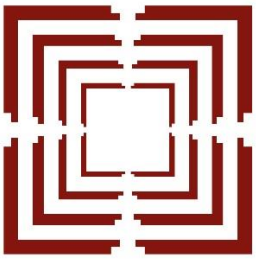


Ahmedabad University

CSE250 – Database Management Systems

Winter Semester – 2022

Student Management System

Date: 05-05-2022

Submitting to: Shefali Naik

AU2040238 – Jainam Shah

AU2040243 – Devansh Purani

AU2040244 – Aryan Bhavsar

AU2040167 – Daksh Suthar

1. Project Details

1.1 Introduction

Today everything has become oriented towards the online virtual world. Even in education in schools and colleges there is a need to manage the data of students, faculties, attendance and much more. So, we have created a database system for managing students' huge data. Using this, one can easily manage the huge data with interactive interface. This reduces the tedious work load of faculty and higher authorities as they will not have to keep all these records and track of students on paper. Our system will be mostly used by faculty and admin. Admin will have privileges to insert new students and faculties in the school's existing database. He/she can comfortably enrol a student into any course, keep the record of every session of each course conducted by respective faculty on what date and time, and also add new courses along with assigned faculty, and also add new faculties employed by the school. Course Faculty has privileges to manage student records like attendance of students, students' marks, grades, etc. He/she can also add what evaluating components will be there in the course like how much weightage they will carry (i.e project can have 50%, end semester exam can have 50% ,etc.)

First thing for the admin is to add details of departments in the Department Table which are department id and department name. Then, the admin can add instructor details like instructor id, name, address, mobile number, his/her respective department, etc into the Instructor Table. Then, the admin can add student details like student id, student department, address, date of birth, mobile number, etc into the Student Table. Later the Instructor will add courses and components which he/she wants to teach and add details like course id, course name, course department into Course Table and add components details like component name and weightage into Component Table. Then, the admin will add details like session date, session time, session number into the Session Table. Now, the instructor or the admin can enrol a student into the course. From here, the instructor can edit the student's attendance data into the Attendance Table by adding details like student id, course id and session id and remark.

So, we can say that by using this student management system it becomes a very easy task for faculty to manage all these records.

1.2 Specifications

Admin

- ✓ Can open any number of departments and then enrol students and faculties into it
- ✓ Enrolment of students in college without concerning about the enrolment numbers – the enrolment numbers will automatically be generated based on the traditional pattern of the Ahmedabad University (AUYYXXXXX)
- ✓ Enrolment of instructors in college without concerning about the enrolment numbers - the enrolment numbers will automatically be generated based on the department of the instructor
- ✓ Can create any number of courses without concerning the ID of the course – the course ID will automatically be generated based on the department of the course
- ✓ Can plan sessions for the courses and also can set the components for the course

- ✓ Can manage the attendance of the students for the scheduled sessions and can also do the assessment of the students based on the created components

Instructor

- ✓ Can enrol any number of students into their courses
- ✓ Can plan sessions for their courses and also can set the components for their courses
- ✓ Can manage the attendance of the students for the scheduled sessions and can also do the assessment of the students based on the created components
- ✓ Can view their daily schedule

Student

- ✓ Can view their assessment and attendance
- ✓ Can view their daily schedule

1.3 System Requirements

1. Operating System: Windows 10
2. Database: MySQL

1.4 How to run the project?

1. Unzip the project folder where you will find one folder named “database”.

2. Open that folder, inside that folder you will find five files named “tables.sql”, “insert.sql”, “functions.sql”, “triggers.sql” and “procedures.sql”.

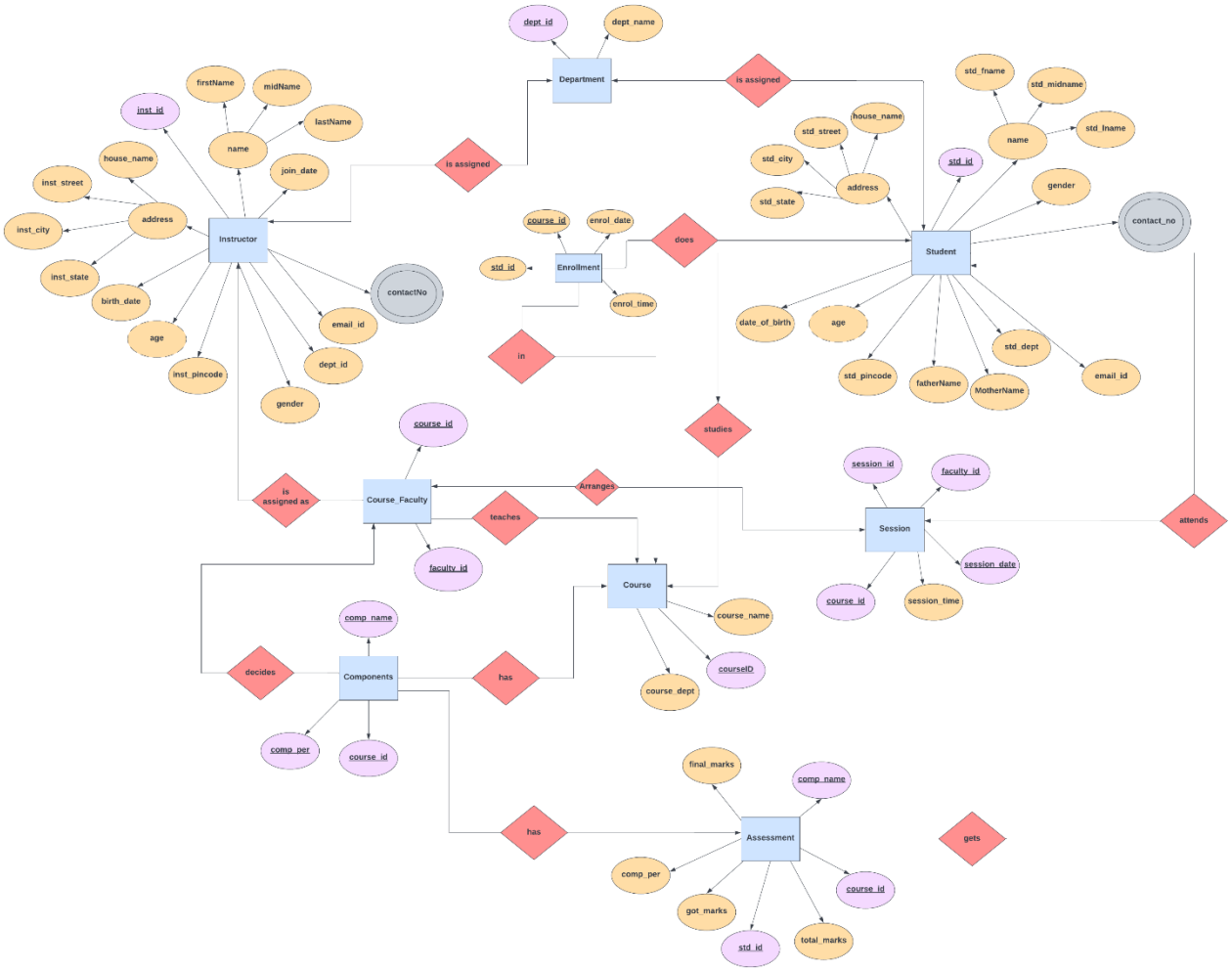
3. First run the “tables.sql” file to create database and tables

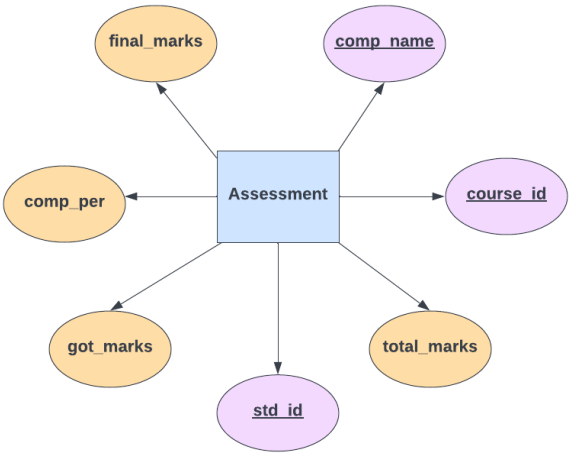
4. Then run “triggers.sql” file to apply appropriate triggers on the tables.

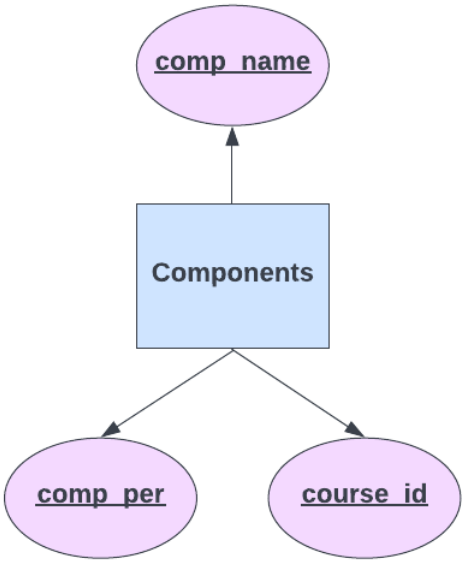
5. Then run “insert.sql” file to insert sample records into tables.

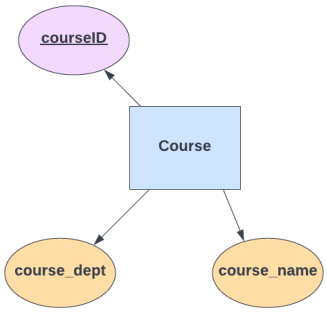
6. Then run “procedures.sql” and “functions.sql”

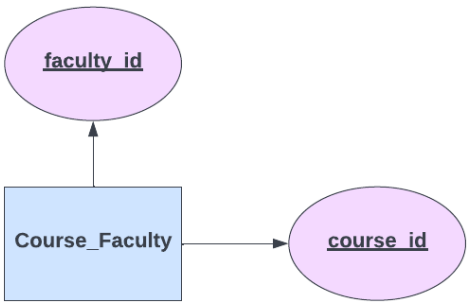
2. Entity Relationship Diagram

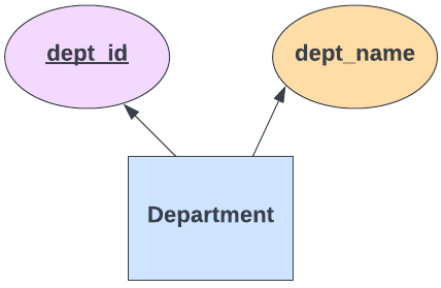


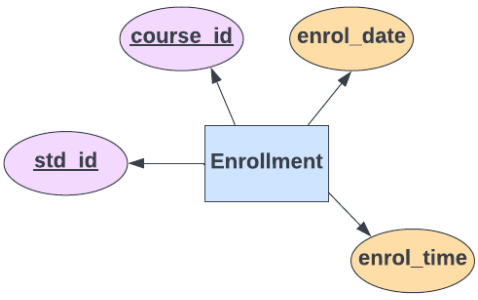

final_marks
comp_name
comp_per
Assessment
course_id
got_marks
total_marks
std_id

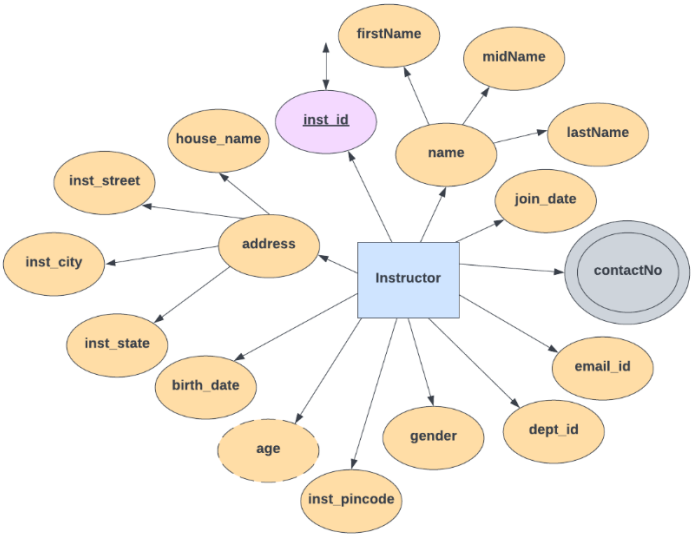

comp_name
Components
comp_per
course_id

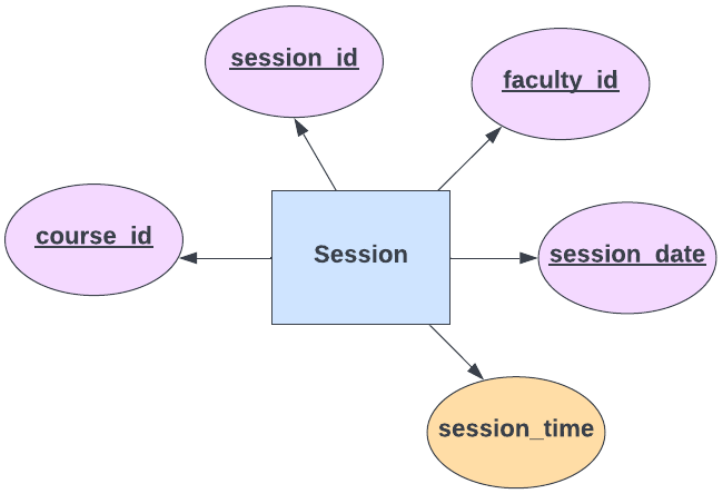

courseID
Course
course_dept
course_name

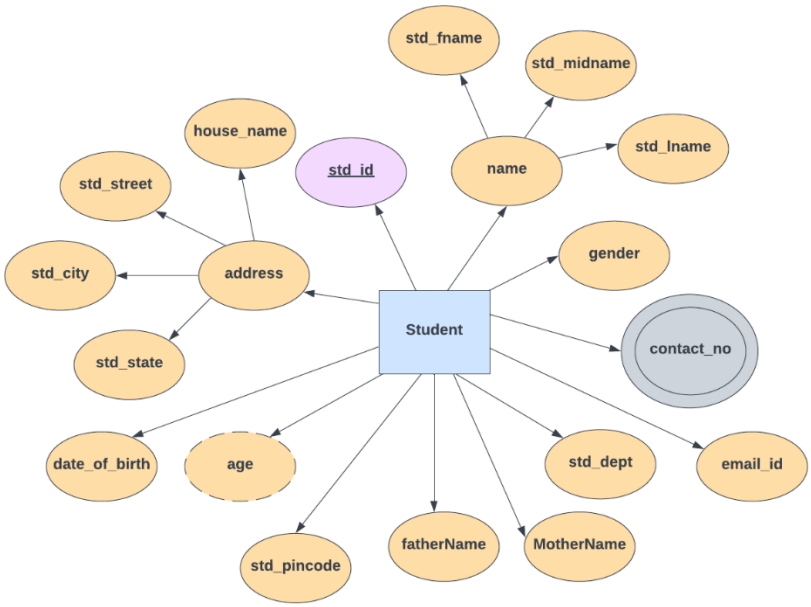

faculty_id
Course_Faculty
course_id


dept_id
dept_name
Department


course_id
enrol_date
std_id
Enrollment
enrol_time

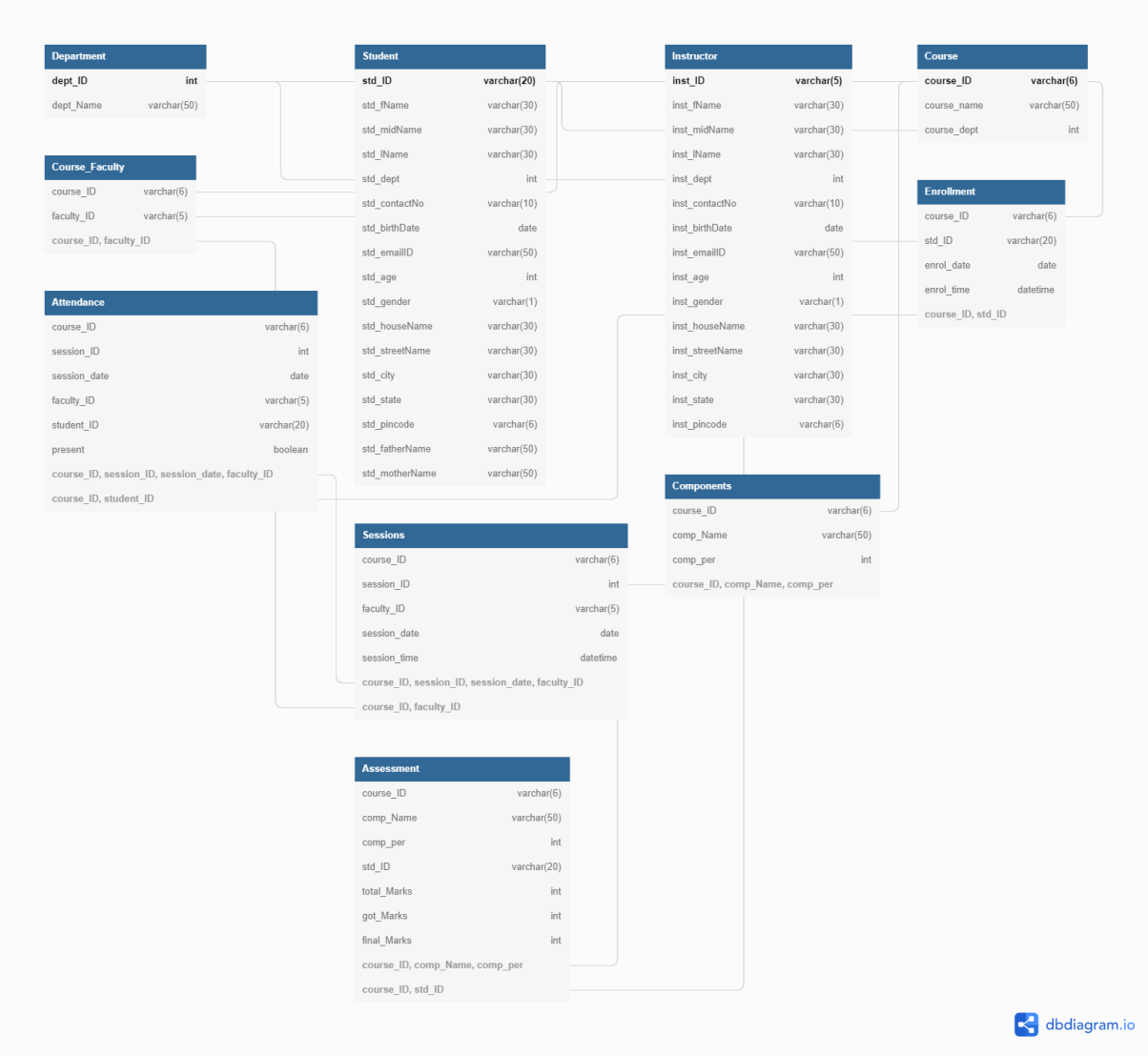

firstName
midName
inst_id
house_name
name
lastName
inst_street
join_date
address
inst_city
Instructor
contactNo
inst_state
email_id
birth_date
gender
dept_id
age
inst_pincode


session_id
faculty_id
course_id
Session
session_date
session_time


std_fname
std_midname
house_name
std_id
name
std_lname
std_street
gender
std_city
address
Student
std_state
contact_no
date_of_birth
age
std_dept
email_id
std_pincode
fatherName
MotherName

3. Tables

3.1 Schema Diagram



3.2 Table description and constraints

Here are the all tables which we have designed in the ER-diagram, along with this there are appropriate constraints like primary and foreign key constraints, not null constraints, unique constraints, default values for some columns etc.

(Note: To view the queries for table creations open **Tables.sql** file under project folder.)

1. Department Table

```
CREATE TABLE Department (
    dept_ID int AUTO_INCREMENT,
    dept_Name varchar(50) NOT NULL,
    PRIMARY KEY(dept_ID)
);
```

Field	Type	Null	Key	Default	Extra
dept_ID	int	NO	PRI	NULL	auto_increment
dept_Name	varchar(50)	NO		NULL	

2. Student Table

```
CREATE TABLE Student (
    std_ID varchar(20),
    std_fName varchar(30) NOT NULL,
    std_midName varchar(30) NOT NULL,
    std_lName varchar(30) NOT NULL,
    std_dept int NOT NULL,
    std_contactNo varchar(10) NOT NULL,
    std_birthDate date NOT NULL,
    std_emailID varchar(50),
    std_age int,
    std_gender varchar(1) NOT NULL,
    std_houseName varchar(30) NOT NULL,
    std_streetName varchar(30) NOT NULL,
    std_city varchar(30) NOT NULL,
    std_state varchar(30) NOT NULL,
    std_pincode varchar(6) NOT NULL,
    std_fatherName varchar(50),
    std_motherName varchar(50),
    PRIMARY KEY(std_ID),
    FOREIGN KEY(std_dept) REFERENCES Department(dept_ID)
);
```

Field	Type	Null	Key	Default	Extra
std_ID	varchar(20)	NO	PRI	AU0000000	
std_fName	varchar(30)	NO		NULL	
std_midName	varchar(30)	NO		NULL	
std_lName	varchar(30)	NO		NULL	
std_dept	int	NO	MUL	NULL	
std_contactNo	varchar(10)	NO		NULL	
std_birthDate	date	NO		NULL	
std_emailID	varchar(50)	YES		NULL	
std_age	int	YES		NULL	
std_gender	varchar(1)	NO		NULL	
std_houseName	varchar(30)	NO		NULL	
std_streetName	varchar(30)	NO		NULL	
std_city	varchar(30)	NO		NULL	
std_state	varchar(30)	NO		NULL	
std_pincode	varchar(6)	NO		NULL	
std_fatherName	varchar(50)	YES		NULL	
std_motherN...	varchar(50)	YES		NULL	

3. Instructor Table

```
CREATE TABLE Instructor (
    inst_ID varchar(5),
    inst_fName varchar(30) NOT NULL,
    inst_midName varchar(30) NOT NULL,
    inst_lName varchar(30) NOT NULL,
    inst_dept int NOT NULL,
    inst_contactNo varchar(10) NOT NULL,
    inst_birthDate date NOT NULL,
    inst_emailID varchar(50) NOT NULL,
    inst_age int,
    inst_gender varchar(1) NOT NULL,
    inst_houseName varchar(30) NOT NULL,
    inst_streetName varchar(30) NOT NULL,
    inst_city varchar(30) NOT NULL,
    inst_state varchar(30) NOT NULL,
    inst_pincode varchar(6) NOT NULL,
    PRIMARY KEY(inst_ID),
    FOREIGN KEY(inst_dept) REFERENCES Department(dept_ID)
);
```

Field	Type	Null	Key	Default	Extra
inst_ID	varchar(5)	NO	PRI	XXX00	
inst_fName	varchar(30)	NO		NULL	
inst_midName	varchar(30)	NO		NULL	
inst_lName	varchar(30)	NO		NULL	
inst_dept	int	NO	MUL	NULL	
inst_contactNo	varchar(10)	NO		NULL	
inst_birthDate	date	NO		NULL	
inst_emailID	varchar(50)	NO		NULL	
inst_age	int	YES		NULL	
inst_gender	varchar(1)	NO		NULL	
inst_houseName	varchar(30)	NO		NULL	
inst_streetName	varchar(30)	NO		NULL	
inst_city	varchar(30)	NO		NULL	
inst_state	varchar(30)	NO		NULL	
inst_pincode	varchar(6)	NO		NULL	

4. Course Table

```
CREATE TABLE Course (
    course_ID varchar(6),
    course_name varchar(50) NOT NULL,
    course_dept int NOT NULL,
    PRIMARY KEY(course_ID),
    FOREIGN KEY(course_dept) REFERENCES Department(dept_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	XXX000	
course_name	varchar(50)	NO		NULL	
course_dept	int	NO	MUL	NULL	

5. Course / Faculty Table

```
CREATE TABLE Course_Faculty (
    course_ID varchar(6) NOT NULL,
    faculty_ID varchar(5) NOT NULL,
    FOREIGN KEY(course_ID) REFERENCES Course(course_ID),
    FOREIGN KEY(faculty_ID) REFERENCES Instructor(inst_ID),
    PRIMARY KEY(course_ID, faculty_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
faculty_ID	varchar(5)	NO	PRI	NULL	

6. Sessions Table

```
CREATE TABLE Sessions (
    course_ID varchar(6) NOT NULL,
    session_ID int NOT NULL,
    faculty_ID varchar(5) NOT NULL,
    session_date date NOT NULL,
    session_time datetime NOT NULL,
    FOREIGN KEY(course_ID, faculty_ID) REFERENCES Course_Faculty(course_ID, faculty_ID),
    PRIMARY KEY(course_ID, session_ID, session_date, faculty_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
session_ID	int	NO	PRI	NULL	
faculty_ID	varchar(5)	NO	PRI	NULL	
session_date	date	NO	PRI	NULL	DEFAULT_GENERATED
session_time	datetime	NO		NULL	

7. Components Table

```
CREATE TABLE Components (
    course_ID varchar(6),
    comp_Name varchar(50) NOT NULL,
    comp_per int NOT NULL,
    FOREIGN KEY(course_ID) REFERENCES Course(course_ID),
    PRIMARY KEY(course_ID, comp_Name, comp_per)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
comp_Name	varchar(50)	NO	PRI	NULL	
comp_per	int	NO	PRI	0	

8. Enrolment Table

```
CREATE TABLE Enrollment (
    course_ID varchar(6) NOT NULL,
    std_ID varchar(20) NOT NULL,
    enrol_date date NOT NULL,
    enrol_time datetime NOT NULL,
    FOREIGN KEY(course_ID) REFERENCES Course(course_ID),
    FOREIGN KEY(std_ID) REFERENCES Student(std_ID),
    PRIMARY KEY(course_ID, std_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
std_ID	varchar(20)	NO	PRI	NULL	
enrol_date	date	NO		curdate()	DEFAULT_GENERATED
enrol_time	datetime	NO		curtime()	DEFAULT_GENERATED

9. Attendance Table

```
CREATE TABLE Attendance (
    course_ID varchar(6) NOT NULL,
    session_ID int NOT NULL,
    session_date date NOT NULL,
    faculty_ID varchar(5) NOT NULL,
    student_ID varchar(20) NOT NULL,
    present boolean NOT NULL,
    FOREIGN KEY(course_ID, session_ID, session_date, faculty_ID) REFERENCES Sessions(course_ID, session_ID, session_date, faculty_ID),
    FOREIGN KEY(course_ID, student_ID) REFERENCES Enrollment(course_ID, std_ID),
    PRIMARY KEY(course_ID, session_ID, faculty_ID, student_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
session_ID	int	NO	PRI	NULL	
session_date	date	NO		NULL	
faculty_ID	varchar(5)	NO	PRI	NULL	
student_ID	varchar(20)	NO	PRI	NULL	
present	tinyint(1)	NO		NULL	

10. Assessment Table

```
CREATE TABLE Assessment (
    course_ID varchar(6) NOT NULL,
    comp_Name varchar(50) NOT NULL,
    comp_per int NOT NULL,
    std_ID varchar(20) NOT NULL,
    total_Marks int NOT NULL,
    got_Marks int NOT NULL,
    final_Marks int NOT NULL,
    FOREIGN KEY(course_ID, comp_Name, comp_per) REFERENCES Components(course_ID, comp_Name, comp_per),
    FOREIGN KEY(course_ID, std_ID) REFERENCES Enrollment(course_ID, std_ID),
    PRIMARY KEY(course_ID, comp_Name, std_ID)
);
```

Field	Type	Null	Key	Default	Extra
course_ID	varchar(6)	NO	PRI	NULL	
comp_Name	varchar(50)	NO	PRI	NULL	
comp_per	int	NO		NULL	
std_ID	varchar(20)	NO	PRI	NULL	
total_Marks	int	NO		NULL	
got_Marks	int	NO		NULL	
final_Marks	int	NO		NULL	

3.3 Validations

Storing the valid value into appropriate field is our responsibility, so we need to validate the data which was entered by the user. We have created various validations for the various attributes of the corresponding table.

(Note: To view the queries for data validations open **Tables.sql** file under project folder.)

1. Student Table

```
-- Contact number validation

ALTER TABLE `Student`
ADD CONSTRAINT `stdContactVal`
CHECK (std_contactNo REGEXP '^[1-9][0-9]{9}$');

-- Email address validation

ALTER TABLE `Student`
ADD CONSTRAINT `stdEmailVal`
CHECK (std_emailID REGEXP '@ahduni.edu.in$');

-- Gender validation

ALTER TABLE `Student`
ADD CONSTRAINT `stdGenderVal`
CHECK (std_gender = 'm' OR std_gender = 'M' OR std_gender = 'f' OR std_gender = 'F');

-- Pincode validation

ALTER TABLE `Student`
ADD CONSTRAINT `stdPincodeVal`
CHECK (std_pinCode REGEXP '[0-9]{6}');
```

2. Instructor Table

```
-- Contact number validation

ALTER TABLE Instructor
ADD CONSTRAINT instContactVal
CHECK (inst_contactNo REGEXP '^[1-9][0-9]{9}$');

-- Email number validation

ALTER TABLE Instructor
ADD CONSTRAINT instEmailVal
CHECK (inst_emailID REGEXP '@ahduni.edu.in$');

-- Gender validation

ALTER TABLE Instructor
ADD CONSTRAINT instGenderVal
CHECK (inst_gender = 'm' OR inst_gender = 'M' OR inst_gender = 'f' OR inst_gender = 'F');

-- Pincode validation

ALTER TABLE Instructor
ADD CONSTRAINT instPincodeVal
CHECK (inst_pinCode REGEXP '[0-9]{6}');
```

3. Components Table

```
-- Component Percantage Validation

ALTER TABLE Components
ADD CONSTRAINT comPerVal
CHECK (com_per >= 0 AND com_PER <= 100);
```

4. Assessment Table

```
-- Marks validation

ALTER TABLE Assessment
ADD CONSTRAINT marksVal
CHECK (got_Marks >= 0 AND total_Marks >= 0 AND got_Marks <= total_Marks);
```

3.4 Sample Records

In order to perform the trigger invocation, procedure or function execution first we have inserted some sample records into all tables.

(Note: To view the queries open **Insert.sql** file under project folder.)

1. Department Table

dept_ID	dept_Name
1	Computer Science And Engineering
2	Chemical Engineering
3	Mechenical Engineering
NULL	NULL

2. Student Table

(Since the table is too large, we have taken the screen shots by dividing into two parts)

std_ID	std_Name	std_dept	std_contactNo	std_birthDate	std_age	std_emailID	std_gender
AU2200001	Jainam Shah	1	9825305409	2003-03-28	19	jainam.s5@ahduni.edu.in	M
AU2200002	Devansh Purani	1	8000119988	2003-03-28	19	devansh.p@ahduni.edu.in	M
AU2200003	Aryan Bhavsar	1	7623909369	2003-08-18	18	aryan.b@ahduni.edu.in	M
AU2200004	Daksh Suthar	2	9016464730	2003-06-22	18	daksh.s@ahduni.edu.in	M
AU2200005	Archana Khurana	2	9512924138	2002-05-07	19	archana.k@ahduni.edu.in	F
AU2200006	Samarth Chaudhari	3	9525024165	2001-03-18	21	samarth.c@ahduni.edu.in	M
AU2200007	Dev Dave	3	6521274198	2001-07-01	20	dev.d@ahduni.edu.in	M

Address	std_fatherName	std_motherName
303. Kamdhenu Appartment,Sudama Street, Vik...	Nitesh Shah	Shilpa Shah
B12/A, Devdarshan Tower,Satelite Area,Gujara...	Vipul Purani	Preeti Purani
B/21, Chitrakut Bungalows,Opp. Sai Mandir, Mal...	BhavinKumar Bhavsar	Manisha Bhavsar
46, Somnathpark society,Sargasan,Gujarat,Ga...	Vijaykumar Suthar	Harshaben Suthar
12, Unnati Society,Ajwa Road,Gujarat,Vadodra...	Amrit Khurana	Priya Khurana
204, Vision Complex,Panjarapole Cross Road,G...	Vipul Chaudhari	Radha Chaudhari
129/a, Nutan Saurabh Society,Harni Road,Guja...	Pranay Dave	Nilanjna Dave

3. Instructor Table

inst_ID	inst_fName	inst_midName	inst_lName	inst_dept	inst_contactNo	inst_birthDate	inst_emailID	inst_age	inst_gender	inst_houseName	inst_streetName	inst_city	inst_state	inst_pincode
CHE01	Akhand	Sunil	Rai	2	6618486177	1975-01-01	akhan.rai@ahduni.edu.in	47	M	P/408	Green Street	Noida	Uttar Pradesh	110096
CHE02	Ramesh	Jignesh	Vaghani	2	9216873414	1980-05-04	ramesh.v@ahduni.edu.in	41	M	C/2	Swastik Road	Ahmedabad	Gujarat	382440
CHE03	Grishma	Manoj	Trivedi	2	9392322688	1987-10-17	grishma.t@ahduni.edu.in	34	F	D/3	Juhapura	Ahmedabad	Gujarat	380321
CSE01	Dhaval	Mukund	Patel	1	9761614564	1981-02-25	dhaval.p@ahduni.edu.in	41	M	B/45	Vatva	Ahmedabad	Gujarat	382440
CSE02	Suresh	Jigar	Parikh	1	9988439712	1985-02-07	suresh.p@ahduni.edu.in	37	M	B/101	Ambawadi	Ahmedabad	Gujarat	380015
CSE03	Minal	Viren	Sheth	1	8952433215	1988-11-23	minal.s@ahduni.edu.in	33	F	J/35	Malpur Road	Modasa	Gujarat	383315
MEC01	Sakshi	Manohar	Soni	3	7464156735	1990-09-09	sakshi.s@ahduni.edu.in	31	F	D/103	Naranpura	Ahmedabad	Gujarat	380139
MEC02	Ratna	Nishit	Desai	3	9313872688	1983-06-19	ratna.d@ahduni.edu.in	38	F	A/21	Shyamal	Ahmedabad	Gujarat	380015
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Course Table

course_ID	course_name	course_dept
CHE001	Surface Chemistry	2
CHE002	Chemical Reactions	2
CHE003	Organic Compounds	2
CSE001	Database Management Systems	1
CSE002	Data Structures and Algorithms	1
CSE003	Design and Analysis of Algorithms	1
MEC001	Rigid Body Dynamics	3
MEC002	Thermodynamics	3
MEC003	Modern Physics	3
NULL	NULL	NULL

5. Course / Faculty Table

course_ID	faculty_ID
CHE003	CHE02
CHE002	CHE03
CSE002	CSE01
CSE003	CSE01
CSE002	CSE02
CSE001	CSE03
MEC002	MEC01
MEC003	MEC01
MEC001	MEC02
NULL	NULL

6. Sessions Table

course_ID	session_ID	faculty_ID	session_date	session_time
CSE002	12	CSE01	2022-01-03	2022-05-02 14:30:00
CSE003	1	CSE01	2022-01-04	2022-05-02 09:30:00
CSE003	2	CSE01	2022-01-11	2022-05-02 09:30:00
CSE003	3	CSE01	2022-01-18	2022-05-02 09:30:00
CSE003	4	CSE01	2022-01-25	2022-05-02 09:30:00
CSE003	5	CSE01	2022-02-02	2022-05-02 09:30:00
CSE003	6	CSE01	2022-02-09	2022-05-02 09:30:00
CSE003	7	CSE01	2022-02-16	2022-05-02 09:30:00
MEC001	1	MEC02	2022-02-02	2022-05-02 09:30:00
MEC001	2	MEC02	2022-02-02	2022-05-02 16:00:00
NULL	NULL	NULL	NULL	NULL

7. Components Table

course_ID	comp_Name	comp_per
CSE001	End Semester Exam	30
CSE001	Mid Semester Exam	30
CSE001	Project	10
CSE001	Quiz	30
CSE002	Class Participation / Attendance	10
CSE002	End Semester Exam	25
CSE002	Practical Exam	20
CSE002	Project	25
CSE002	Viva	20
NULL	NULL	NULL

8. Enrolment Table

course_ID	std_ID	enrol_date	enrol_time
CSE001	AU2200001	2022-05-05	2022-05-05 11:27:23
CSE001	AU2200002	2022-05-05	2022-05-05 11:27:23
CSE001	AU2200003	2022-05-05	2022-05-05 11:27:23
CSE001	AU2200004	2022-05-05	2022-05-05 11:27:23
CSE001	AU2200005	2022-05-05	2022-05-05 11:27:23
CSE001	AU2200006	2022-05-05	2022-05-05 11:27:23
CSE002	AU2200001	2022-05-05	2022-05-05 11:29:44
CSE002	AU2200002	2022-05-05	2022-05-05 11:29:44
CSE002	AU2200003	2022-05-05	2022-05-05 11:29:44
NULL	NULL	NULL	NULL

9. Attendance Table

course_ID	session_ID	session_date	faculty_ID	student_ID	present
CSE001	1	2021-12-01	CSE03	AU2200001	1
CSE001	2	2021-12-03	CSE03	AU2200001	0
CSE001	3	2021-12-05	CSE03	AU2200001	1
CSE001	4	2021-12-08	CSE03	AU2200001	1
CSE001	5	2021-12-10	CSE03	AU2200001	0
CSE001	6	2021-12-12	CSE03	AU2200001	1
CSE001	7	2021-12-15	CSE03	AU2200001	1
CSE001	8	2021-12-17	CSE03	AU2200001	1

10. Assessment Table

course_ID	comp_Name	comp_per	std_ID	total_Marks	got_Marks	final_Marks
CSE001	End Semester Exam	30	AU2200001	50	48	29
CSE001	Mid Semester Exam	30	AU2200001	40	35	27
CSE001	Project	10	AU2200001	25	21	9
CSE001	Quiz	30	AU2200001	30	21	21
NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Procedures and their results

1) Procedure to view the all students of the specified department

Input: department ID

Output: students who are in the specified department

```
DROP PROCEDURE IF EXISTS `stdInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE stdInDept(deptID int)
BEGIN
    SELECT Student.std_ID, CONCAT(Student.std_fName, ' ', Student.std_lName) AS std_name
    FROM Student
    INNER JOIN Department
    ON Student.std_dept = deptID AND Student.std_dept = Department.dept_ID;
END $$
DELIMITER ;

CALL stdInDept(1);
```

Output: All Students who are in the Computer Science Department

std_ID	std_name
AU2200001	Jainam Shah
AU2200002	Devansh Purani
AU2200003	Aryan Bhavsar

2) Procedure to view the no of students in each department

Output: no of students in each department

```
DROP PROCEDURE IF EXISTS `noOfStdInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE noOfStdInDept()
BEGIN
    SELECT Department.dept_Name, COUNT(Student.std_ID) AS noOfStudents
    FROM Student
    INNER JOIN Department
    ON Student.std_dept = Department.dept_ID
    GROUP BY Department.dept_Name;
END $$

CALL noOfStdInDept();
```

Output:

	dept_Name	noOfStudents
▶	Computer Science And Engineering	3
	Chemical Engineering	2
	Mechenical Engineering	2

3) Procedure to view the all instructors of the specified department

Input: department ID

Output: instructors who are in the specified department

```
DROP PROCEDURE IF EXISTS `instInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE instInDept(deptId int)
BEGIN
    SELECT Instructor.inst_ID, CONCAT(Instructor.inst_fName, ' ', Instructor.inst_lName) AS inst_name
    FROM Instructor
    INNER JOIN Department
    ON Instructor.inst_dept = deptID AND Instructor.inst_dept = Department.dept_ID;
END $$

CALL instInDept(1);
```

Output:

	inst_ID	inst_name
▶	CSE01	Dhaval Patel
	CSE02	Suresh Parikh
	CSE03	Minal Sheth

4) Procedure to view the total no of instructors in each department

Output: no of instructors in each department

```
DROP PROCEDURE IF EXISTS `noOfInstInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE noOfInstInDept()
BEGIN
    SELECT Department.dept_Name, COUNT(Instructor.inst_ID) AS noOfInstructors
    FROM Instructor
    INNER JOIN Department
    ON Instructor.inst_dept = Department.dept_ID
    GROUP BY Department.dept_Name;
END $$

DELIMITER ;

CALL noOfInstInDept();
```

Output:

	dept_Name	noOfInstructors
▶	Computer Science And Engineering	3
	Chemical Engineering	3
	Mechenical Engineering	2

5) Procedure to view the details of the course in the specified department

Input: department ID

Output: courses in the specified department

```
DROP PROCEDURE IF EXISTS `courseInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE `courseInDept`(deptID int)
BEGIN
    SELECT course_ID, course_name
    FROM Course
    INNER JOIN Department
    ON Course.course_dept = deptID AND Course.course_dept = Department.dept_ID;
END $$
DELIMITER ;

CALL courseInDept(1);
```

Output:

	course_ID	course_name
▶	CSE001	Database Management Systems
	CSE002	Data Structures and Algorithms
	CSE003	Design and Analysis of Algorithms

6) Procedure to view the total no of courses in each department

Output: no of courses in each department

```
DROP PROCEDURE IF EXISTS `noOfCourseInDept`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE `noOfCourseInDept`()
BEGIN
    SELECT Department.dept_Name, COUNT(Course.course_ID) AS noOfCourses
    FROM Course
    INNER JOIN Department
    ON Course.course_dept = Department.dept_ID
    GROUP BY Department.dept_Name;
END $$
DELIMITER ;

CALL noOfCourseInDept();
```

Output:

	dept_Name	noOfCourses
▶	Computer Science And Engineering	3
	Chemical Engineering	3
	Mechenical Engineering	3

7) Procedure to view the courses which are taught by the specified faculty

Input: faculty ID

Output: courses which are taught by the specified faculty

```
DROP PROCEDURE IF EXISTS `getCourses`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE getCourses(facID varchar(5))
BEGIN
    SELECT CONCAT(Course.course_ID, ' - ', Course.course_name) AS course
    FROM Course
    INNER JOIN Course_Faculty
    ON Course_Faculty.faculty_ID = facID AND Course.course_ID = Course_Faculty.course_ID;
END $$
DELIMITER ;

CALL getCourses('CSE01');
```

Output:

	course
▶	CSE002 - Data Structures and Algorithms
	CSE003 - Design and Analysis of Algorithms

8) Procedure to view all the faculties of the specified course

Input: course ID

Output: name of the faculties who are taking the specified course

```
DROP PROCEDURE IF EXISTS `getFaculties`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE getFaculties(courseID varchar(6))
BEGIN
    SELECT CONCAT(Instructor.inst_fName, ' ', Instructor.inst_lName) AS facName
    FROM Instructor
    INNER JOIN Course_Faculty
    ON Course_Faculty.course_ID = courseID AND Course_Faculty.faculty_ID = Instructor.inst_ID;
END $$
DELIMITER $$

CALL getFaculties('MEC003');
```

Output:

	facName
▶	Sakshi Soni
	Ratna Desai

9) Procedure to view all the sessions of the course

Input: course ID

Output: sessions of the specified course

```
DROP PROCEDURE IF EXISTS `sessionsOfCrse`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE sessionsOfCrse(courseID varchar(6))
BEGIN
    SELECT CONCAT('Session-', session_ID) AS sessionID, session_date, TIME(session_time) AS startTime FROM Sessions
    WHERE course_ID = courseID;
END $$
DELIMITER ;

CALL sessionsOfCrse('CSE003');
```

Output:

	sessionID	session_date	startTime
▶	Session-1	2022-01-04	09:30:00
	Session-2	2022-01-11	09:30:00
	Session-3	2022-01-18	09:30:00
	Session-4	2022-01-25	09:30:00
	Session-5	2022-02-02	09:30:00
	Session-6	2022-02-09	09:30:00
	Session-7	2022-02-16	09:30:00

10) Procedure to display the schedule of the faculty

Input: faculty ID

Output: schedule of the faculty

```
DROP PROCEDURE IF EXISTS `displayFacultyShedule`;

DELIMITER $$
USE Student_Management_System $$
CREATE PROCEDURE displayFacultyShedule(facID varchar(5))
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE courseID varchar(6);
    DECLARE courseName varchar(100);
    DECLARE sessionDate date;
    DECLARE sessionTime datetime;
    DECLARE sessionDay int;
    DECLARE dayName varchar(20);

    DECLARE allSessions CURSOR FOR (
        SELECT Sessions.course_ID, Course.course_name, Sessions.session_date, TIME(Sessions.session_time) AS session_time, WEEKDAY(Sessions.session_date) AS session_day
        FROM Sessions
        INNER JOIN Course
        ON Sessions.faculty_ID = facID AND Sessions.course_ID = Course.course_ID
        GROUP BY Sessions.session_date
        ORDER BY Sessions.session_date, TIME(Sessions.session_time)
    );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    CREATE TEMPORARY TABLE facultyShedule (
        sessionDay varchar(200),
        course varchar(200),
        sessionTime varchar(100)
    );

    OPEN allSessions;
    read_cursor: LOOP
        FETCH allSessions INTO courseID, courseName, sessionDate, sessionTime, sessionDay;
        IF done THEN
            LEAVE read_cursor;
        END IF;

        CASE sessionDay
            WHEN 1 THEN
                SET dayName = 'Monday';
            WHEN 2 THEN
                SET dayName = 'Tuesday';
            WHEN 3 THEN
                SET dayName = 'Wednesday';
            WHEN 4 THEN
                SET dayName = 'Thursday';
            WHEN 5 THEN
                SET dayName = 'Friday';
            WHEN 6 THEN
                SET dayName = 'Saturday';
            ELSE
                ITERATE read_cursor;
        END CASE;

        INSERT INTO facultyShedule
        VALUES(CONCAT(dayName ,' - ', sessionDate), CONCAT(courseID, ' - ', courseName), CONCAT(LEFT(TIME(sessionTime), 5), ' - ', LEFT(DATE_ADD(TIME(sessionTime), INTERVA
    END LOOP;

    SELECT * FROM facultyShedule;
END $$
DELIMITER ;

CALL displayFacultyShedule('CSE01');
```

Output:

sessionDay	course	sessionTime
Thursday - 2021-12-03	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Friday - 2021-12-04	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Wednesday - 2021-12-09	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Thursday - 2021-12-10	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Tuesday - 2021-12-15	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Wednesday - 2021-12-16	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Monday - 2021-12-21	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Tuesday - 2021-12-22	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Monday - 2021-12-28	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Saturday - 2022-01-02	CSE002 - Data Structures and Algorithms	14:30 - 16:00
Monday - 2022-01-04	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Monday - 2022-01-11	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Monday - 2022-01-18	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Monday - 2022-01-25	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Tuesday - 2022-02-02	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Tuesday - 2022-02-09	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00
Tuesday - 2022-02-16	CSE003 - Design and Analysis of Algorit...	09:30 - 11:00

5. Functions and their results

1) Function to count total number of departments

Output: number of departments

```
DROP FUNCTION IF EXISTS `cntDepts`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntDepts() RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(dept_ID) AS UNSIGNED) FROM Department);
END $$
DELIMITER ;

SELECT CONCAT('No of departments: ', cntDepts());
```

Output:

No of departments: 3

2) Function to count total number of students

Output: number of students

```
DROP FUNCTION IF EXISTS `cntStds`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntStds() RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(std_ID) AS UNSIGNED) FROM Student);
END $$
DELIMITER ;

SELECT CONCAT('No of students: ', cntStds());
```

Output:

```
No of students: 8
```

3) Function to count total number of instructors

Output: number of instructors

```
DROP FUNCTION IF EXISTS `cntInst`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntInst() RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(inst_ID) AS UNSIGNED) FROM Instructor);
END $$
DELIMITER ;

SELECT CONCAT('No of Instructors: ', cntInst());
```

Output:

```
No of Instructors: 8
```

4) Function to count total number of courses

Output: number of courses

```
DROP FUNCTION IF EXISTS `cntCourse`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntCourse() RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(course_ID) AS UNSIGNED) FROM Course);
END $$
DELIMITER ;

SELECT CONCAT('No of Courses: ', cntCourse());
```

Output:

```
No of Courses: 9
```

5) Function to count total number of sessions in given course

Input: course ID

Output: number of sessions

```
DROP FUNCTION IF EXISTS `cntSessions`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntSessions(courseID varchar(6)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(session_ID) AS UNSIGNED) FROM Sessions WHERE course_ID = courseID GROUP BY course_ID);
END $$
DELIMITER ;

SELECT CONCAT('No of sessions in course ', 'CSE0001-', (SELECT course_name FROM Course WHERE course_ID = 'CSE001'), ': ', cntSessions('CSE001'));
```

Output:

```
No of sessions in course CSE0001-Database Management Systems: 12
```

6) Function to count total number of enrolments in the given course

Input: course ID

Output: number of enrolments in that course

```
DROP FUNCTION IF EXISTS `cntEnrollments`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntEnrollments(courseID varchar(6)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (SELECT CAST(COUNT(std_ID) AS UNSIGNED) FROM Enrollment WHERE course_ID = courseID GROUP BY course_ID);
END $$
DELIMITER ;

SELECT CONCAT('No of enrolments in course ', 'CSE0001-', (SELECT course_name FROM Course WHERE course_ID = 'CSE001'), ': ', cntEnrollments('CSE001'));
```

Output:

```
No of enrolments in course CSE0001-Database Management Systems: 6
```

7) Function to count total number of attended sessions of the given course by the given student

Input: student ID, course ID

Output: number of sessions of the specified course attended by the specified student

```
DROP FUNCTION IF EXISTS `cntAtdSesOfCourse`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntAtdSesOfCourse(stdID varchar(9), courseID varchar(6)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (
        SELECT CAST(COUNT(session_ID) AS UNSIGNED) FROM Attendance
        WHERE student_ID = stdID AND course_ID = courseID AND present = 1
    );
END $$
DELIMITER ;

SELECT CONCAT('No of sessions of course ', 'CSE0001-', (SELECT course_name FROM Course WHERE course_ID = 'CSE001'), 'attended by ', (SELECT std_name FROM Student WHERE std
```

Output:

```
No of sessions of course CSE001-Database Management Systems attended by Jainam Shah : 10
```

8) Function to calculate the attendance of the given course by the given student

Input: student ID, course ID

Output: attendance of the given student in given course

```
DROP FUNCTION IF EXISTS `cntAtdOfCourse`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntAtdOfCourse(stdID varchar(9), courseID varchar(6)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    DECLARE totalSessions int;
    DECLARE sessionsAttended int;

    SET totalSessions = cntSessions(courseID);
    SET sessionsAttended = cntAtdSesOfCourse(stdID, courseID);

    RETURN CEIL((sessionsAttended * 100) / totalSessions);
END $$
DELIMITER ;

SELECT CONCAT('Attendance of ', (SELECT CONCAT(std_fName, ' ', std_lName) FROM Student WHERE std_ID = 'AU2200001'), ' in CSE001-', (SELECT course_name FROM Course WHERE co
```

Output:

```
Attendance of Jainam Shah in CSE001-Database Management Systems is: 84%
```

9) Function to calculate the total number of sessions attended by the given student

Input: student ID

Output: no of sessions attended by the student

```
DROP FUNCTION IF EXISTS `cntOverallAtdSesOfStd`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntOverallAtdSesOfStd(stdID varchar(9)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    RETURN (
        SELECT CAST(COUNT(session_ID) AS UNSIGNED) FROM Attendance
        WHERE student_ID = stdID AND present = 1
    );
END $$
DELIMITER ;

SELECT CONCAT('No of sessions attended by ', (SELECT CONCAT(std_fName, ' ', std_lName) AS name FROM Student WHERE std_ID = 'AU2200001'), ' : ', cntOverallAtdSesOfStd('AU2
```

Output:

```
No of sessions attended by Jainam Shah : 20
```

10) Function to calculate the overall attendance of the given student

Input: student ID

Output: overall attendance of the given student

```
DROP FUNCTION IF EXISTS `cntOverallAtdOfStd`;

DELIMITER $$
USE Student_Management_System $$
CREATE FUNCTION cntOverallAtdOfStd(stdID varchar(9)) RETURNS int
READS SQL DATA
DETERMINISTIC
BEGIN
    DECLARE totalSessions int;
    DECLARE sessionsAttended int;

    SET totalSessions = (SELECT CAST(COUNT(session_ID) AS UNSIGNED) FROM Attendance WHERE student_ID = stdID);
    SET sessionsAttended = cntOverallAtdSesOfStd(stdID);

    RETURN CEIL((sessionsAttended * 100) / totalSessions);
END $$
DELIMITER ;

SELECT CONCAT('Attendance of ', (SELECT CONCAT(std_fName, ' ', std_lName) FROM Student WHERE std_ID = 'AU2200001'), ' is: ', cntOverallAtdOfStd('AU2200001'), '%');
```

Output:

```
Attendance of Jainam Shah is: 84%
```

6. Triggers and their results

(Note: To view the queries for creating triggers open **Triggers.sql** file under project folder.)

1) Trigger to generate enrolment no of student

```
DROP TRIGGER IF EXISTS `generateStdID`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER `generateStdID` BEFORE INSERT
ON Student FOR EACH ROW
BEGIN
    DECLARE newID varchar(20);
    SET @newID = (SELECT CONCAT('AU', RIGHT(YEAR(CURRENT_DATE()), 2), RIGHT(CONCAT('00000', ((SELECT MAX(CAST(RIGHT(std_id, 5) AS UNSIGNED)) FROM Student) + 1)), 5)));

    SET NEW.std_ID = @newID;
END $$
DELIMITER ;

SELECT std_ID, CONCAT(std_fName, ' ', std_lName) AS std_name
FROM Student;
```

It will generate the unique student ID every time and its format is ‘AU-YY-00000’.

Output:

std_ID	std_name
AU2200001	Jainam Shah
AU2200002	Devansh Purani
AU2200003	Aryan Bhavsar

2) Trigger to automatically insert the age of the student based on his / her birth date

```
DROP TRIGGER IF EXISTS `calcStdAge`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER `calcStdAge` BEFORE INSERT
ON Student FOR EACH ROW
BEGIN
    SET NEW.std_age = TIMESTAMPDIFF(YEAR, NEW.std_birthDate, CURRENT_DATE());
END $$
DELIMITER ;

SELECT std_ID, CONCAT(std_fName, ' ', std_lName) AS std_name, std_birthDate, std_age
FROM Student;
```

Output:

std_ID	std_name	std_birthDate	std_age
AU2200001	Jainam Shah	2003-03-28	19
AU2200002	Devansh Purani	2003-03-28	19
AU2200003	Aryan Bhavsar	2003-08-18	18

3) Trigger to automatically generate unique faculty ID based on his / her department

```
DROP TRIGGER IF EXISTS `generateInstID`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER `generateInstID` BEFORE INSERT
ON Instructor FOR EACH ROW
BEGIN
    DECLARE newID varchar(20);
    DECLARE deptCode varchar(3);
    DECLARE dept int;
    DECLARE lastID int;

    SET dept = NEW.inst_dept;
    SET lastID = (SELECT MAX(CAST(RIGHT(inst_ID, 2) AS UNSIGNED)) FROM (SELECT inst_ID FROM Instructor WHERE inst_dept = NEW.inst_dept) AS allInst);

    IF lastID IS NULL THEN
        SET lastID = 0;
    END IF;

    IF dept = 1 THEN
        SET deptCode = 'CSE';
    ELSEIF dept = 2 THEN
        SET deptCode = 'CHE';
    ELSEIF dept = 3 THEN
        SET deptCode = 'MEC';
    END IF;

    SET newID = CONCAT(deptCode, RIGHT(CONCAT('00', (lastID + 1)), 2));
    SET NEW.inst_ID = newID;
END $$
DELIMITER ;

SELECT inst_ID, CONCAT(inst_fName, ' ', inst_lName) AS name FROM Instructor;
```

Output:

inst_ID	name
CHE01	Akhand Rai
CHE02	Ramesh Vaghani
CHE03	Grishma Trivedi
CSE01	Dhaval Patel
CSE02	Suresh Parikh
CSE03	Minal Sheth
MEC01	Sakshi Soni
MEC02	Ratna Desai

4) Trigger to compute the age of the faculty based on his / her birthdate

```
DROP TRIGGER IF EXISTS `calcInstAge`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER `calcInstAge` BEFORE INSERT
ON Instructor FOR EACH ROW
BEGIN
    SET NEW.inst_age = TIMESTAMPDIFF(YEAR, NEW.inst_birthDate, CURRENT_DATE());
END $$
DELIMITER ;

SELECT inst_ID, CONCAT(inst_fName, ' ', inst_lName) AS inst_name, inst_birthDate, inst_age
FROM Instructor;
```

Output:

inst_ID	inst_name	inst_birthDate	inst_age
CHE01	Akhand Rai	1975-01-01	47
CHE02	Ramesh Vaghani	1980-05-04	41
CHE03	Grishma Trivedi	1987-10-17	34
CSE01	Dhaval Patel	1981-02-25	41
CSE02	Suresh Parikh	1985-02-07	37
CSE03	Minal Sheth	1988-11-23	33
MEC01	Sakshi Soni	1990-09-09	31
MEC02	Ratna Desai	1983-06-19	38

5) Trigger to generate the unique course ID based on the course department

```
DROP TRIGGER IF EXISTS `generateCourseID`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER `generateCourseID` BEFORE INSERT
ON Course FOR EACH ROW
BEGIN
    DECLARE newID varchar(20);
    DECLARE deptCode varchar(3);
    DECLARE dept int;
    DECLARE lastID varchar(6);

    SET dept = NEW.course_dept;
    SET lastID = (SELECT MAX(RIGHT(course_ID, 3)) FROM (SELECT course_ID FROM Course WHERE course_dept = dept) AS allIDs);

    IF lastID IS NULL THEN
        SET lastID = 0;
    END IF;

    IF dept = 1 THEN
        SET deptCode = 'CSE';
    ELSEIF dept = 2 THEN
        SET deptCode = 'CHE';
    ELSEIF dept = 3 THEN
        SET deptCode = 'MEC';
    END IF;

    SET newID = CONCAT(deptCode, RIGHT(CONCAT('000', (lastID + 1)), 3));
    SET NEW.course_ID = newID;
END $$
DELIMITER ;
```

Output:

course_ID	course_name
CHE001	Surface Chemistry
CHE002	Chemical Reactions
CHE003	Organic Compounds
CSE001	Database Management Systems
CSE002	Data Structures and Algorithms
CSE003	Design and Analysis of Algorithms
MEC001	Rigid Body Dynamics
MEC002	Thermodynamics
MEC003	Modern Physics
NULL	NULL

6) Trigger to restrict the admin by assigning more than two courses to the faculty

```
DROP TRIGGER IF EXISTS `assignValidation`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER assignValidation BEFORE INSERT
ON Course_Faculty FOR EACH ROW
BEGIN
    DECLARE count int;
    DECLARE msg varchar(300);

    SET count = (SELECT CAST(COUNT(course_ID) AS UNSIGNED) FROM Course_Faculty WHERE faculty_ID = NEW.faculty_ID GROUP BY faculty_ID);

    IF (count = 2) THEN
        SET msg = "Can't assign the faculty to this course as faculty is already assigned in the two courses";
        SIGNAL SQLSTATE '45000' SET message_text = msg;
    END IF;
END $$
DELIMITER ;

INSERT INTO Course_Faculty
VALUES('MEC002', 'MEC02');
```

As the faculty ‘MEC02’ is already teaching two courses the SQL will generate the error which we have created.

Output:

```
90  13:07:56  INSERT INTO Course_Faculty VALUES('MEC002', 'MEC02')   Error Code: 1644. Can't assign the faculty to this course as faculty is already assigned in the two courses
```

7) Trigger to check whether we can plan the session at the specified date and time according to the availability of the faculty

```
DROP TRIGGER IF EXISTS `isFacultyFree`;

DELIMITER $$
CREATE TRIGGER isFacultyFree BEFORE INSERT
ON Sessions FOR EACH ROW
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE sessionDate date;
    DECLARE sessionTime datetime;
    DECLARE msg varchar(300);

    DECLARE sessions CURSOR FOR (
        SELECT session_date, session_time FROM Sessions WHERE faculty_ID = NEW.faculty_ID
    );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN sessions;
    read_cursor: LOOP
        FETCH sessions INTO sessionDate, sessionTime;
        IF done THEN
            LEAVE read_cursor;
        END IF;
        IF (sessionDate = NEW.session_date) THEN
            IF (NEW.session_time BETWEEN sessionTime AND DATE_ADD(sessionTime, INTERVAL 90 MINUTE)) THEN
                SET msg = CONCAT("Can't able to plan the session on ", NEW.session_date, " at ", TIME(NEW.session_time), " as faculty has another session");
                SIGNAL SQLSTATE '45000' SET message_text = msg;
            END IF;
        END IF;
    END LOOP;
    CLOSE sessions;
END $$
DELIMITER ;

INSERT INTO Sessions(course_ID, faculty_ID, session_date, session_time)
VALUES('CSE003', 'CSE01', '2022-01-03', STR_TO_DATE('15:00:00', '%H:%i:%s'));
```

As the faculty ‘CSE01’ is already busy till 16:00 we can’t schedule the session of this faculty on 03-01-2022 at 15:00. SQL will display the error which we have created.

Output:

#	Time	Action	Message
133	19:45:31	CREATE TRIGGER isFacultyFree BEFORE INSERT ON Sessions FOR EACH ROW BEGIN DECLARE done INT DEFAULT FAL...	0 row(s) affected
134	19:45:42	INSERT INTO Sessions(course_ID, faculty_ID, session_date, session_time) VALUES('CSE003', 'CSE01', '2022-01-03', STR_TO_DA...	Error Code: 1644. Can't able to plan the session on 2022-01-03 at 15:00:00 as faculty has another session

8) Trigger to check whether the components are planned well or not

```
DROP TRIGGER IF EXISTS `checkComp`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER checkComp BEFORE INSERT
ON Components FOR EACH ROW
BEGIN
    DECLARE compTotal int;
    DECLARE msg varchar(200);

    SET compTotal = (SELECT CAST(SUM(comp_per) AS UNSIGNED) FROM Components WHERE course_ID = NEW.course_ID GROUP BY course_ID);

    IF ((NEW.comp_per + compTotal) > 100) THEN
        SET msg = 'The total of all components for this course should be equal to 100';
        SIGNAL SQLSTATE '45000' SET message_text = msg;
    END IF;
END $$
DELIMITER ;

INSERT INTO Components
VALUES('CSE001', 'Attendance', 10);
```

Output:

#	Time	Action	Message
11	11:11:43	INSERT INTO Components VALUES('CSE001', 'End Semester Exam', 30)	1 row(s) affected
12	11:13:10	INSERT INTO Components VALUES('CSE001', 'Attendance', 10)	Error Code: 1644. The total of all components for this course should be equal to 100

9) Trigger to calculate final marks based on the weightage of the component

```
DROP TRIGGER IF EXISTS `calcFinalMarks`;

DELIMITER $$
USE Student_Management_System $$
CREATE TRIGGER calcFinalMarks BEFORE INSERT
ON Assessment FOR EACH ROW
BEGIN
    SET NEW.final_marks = CEIL((NEW.comp_per * NEW.got_marks) / NEW.total_marks);
END $$
DELIMITER ;
```

Output:

course_ID	comp_Name	comp_per	std_ID	total_Marks	got_Marks	final_Marks
CSE001	End Semester Exam	30	AU2200001	50	48	29
CSE001	Mid Semester Exam	30	AU2200001	40	35	27
CSE001	Project	10	AU2200001	25	21	9
CSE001	Quiz	30	AU2200001	30	21	21
NULL	NULL	NULL	NULL	NULL	NULL	NULL

7. Contribution of Each Member

Enrolment Number	Name	Contribution
AU2040238	Jainam Shah	Planning Report Writing, Schema Design, Code of whole database
AU2040243	Devansh Purani	Planning Report Writing, ER Diagram, Generated Sample Data
AU2040244	Aryan Bhavsar	Report Writing, ER Diagram, Generated Sample Data
AU2040167	Daksh Suthar	Report Writing