College Data Management System

Database Management Systems

MSCS_542L_256_23S

Super Six

Marist College

School of Computer Science and Mathematics

Submitted To: Dr. Reza Sadeghi

Spring 2023

Project Report of College data management system

Team Name

Super Six

Team members

Pradeep Reddy Baireddy.	pradeepreddy.baireddy1@marist.edu (Team Head)
Prajakta kshirsagar	prajaktakshirsagar2022@gmail.com (Team member)
Deepthi Niharika Gadepalli	DeepthiNiharika.Gadepalli1@marist.edu (Team member)
Bharadwaja Thota	bharadwaja.thota1@marist.edu (Team member)
Pradeep Reddy Macha	pradeepreddy.macha1@marist.edu (Team member)
Chakradhar Reddy Marepally	ChakradharReddy.Marepally1@marist.edu (Team member)

Table of Contents

Table of Figures

Project Objective:

The primary objective of the college data management system is to efficiently manage the data that includes a huge amount of data that related to student, courses, faculty, department, admissions and other vital information that need to be stored and retrieved. At this stage, this phase is presenting the conceptual design of college data management system through an Entity-Relationship (ER) model and an Enhanced ER (EER) model.

Review the Related Work:

[1] The report provides a comprehensive explanation of the system's many different components, such as the database structure, the user interface, and the data processing algorithms.

The SDMS is intended to handle student data for a university or college. Accessing, updating, and retrieving student records, including personal information, scholastic background, and enrolment status, is made possible for administrators and staff members by this software. MySQL is used as the database management system, and Java is used as the programming language, in the construction of the system.

Authentication of users, data protection, and data backup are some of the topics covered in this report, which explains the various requirements and specifications for the SDMS. In addition to that, it provides an explanation of the design of the database schema, which includes tables for the information of students, specifics of courses, registration records, and other pertinent data. The user interface of the system is developed using Java Swing, and it gives users the ability to perform a variety of duties, including adding, editing, and deleting student entries.

The report generation capabilities of the SDMS are among the system's most important characteristics. The system has the capability to produce a variety of records, such as summaries of registration, student profiles, and course schedules. The functionality for the generation of reports is created with Java Reporting Tools (Jasper Reports), and it has the capability to generate reports in a variety of formats, including PDF, HTML, and CSV.

Overall, the Student Data Management System (SDMS) that is explained in the report is an all-encompassing system that is capable of managing student data in an efficient manner for an institution or college. The ability to authenticate users, protect data, and generate reports are just some of the features that make this app a helpful tool for both administrators and staff members. The report gives an in-depth look into the system's design and implementation, which makes it an invaluable resource for anyone who is interested in constructing a system that is comparable to the one described.

[2] This document serves as a project summary for a Student Information Management System (SIMS) that was built with PHP and MySQL. The report provides an overview of the system's architecture as well as its implementation, including details on its features, functionalities, and user interface.

The Student Information Management System (SIMS) is a database that can handle student information for a school of higher education. Accessing, updating, and retrieving student records, including personal information, scholastic background, and enrolment status, is made possible for administrators and staff members by this software. PHP is used as the programming language, and MySQL is used as the database administration system, in the construction of the system.

The report includes an in-depth discussion of the SIMS's many individual components, such as the database structure, user interface design, and data processing algorithms. Tables containing information about students, specifics about courses, registration records, and any other pertinent data are included in the database structure. It is possible for users to perform a variety of duties, including adding, editing, and deleting student information, thanks to the user interface's user-friendly and straightforward design.

The capability to generate reports is regarded as one of the most important aspects of the SIMS. The system is capable of generating a variety of reports, including student names, registration statistics, and course schedules, among others. The functionality for the generation of reports is created with the FPDF library, and it has the capability to generate reports in a variety of formats, including PDF and CSV.

The article also addresses the safety precautions that were taken with the SIMS, including the authentication of users and the encrypting of data. The system protects sensitive data and requires users to authenticate themselves by using a username and password. This is done to prevent unauthorized access to sensitive information.

The Student Information Management System (SIMS) that is described in the report is, in general, a comprehensive system that is able to handle student data for a university or college successfully. The ability to authenticate users, protect data, and generate reports are just some of the features that make this app a helpful tool for both administrators and staff members. The report offers a comprehensive analysis of the system's architecture as well as its implementation, which makes it an invaluable resource for anyone considering the construction of a system comparable to the one described in the report using PHP and MySQL.

[3] A research paper with the topic "Development of a Student Information System using Open-Source Technologies" has been supplied for your review. In this article, we describe the design and implementation of a student information system (SIS) that makes use of open-source technologies such as PHP, MySQL, and Apache.

The Student Information System (SIS) at an institution or college is built to organize and handle student data. Accessing, updating, and retrieving student records, such as personal information, scholastic background, and enrolment status, is made possible for administrators and faculty members by the software. PHP is used as the programming language for the system's construction, while MySQL serves as the database management system, and Apache is used as the web server.

In this study, a comprehensive analysis of the various aspects of the SIS, including its database schema, user interface design, and data processing algorithms, is presented. Tables containing information about students, specifics about courses, registration records, and any other pertinent data are included in the database structure. It is possible for users to perform a variety of duties, including adding, editing, and deleting student information, thanks to the user interface's user-friendly and straightforward design.

The ability to generate reports is regarded as one of the most important aspects of the SIS. The system is capable of generating a variety of reports, including student names, registration statistics, and course schedules, among others. The functionality for the generation of reports is created with the help of the TCPDF library, and it is able to output reports in a variety of formats, including PDF and CSV.

The document also addresses the security mechanisms that were implemented in the SIS, such as data encryption and user authentication, among other things. The system protects sensitive data and requires users to authenticate themselves by using a username and password. This is done to prevent unauthorized access to sensitive information.

In general, the Student Information System (SIS) that is explained in the paper is an all-encompassing system that is capable of managing student data in an efficient manner for an institution or college. The ability to authenticate users, protect data, and generate reports are just a few of the features that make this app a helpful tool for both administrators and academic members. Because it was built with open-source software, the system is not only low-cost but also easily approachable to a diverse variety of organizations. This makes the paper an invaluable resource for anyone who is interested in creating a system comparable to the one described in the paper using open-source software because it offers a comprehensive insight into the system's design and implementation.

The Merits of the Project:

- Data privacy and security in college data management systems able to provide security for sensitive data of students and faculty such as personal and financial information. This will ensure students and faculty that their sensitive data is secure.
- Increased efficiency in the college data management system helps to save time in data storing and retrieving and helps to automate some tasks like attendance.
- Data analysis in college data management system will help the institutions to analyze the previous data that has been stored and using it for better outcomes in future.
- An efficient accessibility in college data management system will allow the students, faculty and administration staff to access the data swiftly such as academic attendance, grades, academic records and schedules.
- Data Validation Integrate data validation into the system so that you can be certain that all of the data that is inputted into the system is correct and comprehensive. This can be accomplished by incorporating form validation into user input fields and checking to see that all mandatory fields are complete before allowing the user to send in the form.
- A search and filter functionality should be added to the system so that users can quickly locate the data they need regarding a particular student. This can be accomplished by including a search box or filter choices on the sites of the system that are pertinent to the question.
- Strength and Complexity of Passwords to make sure that user identities are kept safe, the system should have guidelines for both the strength and the complexity of passwords. Implementing password policies that require users to generate robust passwords and that compel users to update their passwords on a frequent basis is one way to accomplish this goal.

Entity Relationship Model (ER Model):

Entities and their attributes:

Entity	Attributes
Student	Student_ID, FirstName, MiddleName, LastName, DOB, Gender, Age, Hostel_ID, Program_type
Faculty	Fac_ID, Fac_name, Designation, Office_hours, Dept_ID
Courses	Course_ID, Course_Name, Duration, Course_desc, Credits, Program_type
Department	Dept_ID, Dept_Name, Programs_offered, HOD, Office_Location
Admissions	Student_ID, Admission_Number, Application_date, Decision_Date, Test_written, Test_Score, Enrollment_Status
Programs	Program_ID, Program_Name, credits_required, Duration, Number_of_courses, Tution_Fee, Dept_ID
Grades_and_Attendance	Student_ID, Session_ID, Assignment_Score, Lab_score, Attendence, Quiz_Score, Student_grades, Class_ID
Student_Info	Student_ID, Mail, Phone, Father_Name, Mother_name, Address (Door, Street, City, State, Zip)
Sessions	Session_ID, Class_Location, Course_ID, Faculty_ID, Timings, Duration
Hostel	Hostel_ID, Hostel_Name, Amenities, Room_Number, Vacancy, Price_Range

Entity Relationship Diagram

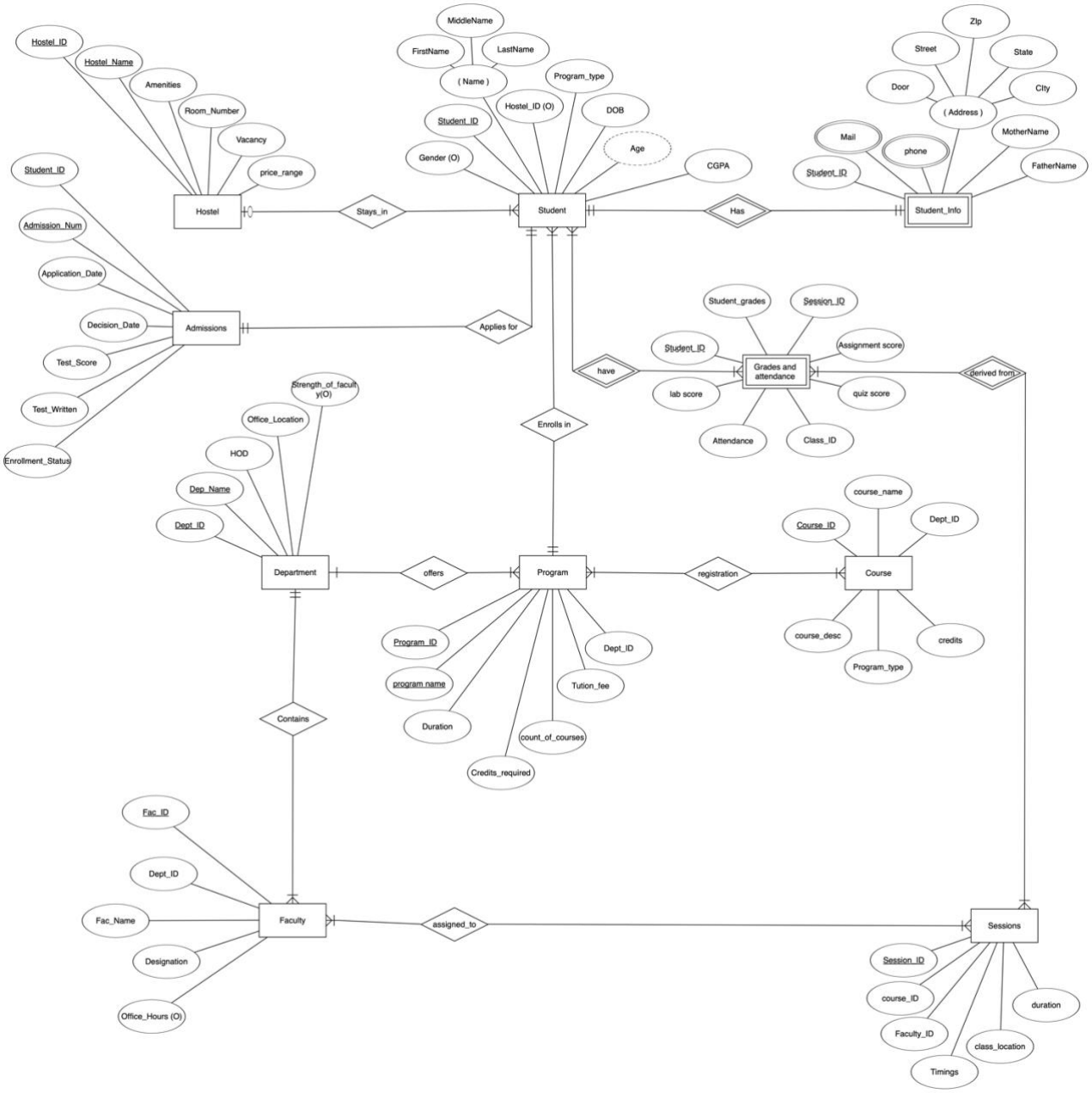

Figure 1 Conceptual design of Entity Relationships

Figure 1 shows the conceptual design of Entity Relationships using multiple entities, attributes and other important relations described below

Implementation of ER Diagram:

Entity: "Student"
Description: A person enrolled in an educational institution or program.
Attributes:
Student_ID: A unique identifier for each pupil.
FirstName: The student's given name or first name.
MiddleName: The student's middle name or initial.
LastName: The student's surname or last name.
Date of birth: The date of the student's birth.
The gender of the pupil, which is typically male or female.
Age: The student's age, calculated based on their date of birth.
Hostel_ID: The identification number for the student's hostel or dormitory, if applicable.
program_type: The type of program or course in which the student is currently enrolled.

These attributes are used to manage and monitor student data in the system of an educational institution.

Entity: “Faculty”
Description: A teacher or professor at an educational institution is described by this entity.
Attributes:
Fac_ID: The system-wide identifier for each faculty member.
FName: The given name or first name of the faculty member.
LName: The last name or surname of the faculty member.
Designation: The academic faculty member's employment title or position within the institution.
Office_hours: The time during which a faculty member is available to meet with students or other faculty members.
Dept_ID: The identifier for the academic department the faculty member belongs to.

These attributes are used to manage and monitor faculty information. The Fac_ID attribute is used to uniquely identify each faculty member. Name attributes (FName and LName) enable the system to store and retrieve faculty member names. The Designation attribute provides details regarding the faculty member's employment title or position. The Office_hours attribute enables the system to retain and retrieve information regarding faculty members' meeting availability. The Dept_ID attribute enables the system to store and retrieve data regarding the academic department to which the faculty member belongs.

Entity: "Course"
Description: A class or subject offered by the educational institution.
Attributes:
Course_ID: A unique identifier for each course in the system.
Course_Name: The name or title of the course.
Duration: The length of time the course is offered, such as "one semester" or "one year".
Course_desc: A brief description of the course content and objectives.
Credits: The number of credits awarded for successfully completing the course.
Program_type: The type of program or degree for which the course is required or elective.

In the educational institution's system, these attributes are utilized to manage and monitor course information. The Course_ID attribute is used to uniquely identify each course.

The Course_Name attribute specifies the course's name or title. The Duration attribute specifies the course's duration of availability. The Course_desc attribute provides a concise summary of the course's objectives and content. The Credits attribute denotes the quantity of credits awarded for completing the course successfully. The Program_type attribute describes the type of program or degree for which the course is required or optional.

Entity: "Department"
Description: An academic department at the educational institution.
Attributes:
Dept_ID: A unique identifier for each department in the system.
Dept_Name: The name of the department.
Courses_offered: The list of courses offered by the department.
HOD: The head of the department.
Office_Location: The location of the department's main office.

These attributes are used to administer and monitor departmental data within the system of an educational institution. The Dept_ID attribute functions as the department's unique identifier. The Dept_Name attribute specifies the department's name. The Courses_offered attribute lists the department's available courses. The HOD attribute designates the department chief. The Office_Location attribute specifies where the department's primary office is located.

Entity: "Admissions"
Description: The process of enrolling a student in the educational institution.
Attributes:
Student_ID: A unique identifier for the student who is applying for admission.
Admission_Number: A unique identifier assigned to the student upon admission to the institution.
Application_date: The date the student submitted the admission application.
Decision_Date: The date the admission decision was made.
Test_written: The type of test (if any) the student took as part of the admission process.
Test_Score: The score the student received on the admission test (if applicable).
Enrollment_Status: The status of the student's admission, such as "accepted", "rejected", or "waitlisted".

These attributes are used to oversee and monitor the admissions process for applicants to an educational institution. The Student_ID attribute functions as the student's unique identifier when applying for admission. Admission_Number is a unique identifier assigned to each pupil upon enrolment. The Application_date attribute specifies the date that the pupil submitted his or her application for admission. Decision_Date is the date that the admissions decision was made. The Test_written attribute indicates whether the applicant was required to take a test. The Test_Score attribute stores the student's grade on the admissions examination. (if applicable). The Enrollment_Status attribute stores the admission status of the pupil, such as "accepted," "rejected," or "waitlisted."

Entity: "Program"
Description: A course of study offered by the educational institution leading to a degree or diploma.
Attributes:
Program_ID: A unique identifier for each program in the system.
Program_Name: The name or title of the program.
Credits_required: The number of credits required to complete the program.
Duration: The length of time required to complete the program.
Number of Courses: The total number of courses required to complete the program.
Tuition_fee: The cost of tuition for the program.
Dept_ID: The identifier for the academic department that offers the program.

These attributes are utilized to administer and monitor program information within the system of an educational institution. The Program_ID attribute functions as the program's unique identifier. The Program_Name attribute specifies the program's name or title. The Credits_required attribute specifies the required quantity of credits to conclude the program. The Duration attribute specifies the amount of time necessary to finish the program. The Number of Courses attribute specifies the total number of required courses for program completion. The Tuition_fee attribute specifies the program's tuition fee. The Dept_ID attribute identifies the academic department offering the program.

Entity: "Grades and Attendance"
Description: A record of a student's academic performance and attendance in a specific class for a particular session.
Attributes:
Student_ID: A unique identifier for the student.
Session_ID: A unique identifier for the class session.
Assignment_score: The score the student received on their assignments.
Lab_score: The score the student received on their laboratory work.
Attendance: The percentage of classes the student attended.
Quiz_Score: The score the student received on their quizzes.
Student_grades: The overall grade the student received in the class.
Class_ID: The unique identifier for the class.

These attributes are used to manage and monitor a student's academic performance and attendance in a particular class for a specific session. The Student_ID attribute functions as a unique identifier for the student. The Session_ID attribute is the class session's unique identifier. The Assignment_score attribute stores the student's assignment grade. The Lab_score attribute stores the student's laboratory performance grade. The Attendance attribute indicates the percentage of classes the student attended. The Quiz_Score attribute stores the student's quiz score. The Student_grades attribute indicates the student's cumulative grade in the class. The Class_ID attribute serves as the class's unique identifier.

Entity: "Student Information"
Description: A record of personal information for a student.
Attributes:
Student_ID: A unique identifier for the student (primary key).
Mail: The email address of the student.
Phone: The phone number of the student.
Father_Name: The name of the student's father.

Mother_name: The name of the student's mother.
Address: The address of the student, including door number, street, city, state, and zip code.

These attributes are used to store a student's personal information. The Student_ID attribute functions as the student's unique identifier. (Primary key). The Mail attribute contains the student's email address. The Phone attribute contains the student's phone number. The Father_Name attribute holds the student's father's name. The Mother_name attribute holds the student's mother's name. The Address attribute stores the student's address, including the door number, street, city, state, and postal code.

Entity: "Session"
Description: A record of a class session for a course.
Attributes:
Session_ID: A unique identifier for the class session.
Class_Location: The location where the class is held.
Course_ID: A unique identifier for the course being taught in the session.
Faculty_ID: A unique identifier for the faculty member teaching the course.
Timings: The time of day when the class is held.
Duration: The length of the class session.

These attributes are used to manage and track a class session for a course. The class session is uniquely identified by the Session_ID property. The location of the class is kept in the Class_Location attribute. The course being taught in the session is uniquely identified by the Course_ID attribute. The faculty member who is instructing the course is uniquely identified by the Faculty_ID attribute. The class's time of day is indicated by the Timings attribute. Finally, the Duration attribute stores the length of the class session.

Entity: "Hostel"
Description: A record of a hostel or dormitory.
Attributes:
Hostel_ID: A unique identifier for the hostel (primary key).
Hostel_Name: The name of the hostel or dormitory.
Amenities: The facilities and services available to the students in the hostel.
Room_Number: The number of rooms in the hostel.
Vacancy: The number of available rooms in the hostel.
Price_Range: The price range for the rooms in the hostel.

These characteristics are employed to control and monitor data pertaining to a hotel or residence. The hostel's unique identifier is the Hostel_ID attribute. (Primary key). The name of the hostel or room is stored in the Hostel_Name property. The amenities attribute lists the amenities and services that the hostel's guests can use. The entire number of accommodations in the hotel is kept in the Room_Number attribute. The number of accommodations the hotel has accessible is indicated by the Vacancy attribute. The price range for the hostel's accommodations is lastly stored in the Price_Range property.

Multivalued Attributes:

For the Student_info which is a weak entity, there are 2 multivalued attributes that are mail and phone. A single student can have multiple phone numbers and mail addresses to be contacted so for our database design, these 2 are the multivalued attributes.

Composite Attributes:

For this design, we have Student name, Faculty name and Address as the composite attributes. Both student name and faculty name are further sub-divided as first name, middle name and last name. For address the sub attributes are door, street, zip, state and city.

Derived Attributes:

We have taken only one derived attribute that is age. Age is a derived attribute from DOB which is itself an attribute for the student entity

Weak Entity:

We have taken Student_info and Grades and Attendance as our weak entities for this database design. Both these weak entities are related to the student entity.

Strong entity:

The student, program, course entities are examples of strong entities. They do not rely on any other entities for representation and identification as they can exist uniquely.

Participations:

Total participation:

Every entity in the first entity set must have a relationship with at least one entity in the second entity set in order for there to be total involvement. In other words, there cannot be any entity in the first set without a corresponding entity in the second set. Total participation, also known as mandatory participation, is indicated by a double line joining the two entities. A student must have a matching record in the student information entity, and each record in the student information entity must be connected to a student, as in the case of a student and their personal information. Similarly, each student must have at least one admission record, at least one program enrolment record, and at least one course enrolment record. Double lines connecting the entities are used to indicate these relationships and total participation.

Partial participation:

A situation where a person or group participates in an activity or circumstance to some degree but not fully or completely is referred to as partial participation. Partial involvement in the context of a hostel and a student could imply that the student is residing in the hostel but is not actively participating in the events or socialising opportunities that the hostel offers.

For instance, a student may decide to reside in a hostel but spend the majority of their time studying by themselves in their rooms, rather than participating in the social events the hostel hosts. Alternatively, the student may select only those things that fit with their interests or preferences and take part in some but not all of them.

Cardinality Ratios:

One to one relationship:

One entity has a direct connection to another entity in such a way that one item in the first entity corresponds to one and only one item in the second entity, and vice versa. This is what is meant by the term "one-to-one relationship," which is a more straightforward explanation of the concept. For instance, in the context of students and student information, a one-to-one relationship can be established by having each student have their own unique collection of personal information (for example, their name, date of birth, and address), and each set of personal information corresponding to only one student. This indicates that there cannot be two or more individuals who share the same information regarding their personal details. The relationship between applicants and students provides yet another illustration of a one-to-one relationship. There can be only one student associated with an admittance record at any given time, and each individual student can have only one admission record. This indicates that a student cannot have numerous admission records, and that a single admission record cannot be linked to more than one student at a time.

One to many relationships:

A one-to-many relationship is one in which one entity is directly linked to another in that each item in the second entity corresponds to just one item in the first entity, but there are one or more items in the first entity. Hostels and students, for instance, can have a one-to-many relationship where one student can be allocated to only one hostel, but a hostel can house multiple students. This implies that while each hostel may be associated with multiple students, each student is only associated with one hostel. The connection between students and programs is another illustration of a one-to-many relationship. Although a student may enrol in numerous programs, only one group of students may be enrolled in any given program at any given time. This implies that each student can be connected to numerous programs, but that each program has a particular group of students that are connected to it.

Many to many relationships:

A many-to-many relationship is when several things in one entity can be linked to several items in another entity. In other words, every element in the first entity may be connected to a variety of elements in the second, and vice versa. A many-to-many connection can be created, for instance, in the context of programs and courses, where each program can offer a variety of courses, and each course can be offered by various programs. This implies that different programs may be linked to various studies. In the context of students, grades, and attendance, we can see another illustration of a many-to-many connection. Each student may take more than one class, receive more than one score, and have multiple students linked with each attendance record. As a result, a student may have numerous grades and attendance records, and multiple students may be linked to a single grade or attendance record. Last but not least, a many-to-many relationship between faculty and sessions can be created in which each faculty member can teach a variety of sessions, and each session can be taught by a variety of faculty members. This implies that different staff members may be connected to various sessions.

Identifying and Non-Identifying Relationships:

The student and student_Info entities are in identifying relationship because the student_Info do not have a unique identifier to identify the entity uniquely. The admission entity and the student entity are in non-identifying relationship as both entities can be represented uniquely without dependency.

Enhanced Entity Relationship Model (EER Model):

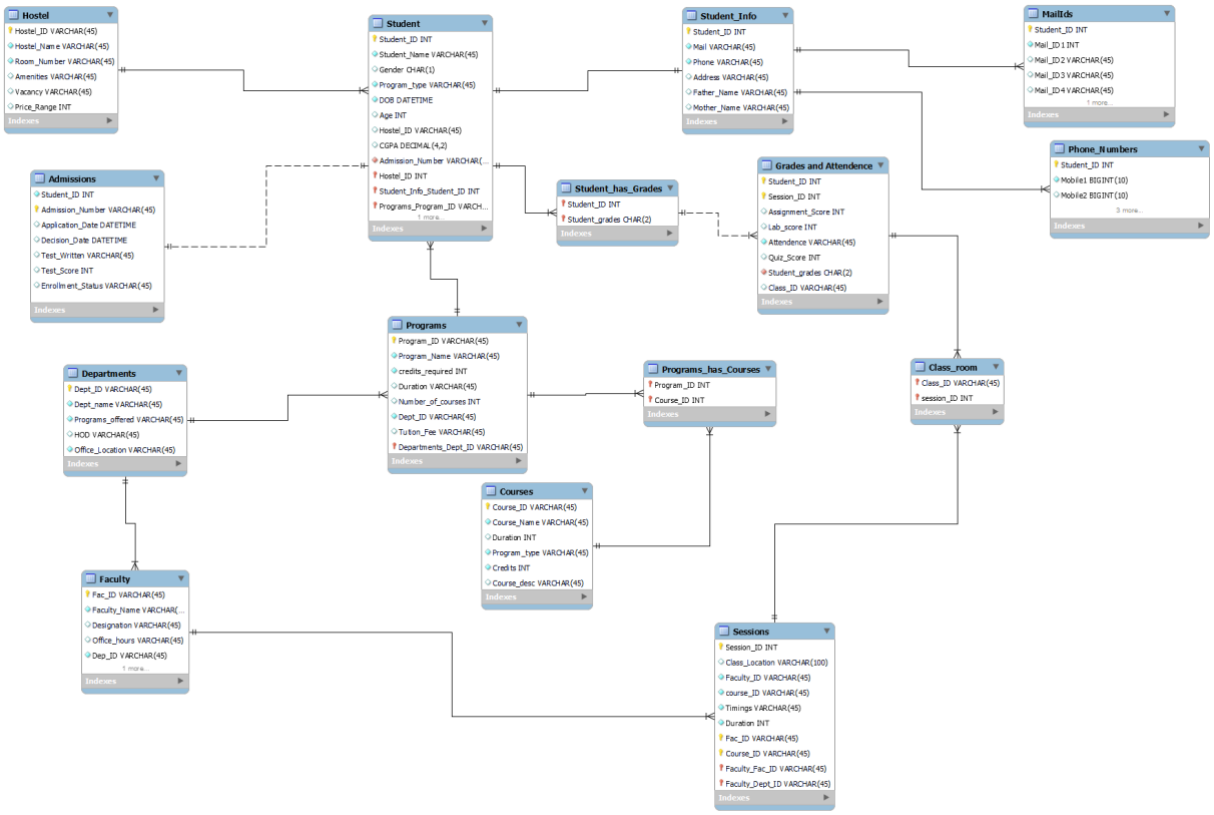

Figure 2 Enhanced Entity Relationship

Implementation of EER:

For creating an EER diagram, we used MySQL Workbench. Instead of going to the query tab we can go to the 2nd option in left menu of the Workbench through which we can design and view the EER diagram for our database. Using different tools, we can create the EER design as per our requirement [4].

The entity Student has an attribute hostel_id, which uniquely identifies each of the row in the entity Hostel, Program_type attribute is refferd as program_ID attribute in the "Program" entity, similarly each of the table interrelated to "Student" Entity can be uniquely related to other. Program. By the same pattern every entity can be related.

Every entity provided in the EER model is interrelated to each of the other entity, we are relating them using the primary and composite keys to manage the attributes in the entities. All the entities in the table except for "Grades and Attendance" has Primary Key. "Grades and Attendance" entity can be identified with the composite key by taking attributes "Student_ID" and "Session_ID". So, every entity in the model is uniquely identified, by means of which we can manage the data.

Database Development:

SUPER_SIX Database :

```
1
2   CREATE DATABASE IF NOT EXISTS SUPER_SIX;
3   USE SUPER_SIX;
4
```

The snippet of code contains two SQL commands.

CREATE DATABASE IF NOT EXISTS SUPER_SIX is the first command; it creates a new database named "SUPER_SIX" if it does not already exist. The "IF NOT EXISTS" clause of this command guarantees the database is only created if it does not already exist.

USE SUPER_SIX is the second command; it selects the "SUPER_SIX" database for use. This indicates that subsequent SQL commands will be executed against this database.

Admission Table:

```
6   CREATE TABLE Admissions (
7   Student_ID INT NOT NULL,
8   Admission_Number VARCHAR(45) PRIMARY KEY,
9   Application_Date DATETIME,
10  Decision_Date DATETIME,
11  Test_Written VARCHAR(45),
12  Test_Score INT,
13  Enrollment_Status VARCHAR(45)
14  );
```

The code snippet creates a table called "Admissions" with columns representing various attributes related to student admissions.

The table has seven columns:

1. "Student_ID" column of type INT which is not nullable (i.e., must contain a value). This column represents the unique identifier of the student applying for admission.
2. "Admission_Number" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier assigned to the admission application.
3. "Application_Date" column of type DATETIME which represents the date and time when the application was submitted.

4. "Decision_Date" column of type DATETIME which represents the date and time when the admission decision was made.
5. "Test_Written" column of type VARCHAR(45) which represents the name of the test written by the student as part of the admission process.
6. "Test_Score" column of type INT which represents the score obtained by the student on the admission test.
7. "Enrollment_Status" column of type VARCHAR(45) which represents the status of the student's enrolment (e.g., accepted, rejected, waitlisted).

The primary key for the table is the "Admission_Number" column, which ensures that each admission application has a unique identifier. This column will be used to uniquely identify each row in the table.
There are no foreign keys or relationships defined in this table.

Hostel Table:

```
16  CREATE TABLE Hostel (
17  Hostel_ID VARCHAR(45) PRIMARY KEY,
18  Hostel_Name VARCHAR(45) NOT NULL,
19  Room_Number VARCHAR(45),
20  Amenities VARCHAR(45),
21  Vacancy VARCHAR(45),
22  Price_Range INT
23  );
```

The code snippet creates a table called "Hostel" with columns representing various attributes related to hostels.
The table has six columns:

1. "Hostel_ID" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier of the hostel.
2. "Hostel_Name" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the name of the hostel.
3. "Room_Number" column of type VARCHAR(45) which represents the room number of the hostel.
4. "Amenities" column of type VARCHAR(45) which represents the amenities available in the hostel.
5. "Vacancy" column of type VARCHAR(45) which represents the vacancy status of the hostel.
6. "Price_Range" column of type INT which represents the price range of the hostel.

The primary key for the table is the "Hostel_ID" column, which ensures that each hostel has a unique identifier. This column will be used to uniquely identify each row in the table.

Departments Table:

```
25 • CREATE TABLE Departments (
26   Dept_ID VARCHAR(45) PRIMARY KEY,
27   Dept_name VARCHAR(45) NOT NULL,
28   programs_offered VARCHAR(45) NOT NULL,
29   HOD VARCHAR(45),
30   Office_Location VARCHAR(45)
31   );
```

The code snippet creates a table called "Departments" with columns representing various attributes related to academic departments.
The table has five columns:

1. "Dept_ID" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier of the department.
2. "Dept_name" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the name of the department.
3. "programs_offered" column of type VARCHAR(45) which is not nullable. This column represents the academic programs offered by the department.
4. "HOD" column of type VARCHAR(45) which represents the name of the head of department.
5. "Office_Location" column of type VARCHAR(45) which represents the location of the departmental office.

The primary key for the table is the "Dept_ID" column, which ensures that each department has a unique identifier. This column will be used to uniquely identify each row in the table.

Programs Table:

```
33 • CREATE TABLE Programs (
34   Program_ID VARCHAR(45) PRIMARY KEY,
35   Program_Name VARCHAR(45) NOT NULL,
36   credits_required INT NOT NULL,
37   Duration VARCHAR(45),
38   Number_of_courses INT,
39   Tuition_Fee INT,
40   Department VARCHAR(45),
41   FOREIGN KEY (Department)
42   REFERENCES Departments (Dept_ID)
43   );
```

The code snippet creates a table called "Programs" with columns representing various attributes related to academic programs offered by a department.
The table has seven columns:

1. "Program_ID" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier of the program.
2. "Program_Name" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the name of the program.

3. "credits_required" column of type INT which is not nullable. This column represents the total number of credits required to complete the program.
4. "Duration" column of type VARCHAR(45) which represents the duration of the program.
5. "Number_of_courses" column of type INT which represents the number of courses in the program.
6. "Tuition_Fee" column of type INT which represents the tuition fee for the program.
7. "Department" column of type VARCHAR(45) which references the "Dept_ID" column in the "Departments" table using a foreign key constraint.

The primary key for the table is the "Program_ID" column, which ensures that each program has a unique identifier. This column will be used to uniquely identify each row in the table.
The "Department" column in this table references the "Dept_ID" column in the "Departments" table using a foreign key constraint. This relationship indicates that each program belongs to a specific department. The foreign key constraint ensures that only valid department IDs are entered into the "Department" column of the "Programs" table.
This table could be used to store information about academic programs, such as their names, the total number of credits required, duration, number of courses, tuition fees, and the department to which they belong.

Student Table:

```
45  CREATE TABLE Student (
46  Student_ID INT PRIMARY KEY,
47  First_Name VARCHAR(45) NOT NULL,
48  Middle_Name VARCHAR(45),
49  Last_Name VARCHAR(45) NOT NULL,
50  Gender CHAR(1),
51  Program_type VARCHAR(45) NOT NULL,
52  DOB DATETIME NOT NULL,
53  Age INT,
54  Hostel_ID VARCHAR(45),
55  CGPA DECIMAL(4,2),
56  Admission_Number VARCHAR(45) NOT NULL,
57  FOREIGN KEY (Admission_Number)
58  REFERENCES Admissions (Admission_Number),
59  FOREIGN KEY (Hostel_ID)
60  REFERENCES Hostel (Hostel_ID),
61  FOREIGN KEY (Program_type)
62  REFERENCES Programs (Program_ID)
63  );
```

The code snippet creates a table called "Student" with columns representing various attributes related to students enrolled in academic programs.
The table has eleven columns:

1. "Student_ID" column of type INT which is defined as the primary key of the table. This column represents the unique identifier of the student.
2. "First_Name" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the first name of the student.

3. "Middle_Name" column of type VARCHAR(45) which represents the middle name of the student.
4. "Last_Name" column of type VARCHAR(45) which is not nullable. This column represents the last name of the student.
5. "Gender" column of type CHAR(1) which represents the gender of the student.
6. "Program_type" column of type VARCHAR(45) which is not nullable. This column represents the academic program that the student is enrolled in.
7. "DOB" column of type DATETIME which is not nullable. This column represents the date of birth of the student.
8. "Age" column of type INT which represents the age of the student.
9. "Hostel_ID" column of type VARCHAR(45) which references the "Hostel_ID" column in the "Hostel" table using a foreign key constraint.
10. "CGPA" column of type DECIMAL(4,2) which represents the cumulative grade point average of the student.
11. "Admission_Number" column of type VARCHAR(45) which references the "Admission_Number" column in the "Admissions" table using a foreign key constraint.

The primary key for the table is the "Student_ID" column, which ensures that each student has a unique identifier. This column will be used to uniquely identify each row in the table.
The "Admission_Number" column in this table references the "Admission_Number" column in the "Admissions" table using a foreign key constraint. This relationship indicates that each student is associated with a specific admission record.
The "Hostel_ID" column in this table references the "Hostel_ID" column in the "Hostel" table using a foreign key constraint. This relationship indicates that each student may be associated with a specific hostel.
The "Program_type" column in this table references the "Program_ID" column in the "Programs" table using a foreign key constraint. This relationship indicates that each student is enrolled in a specific academic program.
This table could be used to store information about students, such as their names, gender, date of birth, age, academic program, admission number, CGPA, and hostel information.

Phone_Numbers Table:

```
76  CREATE TABLE Phone_Numbers(
77  Student_ID INT PRIMARY KEY,
78  mobile1 LONG NOT NULL,
79  mobile2 LONG,
80  mobile3 LONG,
81  mobile4 LONG,
82  FOREIGN KEY (Student_ID)
83  REFERENCES student(Student_ID)
84  );
```

The "Phone_Numbers" table is created to store multivalued attributes related to the "Student" table. It has five columns:

1. "Student_ID" column of type INT which is defined as the primary key of the table. This column references the "Student_ID" column in the "Student" table using a foreign key constraint. This relationship ensures that each row in the "Phone_Numbers" table corresponds to a unique student in the "Student" table.
2. "mobile1" column of type LONGwhich is not nullable (i.e., must contain a value). This column represents the primary mobile number of the student.
3. "mobile2" column of type LONG which represents a secondary mobile number of the student.
4. "mobile3" column of type LONGwhich represents a tertiary mobile number of the student.
5. "mobile4" column of type LONG which represents another mobile number of the student.

The primary key of the "Phone_Numbers" table is the "Student_ID" column, which ensures that each student has a unique entry in the table. The "Student_ID" column also serves as a foreign key that references the "Student" table. This relationship ensures that the mobile numbers stored in this table correspond to valid student records in the "Student" table.

Mail_ID Table:

```
86  CREATE TABLE Mail_ID(
87  Student_ID INT PRIMARY KEY,
88  Mail_ID1 VARCHAR(45) NOT NULL,
89  Mail_ID2 VARCHAR(45),
90  Mail_ID3 VARCHAR(45),
91  Mail_ID4 VARCHAR(45),
92  FOREIGN KEY (Student_ID)
93  REFERENCES student(Student_ID)
94  );
```

Yes, the "Mail_ID" table is created to store multivalued attributes related to the "Student" table. It has five columns:

"Student_ID" column of type INT which is defined as the primary key of the table. This column references the "Student_ID" column in the "Student" table using a foreign key constraint. This relationship ensures that each row in the "Mail_ID" table corresponds to a unique student in the "Student" table.

1. "Mail_ID1" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the primary email address of the student.
2. "Mail_ID2" column of type VARCHAR(45) which represents a secondary email address of the student.
3. "Mail_ID3" column of type VARCHAR(45) which represents a tertiary email address of the student.
4. "Mail_ID4" column of type VARCHAR(45) which represents another email address of the student.

The primary key of the "Mail_ID" table is the "Student_ID" column, which ensures that each student has a unique entry in the table. The "Student_ID" column also serves as a foreign key

that references the "Student" table. This relationship ensures that the email addresses stored in this table correspond to valid student records in the "Student" table.

Student_Info Table

```
65 • CREATE TABLE Student_Info (
66   Student_ID INT PRIMARY KEY,
67   Mail VARCHAR(45) NOT NULL,
68   Phone LONG NOT NULL,
69   Address VARCHAR(45),
70   Father_Name VARCHAR(45),
71   Mother_Name VARCHAR(45),
72   FOREIGN KEY (Student_ID)
73   REFERENCES Student (Student_ID)
74   );
```

The "Student_Info" table is created to store additional information about students that is not captured in other tables. It has six columns:

1. "Student_ID" column of type INT which is defined as the primary key of the table. This column references the "Student_ID" column in the "Student" table using a foreign key constraint. This relationship ensures that each row in the "Student_Info" table corresponds to a unique student in the "Student" table.
2. "Mail" column of type VARCHAR(45) which represents the primary email address of the student. This column is redundant since the "Mail_ID" table already stores the email addresses of each student.
3. "Phone" column of type VARCHAR(45) which represents the phone number of the student. This column is redundant since the "Phone_Numbers" table already stores the phone numbers of each student.
4. "Address" column of type VARCHAR(45) which represents the address of the student.
5. "Father_Name" column of type VARCHAR(45) which represents the name of the student's father.
6. "Mother_Name" column of type VARCHAR(45) which represents the name of the student's mother.

The primary key of the "Student_Info" table is the "Student_ID" column, which ensures that each student has a unique entry in the table. The "Student_ID" column also serves as a foreign key that references the "Student" table. This relationship ensures that the information stored in this table corresponds to valid student records in the "Student" table.

Courses Table:

```
CREATE TABLE Courses (
Course_ID VARCHAR(45) PRIMARY KEY,
Course_Name VARCHAR(45) NOT NULL,
Duration INT,
Program_type VARCHAR(45) NOT NULL,
Credits INT NOT NULL,
Course_desc VARCHAR(150),
Department VARCHAR(45),
FOREIGN KEY (Department)
REFERENCES Departments (Dept_ID)
);
```

The code snippet creates a table called "Courses" with columns representing various attributes related to academic courses.

The table has seven columns:

1. "Course_ID" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier of the course.
2. "Course_Name" column of type VARCHAR(45) which is not nullable. This column represents the name of the course.
3. "Duration" column of type INT which represents the duration of the course.
4. "Program_type" column of type VARCHAR(45) which is not nullable. This column represents the academic program that the course is associated with.
5. "Credits" column of type INT which is not nullable. This column represents the number of credits associated with the course.
6. "Course_desc" column of type VARCHAR(150) which represents a brief description of the course.
7. "Department" column of type VARCHAR(45) which references the "Dept_ID" column in the "Departments" table using a foreign key constraint.

The primary key for the table is the "Course_ID" column, which ensures that each course has a unique identifier. This column will be used to uniquely identify each row in the table.

The "Department" column in this table references the "Dept_ID" column in the "Departments" table using a foreign key constraint. This relationship indicates that each course is associated with a specific department.

This table could be used to store information about academic courses, such as their names, duration, credit hours, program affiliation, department association, and a brief description.

Program_has_Courses Table:

```
CREATE TABLE Program_has_Courses(
Course_ID VARCHAR(45),
Program_ID VARCHAR(45),
PRIMARY KEY(Course_ID,Program_ID),
FOREIGN KEY (Course_ID)
REFERENCES Courses(Course_ID),
FOREIGN KEY (Program_ID)
REFERENCES Programs(Program_ID)
);
```

The code snippet creates a junction table called "Program_has_Courses" with two foreign key constraints to the "Courses" and "Programs" tables.

The table has two columns:

1. "Course_ID" column of type VARCHAR(45) which references the "Course_ID" column in the "Courses" table using a foreign key constraint. This relationship indicates that each row in the "Program_has_Courses" table is associated with a specific course.
2. "Program_ID" column of type VARCHAR(45) which references the "Program_ID" column in the "Programs" table using a foreign key constraint. This relationship indicates that each row in the "Program_has_Courses" table is associated with a specific academic program.

The primary key for the table is a composite key consisting of both "Course_ID" and "Program_ID" columns. This ensures that each combination of course and program is unique in the table.

This table could be used to store information about which courses are included in which academic programs.

Faculty Table:

```
CREATE TABLE Faculty (
Fac_ID VARCHAR(45) PRIMARY KEY,
Faculty_Name VARCHAR(45) NOT NULL,
Designation VARCHAR(45),
Office_hours VARCHAR(45),
Dept_ID VARCHAR(45) NOT NULL,
FOREIGN KEY (Dept_ID)
REFERENCES Departments (Dept_ID)
);
```

This code snippet creates a table called "Faculty" with columns representing various attributes related to academic faculty.

The table has five columns:

1. "Fac_ID" column of type VARCHAR(45) which is defined as the primary key of the table. This column represents the unique identifier of the faculty.
2. "Faculty_Name" column of type VARCHAR(45) which is not nullable (i.e., must contain a value). This column represents the name of the faculty member.
3. "Designation" column of type VARCHAR(45) which represents the position held by the faculty member.
4. "Office_hours" column of type VARCHAR(45) which represents the office hours of the faculty member.
5. "Dept_ID" column of type VARCHAR(45) which is not nullable. This column references the "Dept_ID" column in the "Departments" table using a foreign key constraint. This column represents the department that the faculty member belongs to.

The primary key for the table is the "Fac_ID" column, which ensures that each faculty member has a unique identifier. This column will be used to uniquely identify each row in the table.

The "Dept_ID" column in this table references the "Dept_ID" column in the "Departments" table using a foreign key constraint. This relationship indicates that each faculty member belongs to a specific department.

This table could be used to store information about academic faculty, such as their names, positions, office hours, and department information.

Sessions Table:

```
141 • CREATE TABLE Sessions (
142     Session_ID INT PRIMARY KEY,
143     Class_Location VARCHAR(100),
144     Faculty_ID VARCHAR(45) NOT NULL,
145     Course_ID VARCHAR(45) NOT NULL,
146     Timings VARCHAR(45),
147     Duration INT,
148     FOREIGN KEY (Faculty_ID, Course_ID)
149     REFERENCES Faculty_teaches_Classes (Fac_ID, Course_ID)
150     );
```

The code snippet creates a table called "Sessions" with columns representing various attributes related to academic sessions.

The table has six columns:

1. "Session_ID" column of type INT which is defined as the primary key of the table. This column represents the unique identifier of the session.
2. "Class_Location" column of type VARCHAR(100) which represents the location where the session will take place.
3. "Faculty_ID" column of type VARCHAR(45) which is not nullable. This column represents the unique identifier of the faculty member who will conduct the session.
4. "Course_ID" column of type VARCHAR(45) which is not nullable. This column represents the unique identifier of the course for which the session is being conducted.

5. "Timings" column of type TIME which represents the time when the session will begin.
6. "Duration" column of type INT which represents the duration of the session.

The "Faculty_ID" and "Course_ID" columns in this table reference the "Fac_ID" and "Course_ID" columns in the "Faculty_teaches_Classes" table using a foreign key constraint. This relationship indicates that each session is associated with a specific course that is being taught by a specific faculty member.

This table could be used to store information about academic sessions, such as the location, timings, and duration of the session, and the faculty member and course associated with it.

Class_room Table:

```
152 • CREATE TABLE Class_room(
153   Class_ID VARCHAR(45) NOT NULL,
154   Session_ID INT NOT NULL,
155   PRIMARY KEY (Class_ID,Session_ID),
156   FOREIGN KEY (Session_ID)
157   REFERENCES Sessions(Session_ID)
158   );
```

This code creates a table called "Class_room" which contains information about the classrooms used for specific sessions.

The table has two columns:

1. "Class_ID" column of type VARCHAR(45) which represents the unique identifier of the classroom. This column cannot be null.
2. "Session_ID" column of type INT which represents the identifier of the session during which the classroom is used. This column cannot be null.

The primary key for the table is a combination of "Class_ID" and "Session_ID" columns, which ensures that each classroom is uniquely identified for each session.

The "Session_ID" column in this table references the "Session_ID" column in the "Sessions" table using a foreign key constraint. This relationship indicates that each classroom is associated with a specific session.

This table could be used to store information about the classrooms used for various sessions, such as the class location, faculty ID, course ID, timings, and duration.

Grades _and_Attendance Table:

```
160 • CREATE TABLE Grades_and_Attendance (
161     Student_ID INT NOT NULL,
162     Session_ID INT NOT NULL,
163     Assignment_Score INT,
164     Lab_score INT,
165     Attendance VARCHAR(45),
166     Quiz_Score INT,
167     Student_grades CHAR(2) NOT NULL,
168     Class_ID VARCHAR(45),
169     PRIMARY KEY (Student_ID, Session_ID),
170     FOREIGN KEY (Student_ID)
171     REFERENCES Student(Student_ID),
172     FOREIGN KEY (Session_ID)
173     REFERENCES Sessions (Session_ID),
174     FOREIGN KEY (Class_ID)
175     REFERENCES Class_room(Class_ID)
176     );
```

The code snippet creates a table called "Grades_and_Attendance" with columns representing various attributes related to the grades and attendance of students in a particular session.

The table has eight columns:

1. "Student_ID" column of type INT which represents the unique identifier of the student and is not nullable.
2. "Session_ID" column of type INT which represents the unique identifier of the session and is not nullable.
3. "Assignment_Score" column of type INT which represents the score obtained by the student in assignments.
4. "Lab_score" column of type INT which represents the score obtained by the student in labs.
5. "Attendance" column of type VARCHAR(45) which represents the attendance status of the student.
6. "Quiz_Score" column of type INT which represents the score obtained by the student in quizzes.
7. "Student_grades" column of type CHAR(2) which represents the final grade obtained by the student and is not nullable.
8. "Class_ID" column of type VARCHAR(45) which references the "Class_ID" column in the "Class_room" table using a foreign key constraint.

The primary key for the table is the combination of "Student_ID" and "Session_ID" columns, which ensures that each record represents a unique combination of student and session.

The "Student_ID" column in this table references the "Student_ID" column in the "Student" table using a foreign key constraint. This relationship indicates that each record in this table is associated with a specific student.

The "Session_ID" column in this table references the "Session_ID" column in the "Sessions" table using a foreign key constraint. This relationship indicates that each record in this table is associated with a specific session.

The "Class_ID" column in this table references the "Class_ID" column in the "Class_room" table using a foreign key constraint. This relationship indicates that each record in this table is associated with a specific class.

Students_has_grades Table:

```
178 • CREATE TABLE Student_has_grades(
179     Student_ID INT NOT NULL,
180     Student_grades CHAR(2) NOT NULL,
181     PRIMARY KEY(Student_ID,Student_grades),
182     FOREIGN KEY (Student_ID)
183     REFERENCES Student(Student_ID)
184     );
```

The code snippet creates a table called "Student_has_grades" with two columns that represent the relationship between the "Student" table and the "Grades_and_Attendance" table.

The table has two columns:

1. "Student_ID" column of type INT which is defined as the primary key of the table. This column represents the unique identifier of the student in the "Student" table.
2. "Student_grades" column of type CHAR(2) which is not nullable (i.e., must contain a value). This column represents the grades obtained by the student in the "Grades_and_Attendance" table.

The primary key for the table is a composite key that consists of both the "Student_ID" and "Student_grades" columns. This ensures that each combination of student and grades is unique.

This table could be used to store information about the grades obtained by each student in different courses.

References:

[1] *https://insights.blackcoffer.com/wp-content/uploads/2019/02/Student_Database_Management_System.pdf*

[2] *https://www.studocu.com/in/document/g-d-goenka-university/certified-exam-dumps/project-report-on-student-information-management-system-php-mysql/17903752*

[3] *https://thesai.org/Downloads/Volume12No2/Paper_10-Student_Information_System.pdf*

[4] *https://www.edrawsoft.com/article/what-is-eer-diagram.html*

GitHub repository address:

https://github.com/PradeepReddy-Baireddy/MSCS_542L_256_23S_College-Data-Management-System_Super-Six