

STUDENT MANAGEMENT SYSTEM

INTRODUCTION

The Student Management System is a database-driven application designed to efficiently manage student-related data, course information, and enrollment details. The system provides a centralized platform for educational institutions to store, retrieve, and analyze student and course data with ease. By leveraging relational database technology, the system ensures data consistency, integrity, and accessibility.

OBJECTIVE

The primary objective of this system is to:

1. Maintain detailed records of students, courses, and enrollments.
2. Facilitate the management of academic information in a structured manner.
3. Provide tools for querying and analyzing data to enhance decision-making processes.
4. Ensure data security and integrity through proper database constraints and relationships.

KEY FEATURES

1. **Student Records Management:** Stores personal details like name, date of birth, email, and phone number.
2. **Course Management:** Maintains course details such as course name, description, and credits.
3. **Enrollment Tracking:** Links students with courses and records their grades.
4. **Data Retrieval:** Supports SQL-based queries for fetching and analyzing data.
5. **Relational Integrity:** Ensures proper relationships between entities using foreign keys.

6. **Scalability:** Can be expanded to include additional features such as attendance tracking and fee management.

DATABASE SCHEMA

1. Students Table:

- StudentID: Primary key, unique identifier for each student.
- FirstName: First name of the student.
- LastName: Last name of the student.
- DOB: Date of birth of the student.
- Email: Email address of the student.
- PhoneNumber: Contact number of the student.

2. Courses Table:

- CourseID: Primary key, unique identifier for each course.
- CourseName: Name of the course.
- CourseDescription: Brief description of the course.
- Credits: Number of credits assigned to the course.

3. Enrollments Table:

- EnrollmentID: Primary key, unique identifier for each enrollment.
- StudentID: Foreign key referencing Students.StudentID.
- CourseID: Foreign key referencing Courses.CourseID.
- Grade: Grade achieved by the student in the course.

LIMITATIONS

1. Lack of user interface for non-technical users to interact with the database.

2. Limited to academic records; does not include features like attendance or financial data.
3. Manual SQL query execution required for data retrieval and analysis.
4. No automated notification or alert system for updates or deadlines.

FUTURE IMPROVEMENTS

1. Development of a user-friendly web or mobile application interface.
2. Integration of attendance and fee management modules.
3. Implementation of data analytics dashboards for performance tracking.
4. Inclusion of automated email and SMS notifications.
5. Support for multiple languages and localization.

SYSTEM STUDY

The system is based on a relational database model, ensuring that data is organized in interconnected tables. It employs SQL for data manipulation and retrieval, providing a robust and scalable solution. The study reveals that most academic institutions face challenges in managing student data due to scattered and unstructured storage. This system addresses these issues by consolidating data into a single platform.

SYSTEM DESIGN

1. **Entity-Relationship (ER) Diagram:** Represents the relationships between Students, Courses, and Enrollments.
2. **Normalization:** Ensures data is stored in a structured manner without redundancy.
3. **Database Constraints:** Includes primary keys, foreign keys, and data type restrictions to maintain integrity.
4. **Logical Flow:**

- Input: Data entry into Students and Courses tables.
- Processing: Enrollments linking students to courses.
- Output: Query results for analysis and reporting.

MODULES

1. Student Module:

- Add, update, and delete student records.
- View student details.

2. Course Module:

- Add, update, and delete course information.
- View course details.

3. Enrollment Module:

- Register students for courses.
- Assign and update grades.

4. Reporting Module:

- Generate reports on student performance.
- Count and list students enrolled in each course.

CONCLUSION

The Student Management System provides an efficient and reliable method for managing academic records. By leveraging relational databases, it ensures data integrity and accessibility. While the current system meets fundamental requirements, future enhancements will make it more comprehensive and user-friendly. This system is a stepping stone toward fully digitalized academic management.

CODING

SQL*Plus: Release 21.0.0.0.0 - Production on Wed Jan 15 19:06:47 2025

Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: system

Enter password:

Last Successful login time: Wed Jan 15 2025 19:00:14 +05:30

Connected to:

Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

#CREATING TABLES

(1)STUDENTS:

```
SQL> create table Students(  
  2 StudentID int primary key,  
  3 FirstName varchar(50),  
  4 LastName varchar(50),  
  5 DOB date,  
  6 Email varchar(100),  
  7 PhoneNumber integer);
```

Table created.

(2)COURSES:

```
SQL> create table Courses(
```

- 2 CourseID int primary key,
- 3 CourseName varchar(100),
- 4 CourseDescription varchar(50),
- 5 Credits integer);

Table created.

(3)ENROLLMENTS:

SQL> create table Enrollments(

- 2 EnrollmentID int primary key,
- 3 StudentID int,
- 4 CourseID int,
- 5 Grade char(20),
- 6 foreign key (StudentID) references Students (StudentID),
- 7 foreign key (CourseID) references Courses (CourseID));

Table created.

#DESCRIBING TABLES

(1)STUDENTS:

SQL> desc Students;

Name	Null?	Type

STUDENTID	NOT NULL	NUMBER(38)
FIRSTNAME		VARCHAR2(50)
LASTNAME		VARCHAR2(50)
DOB		DATE
EMAIL		VARCHAR2(100)

PHONENUMBER NUMBER(38)

(2)COURSES:

SQL> desc Courses;

Name	Null?	Type

COURSEID	NOT NULL	NUMBER(38)
COURSENAME		VARCHAR2(100)
COURSEDESCRIPTION		VARCHAR2(50)
CREDITS		NUMBER(38)

(3)ENROLLMENTS:

SQL> desc Enrollments;

Name	Null?	Type

ENROLLMENTID	NOT NULL	NUMBER(38)
STUDENTID		NUMBER(38)
COURSEID		NUMBER(38)
GRADE		CHAR(20)

#INSERTING VALUES

(1)STUDENTS:

SQL> insert into Students values(001,'John','Doe',to_date('2000-01-01','yy-mm-dd'),'john@gmail.com',9674376345);

1 row created.

```
SQL> insert into Students values(002,'Jane','smith',to_date('1999-05-15','yy-mm-dd'),'smith@gmail.com',9865443223);
```

1 row created.

(2)COURSES:

```
SQL> insert into Courses values(1,'Database Management','Learn about relational databases',4);
```

1 row created.

```
SQL> insert into Courses values(2,'Web Development','Introduction to front-end and back-end',3);
```

1 row created.

(3)ENROLLMENTS:

```
SQL> insert into Enrollments values(11,001,1,'A');
```

1 row created.

```
SQL> insert into Enrollments values(22,002,2,'B');
```

1 row created.

#QUERYING DATA

(1)FETCH ALL STUDENTS AND THEIR COURSES:


```
SQL> select Students.FirstName,Students.LastName,Courses.CourseName,Enrollments.Grade from  
Enrollments join
```

```
2 Students on Enrollments.StudentID=Students.StudentID join
```

```
3 Courses on Enrollments.CourseID=Courses.CourseID;
```

FIRSTNAME

LASTNAME

COURSENAME

GRADE

John

Doe

Database Management

A

FIRSTNAME

LASTNAME

COURSENAME

GRADE

Jane

smith

Web Development

B

(2)COUNT THE NUMBER OF STUDENTS IN EACH COURSE:

```
SQL> select Courses.CourseName,count(Enrollments.StudentID)as TotalStudents from Enrollments
2 join Courses on Enrollments.CourseID=Courses.CourseID group by Courses.CourseName;
```

COURSENAME

TOTALSTUDENTS

Database Management

1

Web Development

1

(3)GET STUDENT DETAILS WHO SCORED 'A' IN ANY COURSE:

```
SQL> select Students.FirstName,Students.LastName,Courses.CourseName from Enrollments
2 join Students on Enrollments.StudentID=Students.StudentID
3 join Courses on Enrollments.CourseID=Courses.CourseID
4 where Enrollments.Grade='A';
```

FIRSTNAME

LASTNAME

COURSENAME

John

Doe

Database Management

OUTPUT SCREENSHOT

```
SQL Plus
SQL*Plus: Release 21.0.0.0.0 - Production on Wed Jan 15 19:06:47 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Wed Jan 15 2025 19:00:14 +05:30

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> create table Students(
  2  StudentID int primary key,
  3  FirstName varchar(50),
  4  LastName varchar(50),
  5  DOB date,
  6  Email varchar(100),
  7  PhoneNumber integer);

Table created.

SQL> create table Courses(
  2  CourseID int primary key,
  3  CourseName varchar(100),
  4  CourseDescription varchar(50),
  5  Credits integer);

Table created.

SQL> create table Enrollments(
  2  EnrollmentID int primary key,
  3  StudentID int,
  4  CourseID int,
  5  Grade char(20),
  6  foreign key (StudentID) references Students (StudentID),
  7  foreign key (CourseID) references Courses (CourseID));

Table created.
```

```
SQL> insert into Students values(001,'John','Doe',to_date('2000-01-01','yy-mm-dd'),'john@gmail.com',9674376345);
1 row created.

SQL> insert into Students values(002,'Jane','smith',to_date('1999-05-15','yy-mm-dd'),'smith@gmail.com',9865443223);
1 row created.
```

```
SQL> insert into Courses values(1,'Database Management','Learn about relational databases',4);  
1 row created.  
SQL> insert into Courses values(2,'Web Development','Introduction to front-end and back-end',3);  
1 row created.
```

```
SQL> insert into Enrollments values(11,001,1,'A');  
1 row created.  
SQL> insert into Enrollments values(22,002,2,'B');  
1 row created.
```

```
SQL> desc Students;
```

Name	Null?	Type
STUDENTID	NOT NULL	NUMBER(38)
FIRSTNAME		VARCHAR2(50)
LASTNAME		VARCHAR2(50)
DOB		DATE
EMAIL		VARCHAR2(100)
PHONENUMBER		NUMBER(38)

```
SQL> desc Courses;
```

Name	Null?	Type
COURSEID	NOT NULL	NUMBER(38)
COURSENAME		VARCHAR2(100)
COURSEDESCRIPTION		VARCHAR2(50)
CREDITS		NUMBER(38)

```
SQL> desc Enrollments;
```

Name	Null?	Type
ENROLLMENTID	NOT NULL	NUMBER(38)
STUDENTID		NUMBER(38)
COURSEID		NUMBER(38)
GRADE		CHAR(20)

```
SQL> select Students.FirstName,Students.LastName,Courses.CourseName,Enrollments.Grade from Enrollments join
2 Students on Enrollments.StudentID=Students.StudentID join
3 Courses on Enrollments.CourseID=Courses.CourseID;
```

```
FIRSTNAME
-----
LASTNAME
-----
COURSENAME
-----
GRADE
-----
John
Doe
Database Management
A
```

```
FIRSTNAME
-----
LASTNAME
-----
COURSENAME
-----
GRADE
-----
Jane
smith
Web Development
B
```

```
SQL> select Courses.CourseName,count(Enrollments.StudentID)as TotalStudents from Enrollments
2 join Courses on Enrollments.CourseID=Courses.CourseID group by Courses.CourseName;
```

```
COURSENAME
-----
TOTALSTUDENTS
-----
Database Management
1
Web Development
1
```

```
SQL> select Students.FirstName,Students.LastName,Courses.CourseName from Enrollments
2 join Students on Enrollments.StudentID=Students.StudentID
3 join Courses on Enrollments.CourseID=Courses.CourseID
4 where Enrollments.Grade='A';
```

```
FIRSTNAME
-----
LASTNAME
-----
COURSENAME
-----
John
Doe
Database Management
```
