CREATE TABLE TeacherRegKey (ID INT PRIMARY KEY AUTO\_INCREMENT, regKey VARCHAR(250) NOT NULL);

CREATE TABLE Admin(ID INT PRIMARY KEY AUTO\_INCREMENT, username VARCHAR(15) NOT NULL, password VARCHAR(25) NOT NULL, superAuthKey VARCHAR(450) NOT NULL, firstName VARCHAR(20) NOT NULL, lastName VARCHAR(20) NOT NULL, sex VARCHAR(1) NOT NULL, birthDate DATE NOT NULL, phoneNumber BIGINT NOT NULL, connected VARCHAR(3) DEFAULT 'off', email VARCHAR(30) NOT NULL);

CREATE TABLE StudentRegKEY (ID INT PRIMARY KEY AUTO\_INCREMENT, classID INT, regKey VARCHAR(250) NOT NULL);

CREATE TABLE teacher(ID INT PRIMARY KEY AUTO\_INCREMENT, username VARCHAR(15) NOT NULL, password VARCHAR(25) NOT NULL, email VARCHAR(30) NOT NULL, phoneNumber BIGINT NOT NULL, authkey VARCHAR (20) NOT NULL, sex VARCHAR(1) NOT NULL, birthDate DATE NOT NULL, lastName VARCHAR(20) NOT NULL, firstName VARCHAR(20) NOT NULL, connected VARCHAR(3) NOT NULL DEFAULT 'off');

CREATE TABLE student(ID INT PRIMARY KEY AUTO\_INCREMENT, classID INT, email VARCHAR(30) NOT NULL, phoneNumber BIGINT NOT NULL, studyingYear INT(1) NOT NULL, username VARCHAR(15) NOT NULL, password VARCHAR(25) NOT NULL, authkey VARCHAR(20) NOT NULL, sex VARCHAR(1) NOT NULL, birthDate DATE NOT NULL, lastName VARCHAR(20) NOT NULL, firstName VARCHAR(20) NOT NULL, connected VARCHAR(3) NOT NULL DEFAULT 'off');

CREATE TABLE Question(ID INT PRIMARY KEY AUTO\_INCREMENT, studentID INT, teacherID INT, title VARCHAR(170) NOT NULL, description VARCHAR(1000) NOT NULL, priority VARCHAR(6), date DATETIME DEFAULT CURRENT\_TIMESTAMP, state VARCHAR(12) NOT NULL DEFAULT 'not answered' , answer VARCHAR(1000) DEFAULT ‘The teacher has not respond to your question yet');

CREATE TABLE Class(ID INT PRIMARY KEY AUTO\_INCREMENT, teacherID INT, year INT(1), location VARCHAR(50) NOT NULL, available VARCHAR(1) NOT NULL, capacity INT(3) NOT NULL);

CREATE TABLE Assignment (ID INT PRIMARY KEY AUTO\_INCREMENT, teacherID INT, description VARCHAR(1000) NOT NULL, deadline DATETIME NOT NULL, title VARCHAR(30) NOT NULL, state VARCHAR(10) NOT NULL DEFAULT 'OPEN', minGrade INT(2) NOT NULL);

CREATE TABLE Course(ID INT PRIMARY KEY AUTO\_INCREMENT, teacherID INT, name VARCHAR(50) NOT NULL, description VARCHAR(1000) NOT NULL );

CREATE TABLE ClassCourse( courseID INT, classID INT);

CREATE TABLE ClassAssignment(classID INT, assignmentID INT);

CREATE TABLE StudentAssignmentSolution(

studentID INT,

assignmentID INT,

solution VARCHAR(1000),

state VARCHAR(30) DEFAULT 'Not solved',

solutionDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

grade INT(2) DEFAULT 0

);

CREATE TABLE Grade(

id INT PRIMARY KEY AUTO\_INCREMENT,

courseID INT,

assignmentID INT,

studentID INT,

assignmentTitle VARCHAR(30),

grade INT(2)

);

CREATE TRIGGER newClassAssignment AFTER INSERT ON classassignment FOR EACH ROW INSERT INTO StudentAssignmentSolution(studentID, assignmentID)

SELECT s.ID, NEW.assignmentID

FROM Student s

WHERE s.classID = NEW.classID;

CREATE TRIGGER StudentClassAssignmentCourseOnDelete AFTER DELETE

ON classcourse FOR EACH ROW

Begin

DELETE FROM classassignment

WHERE assignmentID in

(select id from assignment where teacherID =

(select teacherID from course where id = old.courseID))

&& classID = old.classID;

Delete FROM studentassignmentsolution where studentID in

(select id from student where classid = old.classID)

&& assignmentID = (select id from assignment where teacherID =

(select teacherID from course where id = old.courseID));

End;

CREATE TRIGGER classAssignmentDeleted AFTER DELETE ON

ClassAssignment FOR EACH ROW

DELETE FROM studentassignmentsolution where assignmentID = old.assignmentID;

CREATE TRIGGER StudentQuestionOnClassCourseDelete AFTER DELETE

ON classcourse FOR EACH ROW DELETE FROM QUESTION

WHERE studentID in (select id from student where classid = old.classid)

&& teacherID = (select teacherid from course where id = old.courseID);

CREATE TRIGGER StudentQuestionOnClassCourseUpdate AFTER UPDATE

ON classcourse FOR EACH ROW IF(old.courseID <> new.courseID)

THEN

DELETE FROM QUESTION where studentID in (select id from student where classid = old.classid)

&& teacherID = (select teacherid from course where id = old.courseID);

end if;

CREATE TRIGGER deleteTeacherAssignedQuestionOnCourseDelete AFTER DELETE ON course

FOR EACH ROW

DELETE FROM question WHERE old.teacherID = teacherID;

CREATE TRIGGER deleteStudentQuestionOnStudentDelete

AFTER DELETE ON student FOR EACH ROW

DELETE FROM question WHERE studentID = old.id;

CREATE TRIGGER deleteTeacherAssignedQuestionOnTeacherDelete

AFTER DELETE ON teacher FOR EACH ROW

DELETE FROM question WHERE teacherID = old.id;

CREATE TRIGGER StudentClassAssignmentCourseOnUpdate AFTER UPDATE

ON classcourse FOR EACH ROW IF(old.courseID <> new.courseID) THEN

DELETE FROM classassignment

WHERE assignmentID in

(select id from assignment where teacherID =

(select teacherID from course where id = old.courseID))

&& classID = old.classID;

Delete FROM studentassignmentsolution where studentID in

(select id from student where classid = old.classID)

&& assignmentID = (select id from assignment where teacherID =

(select teacherID from course where id = old.courseID));

End If;

CREATE TRIGGER assignmentClosed after update on assignment for each row

IF (old.state <> new.state) THEN

update studentassignmentsolution set solutionDate = (SELECT DATE\_ADD(old.deadline, INTERVAL 1 DAY)), Grade = old.minGrade, state = 'Graded' where assignmentID = old.id && state = 'Not solved';

END if;

CREATE TRIGGER insertStudentGrade AFTER UPDATE ON studentassignmentsolution FOR EACH ROW

IF(old.grade <> new.grade) THEN INSERT INTO grade(studentID, courseID, grade, assignmentTitle)

VALUES(old.studentID, (SELECT id FROM course WHERE teacherID =

(SELECT teacherID FROM assignment WHERE id = old.assignmentID)), new.grade,

(select title from assignment where id = old.assignmentID));

END IF;

CREATE TRIGGER deleteTeacherQuestionOnCourseUpdate AFTER UPDATE ON course

FOR EACH ROW

IF old.teacherID <> new.teacherID THEN DELETE FROM question WHERE teacherID = old.teacherID;

END IF ;;

CREATE TRIGGER deleteStudentQuestionOnStudentUpdate AFTER UPDATE ON student

FOR EACH ROW

IF old.classID <> new.classID THEN DELETE FROM question WHERE studentID = new.id;

END IF ;;

ALTER TABLE Grade ADD FOREIGN KEY(courseID) REFERENCES course(ID) ON DELETE CASCADE;

ALTER TABLE Grade ADD FOREIGN KEY(studentID) REFERENCES student(ID) ON DELETE CASCADE;

ALTER TABLE Question ADD CONSTRAINT QuestionAskedBy FOREIGN KEY(studentID) REFERENCES student(ID) ON DELETE CASCADE;

ALTER TABLE Question ADD CONSTRAINT QuestionAssignedTo FOREIGN KEY(teacherID) REFERENCES teacher(ID) ON DELETE CASCADE;

ALTER TABLE StudentRegKey ADD CONSTRAINT StudentHasClass FOREIGN KEY(classID) REFERENCES class(ID) ON DELETE CASCADE;

ALTER TABLE Student ADD CONSTRAINT StudentBelongsTo FOREIGN KEY(classID) REFERENCES class(ID) ON DELETE SET NULL;

ALTER TABLE Class ADD CONSTRAINT ClassSupervisedBy FOREIGN KEY(teacherID) REFERENCES teacher(ID) ON DELETE SET NULL;

ALTER TABLE Assignment ADD CONSTRAINT AssignmentAddedBy FOREIGN KEY(teacherID) REFERENCES teacher(ID) ON DELETE CASCADE;

ALTER TABLE Course ADD CONSTRAINT CourseTeachedBy FOREIGN KEY(teacherID) REFERENCES teacher(ID) ON DELETE CASCADE;

ALTER TABLE ClassCourse ADD CONSTRAINT ClassHasCourse FOREIGN KEY(courseID) REFERENCES course(ID) ON DELETE CASCADE;

ALTER TABLE ClassCourse ADD CONSTRAINT CourseAssignedTo FOREIGN KEY(classID) REFERENCES class(ID) ON DELETE CASCADE;

ALTER TABLE ClassAssignment ADD CONSTRAINT ClassHasAssignment FOREIGN KEY(assignmentID) REFERENCES assignment(ID) ON DELETE CASCADE;

ALTER TABLE ClassAssignment ADD CONSTRAINT ClassAssignmentAssignedTo FOREIGN KEY(classID) REFERENCES class(ID) ON DELETE CASCADE;

ALTER TABLE StudentAssignmentSolution ADD CONSTRAINT AssignmentSolutionStudent

FOREIGN KEY(studentID) references student(ID) ON DELETE CASCADE;

ALTER TABLE StudentAssignmentSolution ADD CONSTRAINT AssignmentSolution

FOREIGN KEY(assignmentID) references assignment(ID) ON DELETE CASCADE;

ALTER TABLE Student ADD CONSTRAINT UNIQUE(username);

ALTER TABLE Student ADD CONSTRAINT UNIQUE(email);

ALTER TABLE Teacher ADD CONSTRAINT UNIQUE(username);

ALTER TABLE Teacher ADD CONSTRAINT UNIQUE(email);

ALTER TABLE TeacherRegKey ADD CONSTRAINT UNIQUE(regKey);

ALTER TABLE StudentRegKey ADD CONSTRAINT UNIQUE(regKey);

ALTER TABLE ClassCourse ADD CONSTRAINT UNIQUE(courseID, classID);

ALTER TABLE ClassAssignment ADD CONSTRAINT UNIQUE(classID, assignmentID);