DECIMAL(10, 2),

paid\_amount DECIMAL(10, 2),

due\_date DATE,

status ENUM('Paid', 'Unpaid') DEFAULT 'Unpaid',

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

);

INSERT INTO Fees (student\_id, total\_fee, paid\_amount, due\_date, status)

VALUES

(1, 1500.00, 500.00, '2024-12-15', 'Unpaid'),

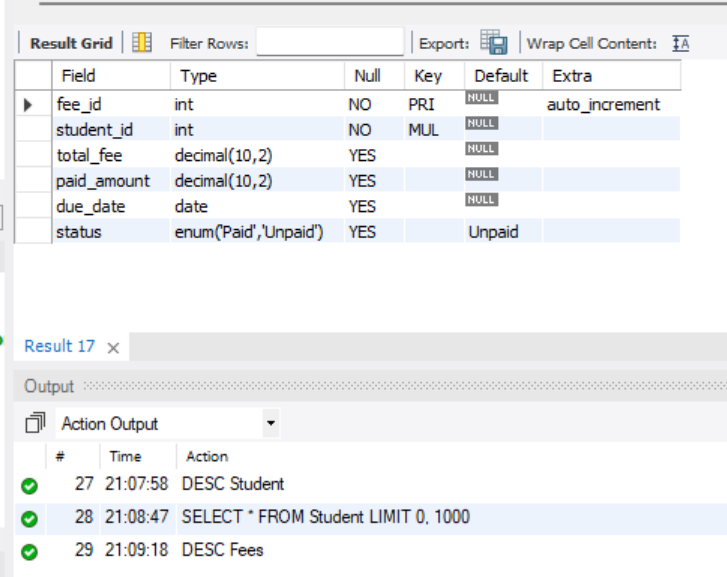
(2, 1200.00, 1200.00, '2024-11-20', 'Paid'),

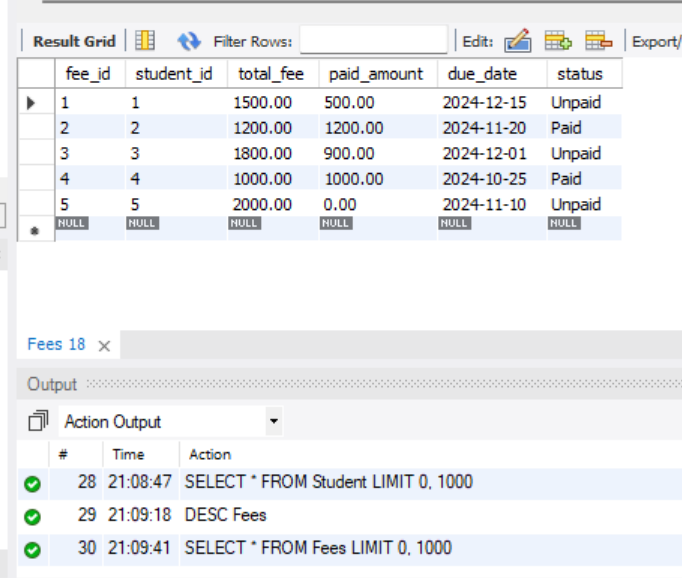
(3, 1800.00, 900.00, '2024-12-01', 'Unpaid'),

(4, 1000.00, 1000.00, '2024-10-25', 'Paid'),

(5, 2000.00, 0.00, '2024-11-10', 'Unpaid');

**Output :**

****

****

**Table 4 : Marks**

**Query :**

CREATE TABLE Marks (

mark\_id INT PRIMARY KEY,

student\_id INT NOT NULL,

tamil\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

english\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

maths\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

science\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

social\_marks INT CHECK (marks\_obtained BETWEEN 0 AND 100),

total\_marks INT,

grade CHAR(1) ,

FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE CASCADE

);

INSERT INTO Marks (mark\_id, student\_id, tamil\_marks, english\_marks, maths\_marks,

science\_marks, social\_marks, total\_marks, grade)

VALUES

(1, 1, 85, 90, 88, 92, 80, 435, 'A'),

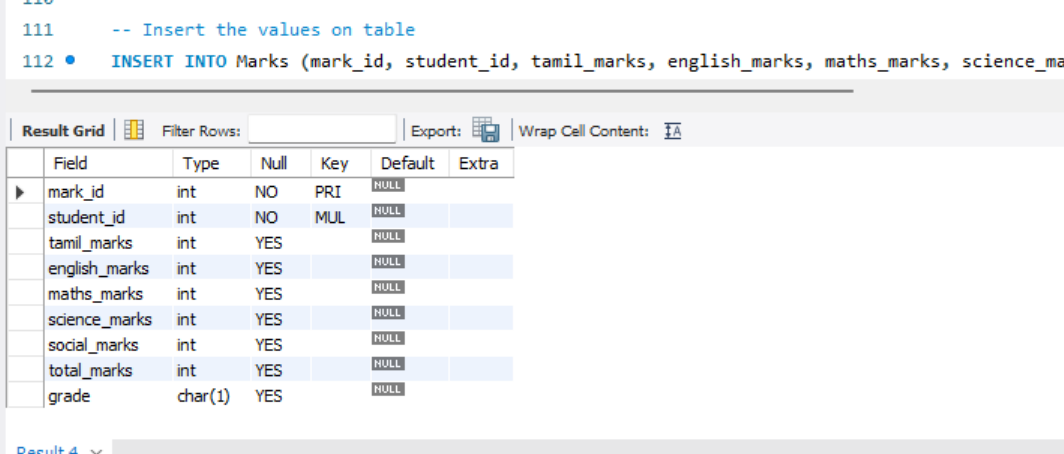
(2, 2, 78, 82, 75, 80, 88, 403, 'B'),

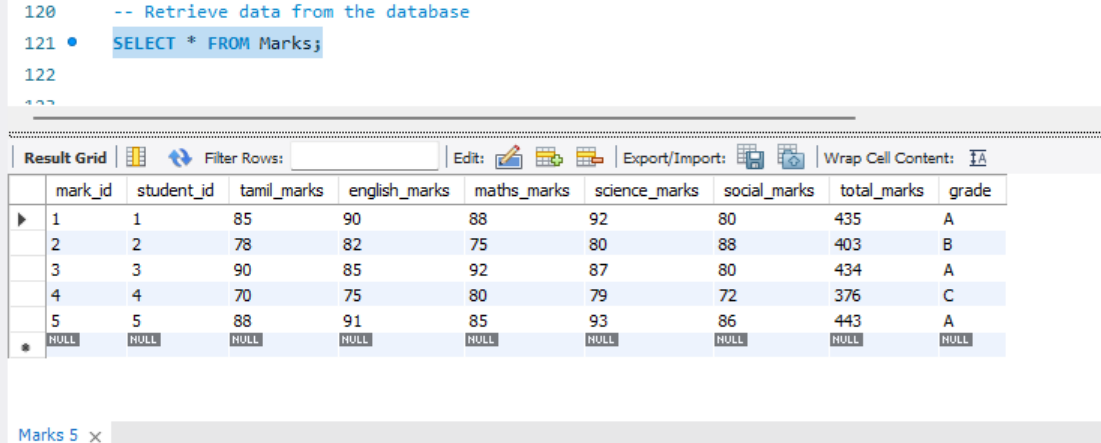
(3, 3, 90, 85, 92, 87, 80, 434, 'A'),

(4, 4, 70, 75, 80, 79, 72, 376, 'C'),

(5, 5, 88, 91, 85, 93, 86, 443, 'A');

**Output :**





Operations based on the current Student Information System design to:

1. **Add a New Student**
2. **Update a Student details**
3. **Remove a Student**
4. **Mark Entry**
5. **View Entire Student Details**
6. **Add a New Student :**

**Query:**

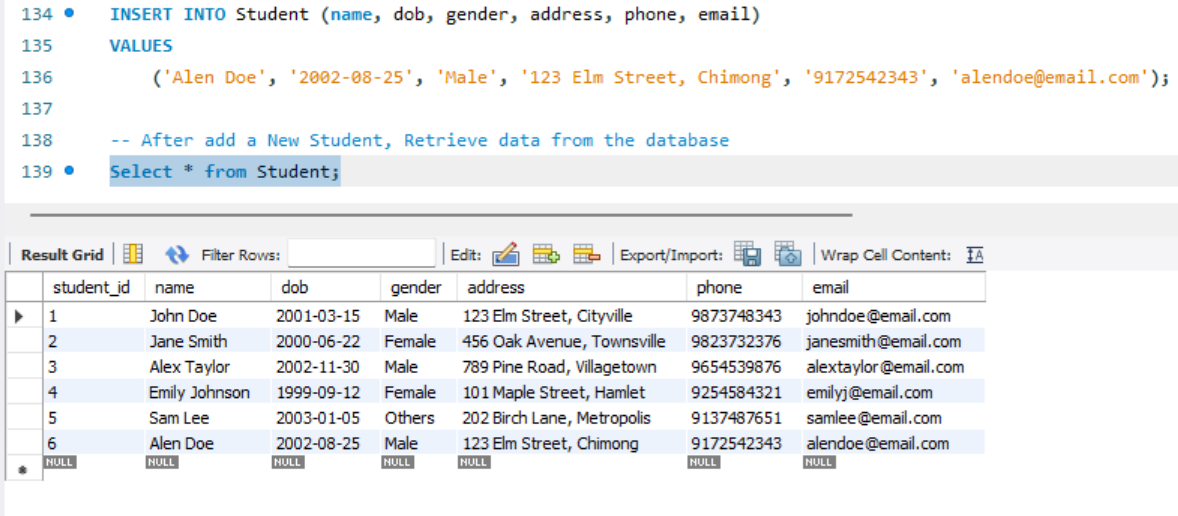
INSERT INTO Student (name, dob, gender, address, phone, email)

VALUES

('Alen Doe', '2002-08-25', 'Male', '123 Elm Street, Chimong', '9172542343', 'alendoe@email.com');

Select \* from Student;

**Output :**



1. **Update a Student details :**

**Query:**

INSERT INTO Fees (student\_id, total\_fee, paid\_amount, due\_date, status)

VALUES

(6, 1500.00, 700.00, '2024-12-28', 'Unpaid');

INSERT INTO Marks (mark\_id, student\_id, tamil\_marks, english\_marks, maths\_marks,

science\_marks, social\_marks, total\_marks, grade)

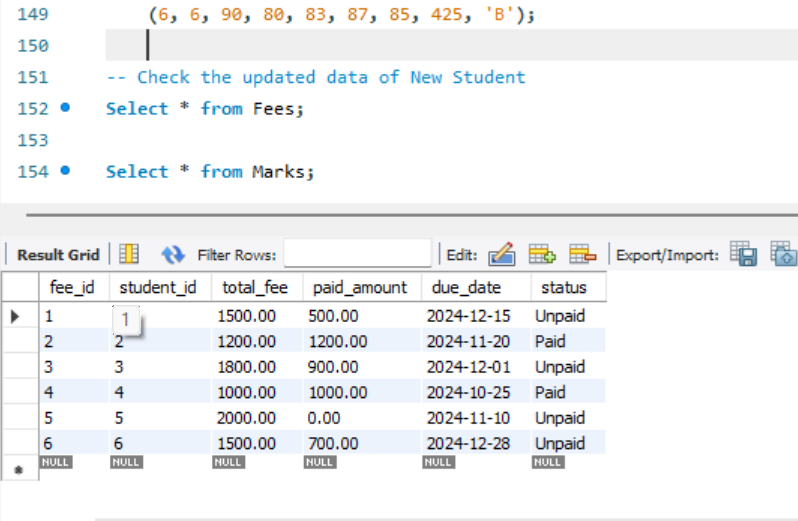
VALUES

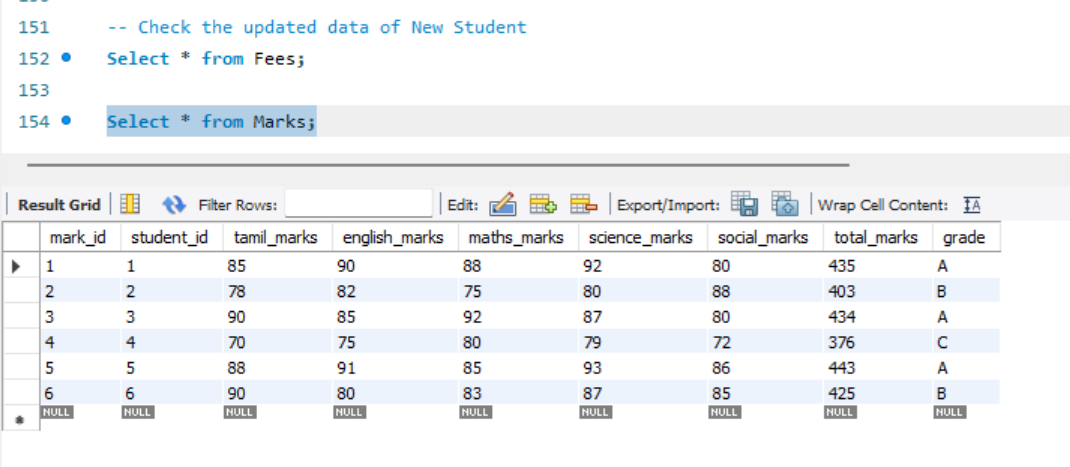
(6, 6, 90, 80, 83, 87, 85, 425, 'B');

Select \* from Fees;

Select \* from Marks;

**Output:**





1. **Remove a Student :**

**Query:**

DELETE FROM Student WHERE student\_id = 2;

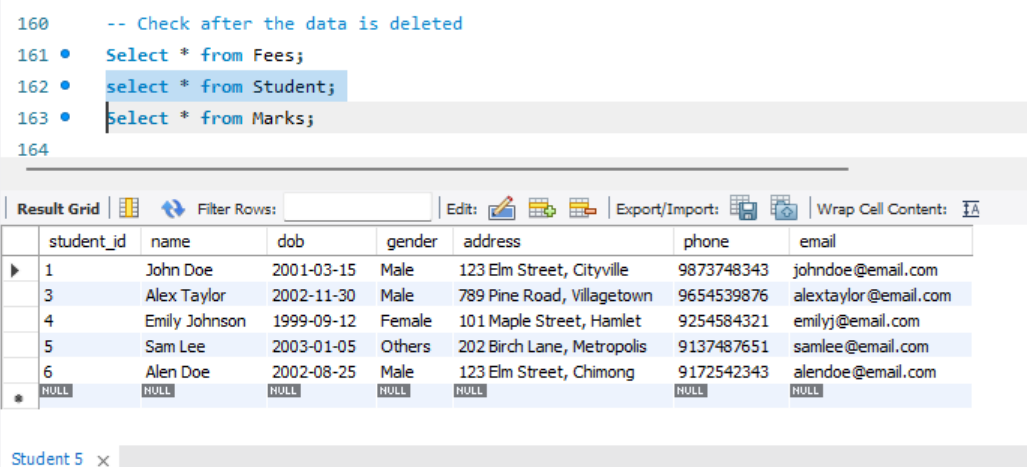
DELETE FROM Fees WHERE student\_id = 2;

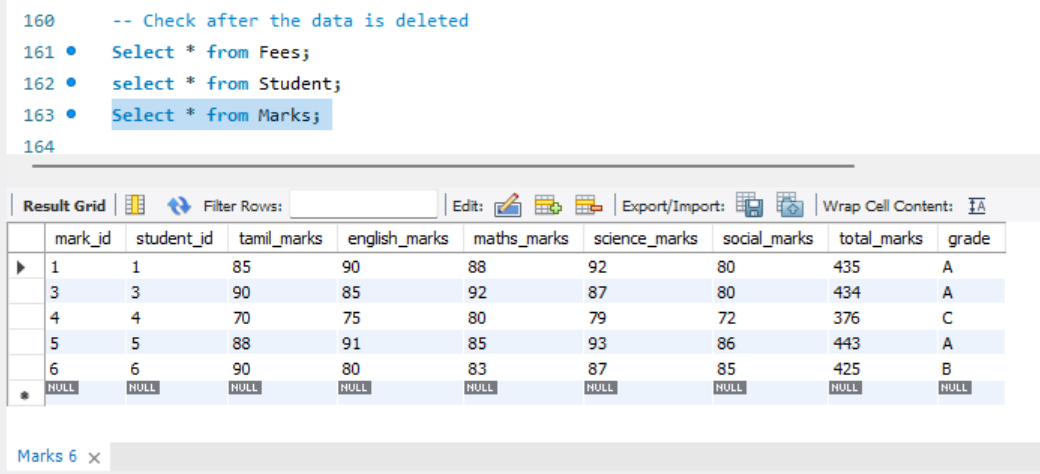
Select \* from Fees;

select \* from Student;

Select \* from Marks;

**Output:**





**4. Mark Entry for New Student :**

**Query:**

INSERT INTO Marks (mark\_id, student\_id, tamil\_marks, english\_marks, maths\_marks,

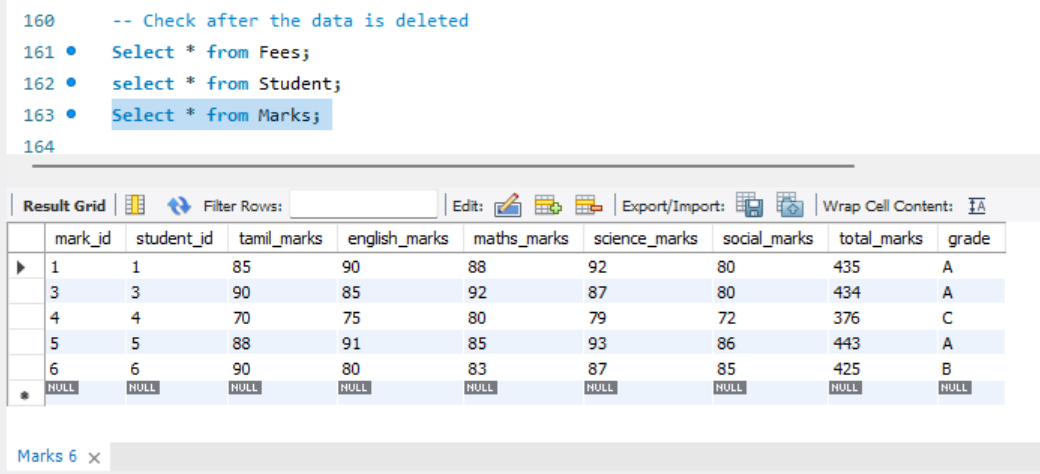
science\_marks, social\_marks, total\_marks, grade)

VALUES

(6, 6, 90, 80, 83, 87, 85, 425, 'B');

Select \* from Marks;

**Output:**



**5. View Entire Student Details**

**Query :**

SELECT

s.student\_id,

s.name,

s.dob,

s.gender,

s.address,

s.phone,

s.email

f.fee\_id,

f.total\_fee,

f.paid\_amount,

f.due\_date,

f.status,

m.mark\_id,

m.tamil\_marks,

m.english\_marks,

m.maths\_marks,

m.science\_marks,

m.social\_marks,

m.total\_marks,

m.grade

FROM

Student s

JOIN

Marks m ON s.student\_id = m.student\_id

JOIN

Fees f ON s.student\_id = f.student\_id

WHERE

s.student\_id = 4;

**Output :**

