**Report for Database Project**

**Morgan K. Thornton (Student ID: 03004063)**

**Jawad Adall**

**CSCI 432-01 Database Systems**

1. The ER diagram (with the attributes and foreign keys/primary keys indicated);

A screenshot of a computer

Description automatically generated

1. Write the commands for creating tables and inserting values;

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

CREATE DATABASE IF NOT EXISTS `grade\_book` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

USE `grade\_book`;

DROP TABLE IF EXISTS `course`;

CREATE TABLE `course` (

`c\_id` int(11) NOT NULL,

`lec\_id` int(11) DEFAULT NULL,

`c\_name` varchar(255) DEFAULT NULL,

`c\_year` int(11) DEFAULT NULL,

`c\_sem` varchar(255) DEFAULT NULL,

`c\_weeks` int(11) DEFAULT NULL,

`c\_participation\_pts` int(11) DEFAULT NULL COMMENT 'Percentage of grade for participation',

`c\_assignment\_pts` int(11) DEFAULT NULL COMMENT 'Percentage of grade for assignments',

`c\_project\_pts` int(11) DEFAULT NULL COMMENT 'Percentage of grade for projects',

`c\_exam\_pts` int(11) DEFAULT NULL COMMENT 'Percentage of grade for exams',

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TRIGGER IF EXISTS `total\_check\_before\_insert`;

DELIMITER $$

CREATE TRIGGER `total\_check\_before\_insert` BEFORE INSERT ON `course` FOR EACH ROW BEGIN

DECLARE total\_marks integer;

DECLARE participation integer;

DECLARE assignment integer;

DECLARE project integer;

DECLARE exam integer;

SET participation = NEW.c\_participation\_pts;

SET assignment = NEW.c\_assignment\_pts;

SET project = NEW.c\_project\_pts;

SET exam = NEW.c\_exam\_pts;

SET total\_marks = participation + assignment + project + exam;

IF total\_marks < 100 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'All categories need to add upto exactly 100', MYSQL\_ERRNO = 1001;

ELSEIF total\_marks > 100 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'All categories need to add upto exactly 100', MYSQL\_ERRNO = 1001;

END IF;

END

$$

DELIMITER ;

DROP TABLE IF EXISTS `course\_coursework`;

CREATE TABLE `course\_coursework` (

`cw\_id` int(11) NOT NULL,

`cw\_c\_id` int(11) DEFAULT NULL,

`cw\_course\_type` enum('assignment','project','exam') DEFAULT NULL COMMENT 'Allowed types: project, assignment, exam',

`cw\_content` text DEFAULT NULL COMMENT 'course work content',

`cw\_release\_date` date DEFAULT NULL,

`cw\_due\_date` date DEFAULT NULL,

`cw\_total` int(11) DEFAULT 100,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TABLE IF EXISTS `course\_participation`;

CREATE TABLE `course\_participation` (

`cp\_id` int(11) NOT NULL,

`cp\_stc\_id` int(11) DEFAULT NULL,

`cp\_week` int(11) DEFAULT NULL,

`cp\_present` tinyint(1) DEFAULT NULL,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TRIGGER IF EXISTS `ensure\_participation\_not\_exceeded`;

DELIMITER $$

CREATE TRIGGER `ensure\_participation\_not\_exceeded` BEFORE INSERT ON `course\_participation` FOR EACH ROW BEGIN

DECLARE total\_participation integer;

DECLARE total\_weeks integer;

SET total\_weeks = (SELECT course.c\_weeks FROM student\_courses

JOIN course ON student\_courses.c\_id = course.c\_id WHERE student\_courses.stc\_id = NEW.cp\_stc\_id);

SET total\_participation = (SELECT COUNT(cp\_id) FROM course\_participation WHERE cp\_stc\_id = NEW.cp\_stc\_id);

IF total\_participation >= total\_weeks THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'You have reached the maximum number of attendances for this course';

END IF;

END

$$

DELIMITER ;

DROP TABLE IF EXISTS `lecturer`;

CREATE TABLE `lecturer` (

`lec\_id` int(11) NOT NULL,

`lec\_name` varchar(255) DEFAULT NULL,

`lec\_gender` enum('male','female','other') DEFAULT NULL,

`lec\_email` varchar(255) DEFAULT NULL,

`lec\_phone` varchar(255) DEFAULT NULL,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TABLE IF EXISTS `student`;

CREATE TABLE `student` (

`st\_id` int(11) NOT NULL,

`st\_name` varchar(255) DEFAULT NULL,

`st\_email` varchar(255) DEFAULT NULL,

`st\_phone` varchar(255) DEFAULT NULL,

`st\_gender` enum('male','female','other') DEFAULT NULL,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TABLE IF EXISTS `student\_courses`;

CREATE TABLE `student\_courses` (

`stc\_id` int(11) NOT NULL,

`st\_id` int(11) DEFAULT NULL,

`c\_id` int(11) DEFAULT NULL,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

DROP TRIGGER IF EXISTS `ensure\_no\_duplicate\_student\_course`;

DELIMITER $$

CREATE TRIGGER `ensure\_no\_duplicate\_student\_course` BEFORE INSERT ON `student\_courses` FOR EACH ROW BEGIN

DECLARE course\_count integer;

SET course\_count = (SELECT COUNT(stc\_id) FROM student\_courses WHERE st\_id = NEW.st\_id AND c\_id = NEW.c\_id);

IF course\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'A student can enroll to a course once';

END IF;

END

$$

DELIMITER ;

DROP TABLE IF EXISTS `student\_course\_coursework`;

CREATE TABLE `student\_course\_coursework` (

`cw\_id` int(11) NOT NULL,

`stc\_id` int(11) NOT NULL,

`marks` int(11) DEFAULT NULL,

`created\_at` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

ALTER TABLE `course`

ADD PRIMARY KEY (`c\_id`),

ADD KEY `lec\_id` (`lec\_id`);

ALTER TABLE `course\_coursework`

ADD PRIMARY KEY (`cw\_id`),

ADD KEY `cw\_c\_id` (`cw\_c\_id`);

ALTER TABLE `course\_participation`

ADD PRIMARY KEY (`cp\_id`),

ADD KEY `cp\_stc\_id` (`cp\_stc\_id`);

ALTER TABLE `lecturer`

ADD PRIMARY KEY (`lec\_id`);

ALTER TABLE `student`

ADD PRIMARY KEY (`st\_id`);

ALTER TABLE `student\_courses`

ADD PRIMARY KEY (`stc\_id`),

ADD KEY `st\_id` (`st\_id`),

ADD KEY `c\_id` (`c\_id`);

ALTER TABLE `student\_course\_coursework`

ADD PRIMARY KEY (`cw\_id`,`stc\_id`),

ADD KEY `stc\_id` (`stc\_id`);

ALTER TABLE `course`

MODIFY `c\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `course\_coursework`

MODIFY `cw\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `course\_participation`

MODIFY `cp\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `lecturer`

MODIFY `lec\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `student`

MODIFY `st\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `student\_courses`

MODIFY `stc\_id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `course`

ADD CONSTRAINT `course\_ibfk\_1` FOREIGN KEY (`lec\_id`) REFERENCES `lecturer` (`lec\_id`);

ALTER TABLE `course\_coursework`

ADD CONSTRAINT `course\_coursework\_ibfk\_1` FOREIGN KEY (`cw\_c\_id`) REFERENCES `course` (`c\_id`);

ALTER TABLE `course\_participation`

ADD CONSTRAINT `course\_participation\_ibfk\_1` FOREIGN KEY (`cp\_stc\_id`) REFERENCES `student\_courses` (`stc\_id`);

ALTER TABLE `student\_courses`

ADD CONSTRAINT `student\_courses\_ibfk\_1` FOREIGN KEY (`st\_id`) REFERENCES `student` (`st\_id`),

ADD CONSTRAINT `student\_courses\_ibfk\_2` FOREIGN KEY (`c\_id`) REFERENCES `course` (`c\_id`);

ALTER TABLE `student\_course\_coursework`

ADD CONSTRAINT `student\_course\_coursework\_ibfk\_1` FOREIGN KEY (`cw\_id`) REFERENCES `course\_coursework` (`cw\_id`),

ADD CONSTRAINT `student\_course\_coursework\_ibfk\_2` FOREIGN KEY (`stc\_id`) REFERENCES `student\_courses` (`stc\_id`);

COMMIT;

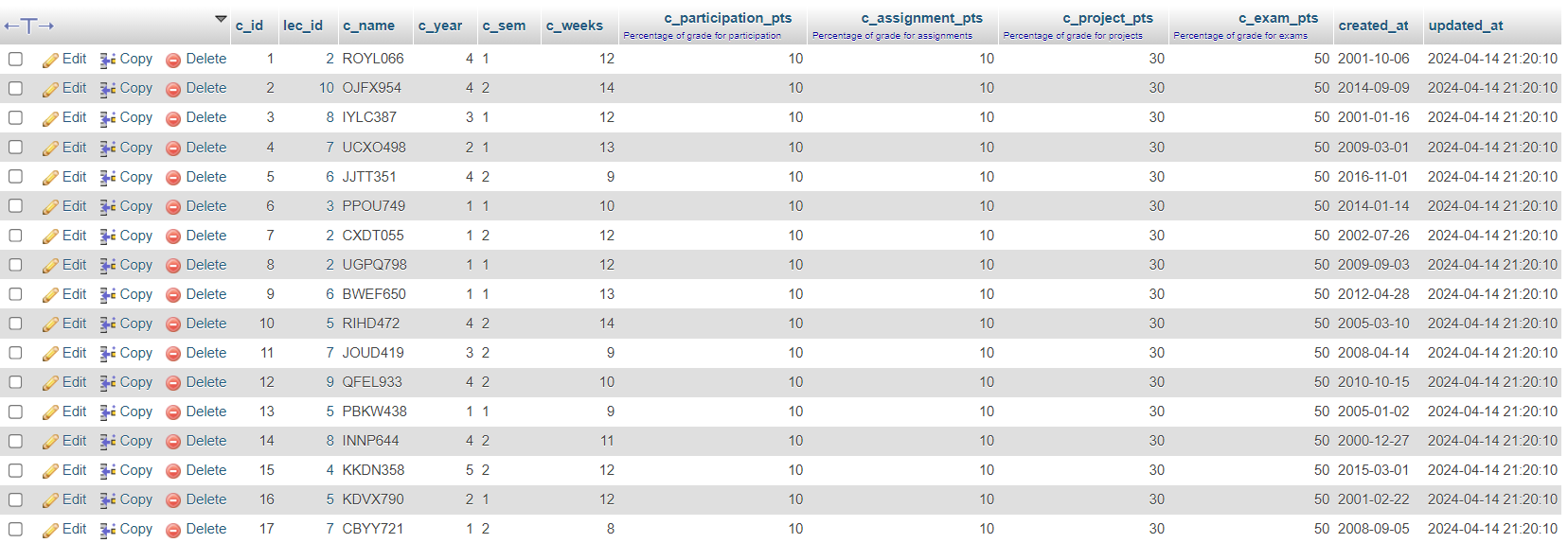
/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

1. Show the tables with the contents that you have inserted;

courses



Course coursework (assignments, projects and exams)



Lecturers

A screenshot of a computer

Description automatically generated

Students

A screenshot of a computer

Description automatically generated

Student courses

A screenshot of a computer

Description automatically generated

Student’s attempted coursework (student\_course\_coursework)

A screenshot of a computer

Description automatically generated

Course\_participation (attendance history of students)

A screenshot of a computer

Description automatically generated

1. Compute the average/highest/lowest score of an assignment;

Code:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `calculateMinMaxAve`(IN `assignment\_id` INT)

BEGIN

DECLARE assignment\_count integer;

DECLARE total\_assignment integer;

DECLARE min\_assignment integer;

DECLARE max\_assignment integer;

DECLARE ave\_assignment integer;

SET assignment\_count = (SELECT COUNT(marks) FROM student\_course\_coursework WHERE cw\_id = assignment\_id);

SET total\_assignment = (SELECT SUM(marks) FROM student\_course\_coursework WHERE cw\_id = assignment\_id);

SET @min\_assignment = (SELECT MIN(marks) FROM student\_course\_coursework WHERE cw\_id = assignment\_id);

SET @max\_assignment = (SELECT MAX(marks) FROM student\_course\_coursework WHERE cw\_id = assignment\_id);

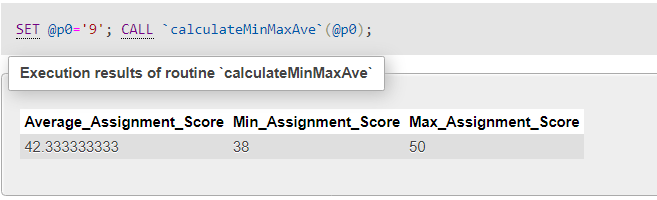
SET @ave\_assignment = total\_assignment/assignment\_count;

SELECT @ave\_assignment AS `Average\_Assignment\_Score`, @min\_assignment AS `Min\_Assignment\_Score`,@max\_assignment AS `Max\_Assignment\_Score`;

END$$

DELIMITER ;

Test:



1. List all of the students in a given course;

Code:

BEGIN

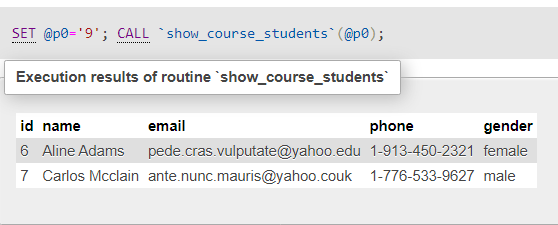
SELECT student\_courses.st\_id AS id, student.st\_name AS name, student.st\_email AS email, @student.st\_phone AS phone, student.st\_gender AS gender FROM student\_courses

INNER JOIN student on student\_courses.st\_id = student.st\_id

WHERE student\_courses.c\_id = course\_id;

END$$

Test:



1. List all of the students in a course and all of their scores on every assignment;

Code:

select student.st\_name, course.c\_name, course\_coursework.cw\_id, course\_coursework.cw\_course\_type, student\_course\_coursework.marks FROM student\_course\_coursework

JOIN course\_coursework ON student\_course\_coursework.cw\_id = course\_coursework.cw\_id

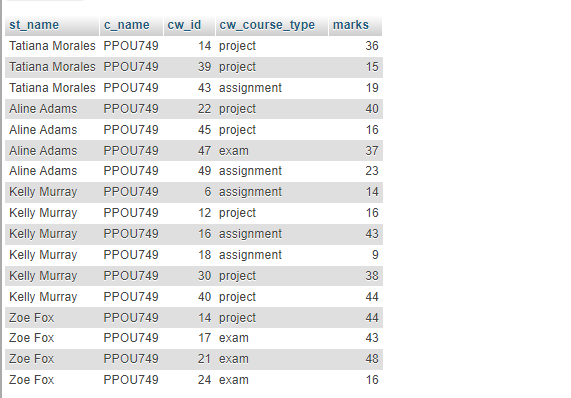
JOIN student\_courses ON student\_course\_coursework.stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

JOIN course ON student\_courses.c\_id = course.c\_id

WHERE course.c\_id = 6 GROUP BY student.st\_name;

Test (course id = 6):



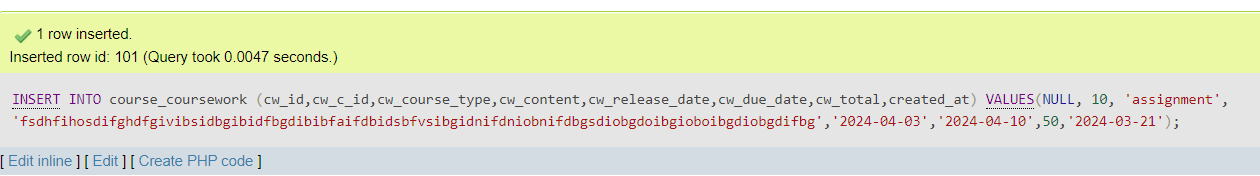
1. Add an assignment to a course;

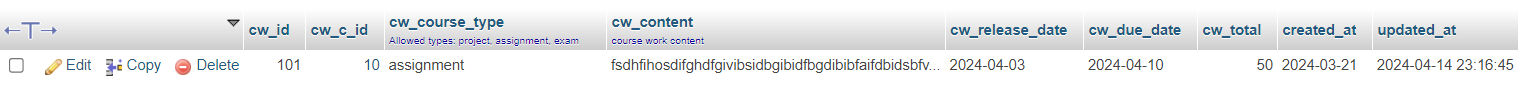
Code:

INSERT INTO course\_coursework (cw\_id,cw\_c\_id,cw\_course\_type,cw\_content,cw\_release\_date,cw\_due\_date,cw\_total,created\_at)

VALUES(NULL, 10, 'assignment', 'fsdhfihosdifghdfgivibsidbgibidfbgdibibfaifdbidsbfvsibgidnifdniobnifdbgsdiobgdoibgioboibgdiobgdifbg','2024-04-03','2024-04-10',50,'2024-03-21');

Test:



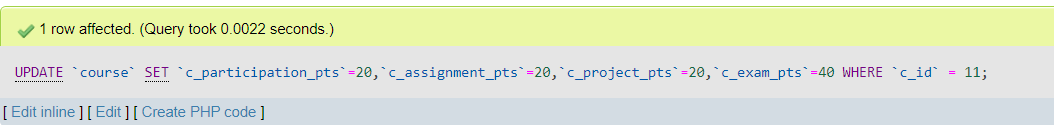


1. Change the percentages of the categories for a course;

Code:

UPDATE `course` SET `c\_participation\_pts`=20,`c\_assignment\_pts`=20,`c\_project\_pts`=20,`c\_exam\_pts`=40 WHERE `c\_id` = 11;

Test:





1. Add 2 points to the score of each student on an assignment;

Code:

BEGIN

DECLARE total\_score integer;

SET total\_score = (SELECT cw\_total FROM course\_coursework WHERE cw\_id = assignment\_id);

UPDATE student\_course\_coursework SET marks = (

CASE WHEN (marks + points) <= total\_score THEN (marks + points)

WHEN (marks + points) > total\_score THEN (marks + (points - ((points + marks)- total\_score)))

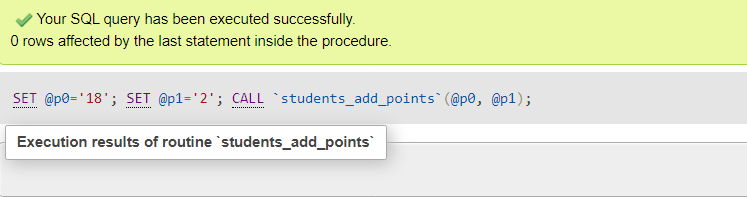
END)

WHERE cw\_id = assignment\_id;

END

Test:





A screenshot of a calendar

Description automatically generated

1. Add 2 points just to those students whose last name contains a ‘Q’.

Code:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `students\_add\_points\_q`(IN `assignment\_id` INT, IN `points` INT)

BEGIN

DECLARE total\_score integer;

SET total\_score = (SELECT cw\_total FROM course\_coursework WHERE cw\_id = assignment\_id);

UPDATE student\_course\_coursework

JOIN student\_courses ON student\_course\_coursework.stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

SET student\_course\_coursework.marks = (

CASE WHEN (student\_course\_coursework.marks + points) <= total\_score THEN (student\_course\_coursework.marks + points)

WHEN (student\_course\_coursework.marks + points) > total\_score THEN (student\_course\_coursework.marks + (points - ((points + student\_course\_coursework.marks)- total\_score)))

ELSE (student\_course\_coursework.marks + points)

END)

WHERE student\_course\_coursework.cw\_id = assignment\_id AND student.st\_name LIKE '%q%';

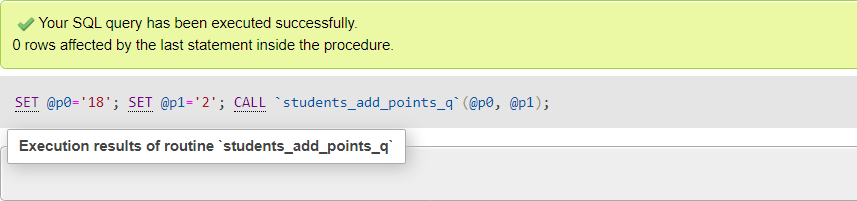
END$$

DELIMITER ;

Test: (No change expected since no student’s name starts with q

A screenshot of a calendar

Description automatically generated



A screenshot of a number

Description automatically generated

1. Compute the grade for a student;

Code:

BEGIN

DECLARE participation\_count integer;

DECLARE participation\_present integer;

DECLARE participation\_g\_total integer;

DECLARE participation\_pct integer;

DECLARE assignment\_pct integer;

DECLARE project\_pct integer;

DECLARE exam\_pct integer;

DECLARE assignment\_grade decimal(10,1);

DECLARE project\_grade decimal(10,1);

DECLARE exam\_grade decimal(10,1);

DECLARE student\_grade decimal(10,1);

SET participation\_count = (SELECT c\_weeks FROM course WHERE c\_id = course\_id);

SET participation\_pct = (SELECT c\_participation\_pts FROM course WHERE c\_id = course\_id);

SET assignment\_pct = (SELECT c\_assignment\_pts FROM course WHERE c\_id = course\_id);

SET project\_pct = (SELECT c\_project\_pts FROM course WHERE c\_id = course\_id);

SET exam\_pct = (SELECT c\_exam\_pts FROM course WHERE c\_id = course\_id);

SET participation\_present = (SELECT COUNT(cp\_id) FROM course\_participation

JOIN student\_courses ON course\_participation.cp\_stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

JOIN course ON student\_courses.c\_id = course.c\_id

WHERE student.st\_id = student\_id AND course.c\_id = course\_id AND cp\_present = TRUE);

SET @participation\_g\_total = ((participation\_present/participation\_count)\*100)\*(participation\_pct/100);

SET @assignment\_grade = (calculate\_category\_grade(student\_id, course\_id, 'assignment', assignment\_pct));

SET @project\_grade = (calculate\_category\_grade(student\_id, course\_id, 'project', project\_pct));

SET @exam\_grade = (calculate\_category\_grade(student\_id, course\_id, 'exam', exam\_pct));

SET @student\_grade = (@participation\_g\_total + @assignment\_grade + @project\_grade + @exam\_grade);

SELECT @participation\_g\_total AS Participation, @assignment\_grade AS Assignments, @project\_grade AS Projects, @exam\_grade AS Exams, @student\_grade AS Student\_Grade;

END

Calculate\_category\_grade function:

BEGIN

DECLARE done integer DEFAULT FALSE;

DECLARE cw\_total integer;

DECLARE cw\_id integer;

DECLARE cw\_marks integer;

DECLARE cw\_grade decimal(10,1);

DECLARE cur1 CURSOR FOR SELECT cw\_id, cw\_total FROM course\_coursework WHERE cw\_c\_id = course\_id AND cw\_course\_type = type;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET @cw\_grade = 0.00;

OPEN cur1;

cw\_loop: LOOP

FETCH cur1 INTO cw\_id, cw\_total;

IF done THEN

LEAVE cw\_loop;

END IF;

SET cw\_marks = ( SELECT marks FROM student\_course\_coursework

JOIN student\_courses ON student\_course\_coursework.stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

WHERE student\_course\_coursework.cw\_id = cw\_id AND student.st\_id = student\_id);

IF cw\_marks IS NOT NULL THEN

SET @cw\_grade = cw\_grade + (((cw\_marks/cw\_total)\*100)\*(grade\_pct/100));

END IF;

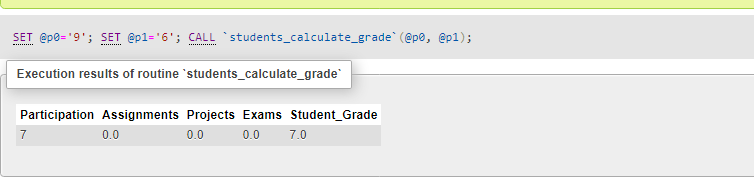
END LOOP;

CLOSE cur1;

RETURN @cw\_grade;

END

Test:



(student id: 9 and course id: 6)

1. Compute the grade for a student, where the lowest score for a given category is dropped

Code:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `students\_calculate\_grade\_minus\_min`(IN `student\_id` INT, IN `course\_id` INT)

BEGIN

DECLARE participation\_count integer;

DECLARE participation\_present integer;

DECLARE participation\_g\_total integer;

DECLARE participation\_pct integer;

DECLARE assignment\_pct integer;

DECLARE project\_pct integer;

DECLARE exam\_pct integer;

DECLARE assignment\_grade decimal(10,1);

DECLARE project\_grade decimal(10,1);

DECLARE exam\_grade decimal(10,1);

DECLARE student\_grade decimal(10,1);

SET participation\_count = (SELECT c\_weeks FROM course WHERE c\_id = course\_id);

SET participation\_pct = (SELECT c\_participation\_pts FROM course WHERE c\_id = course\_id);

SET assignment\_pct = (SELECT c\_assignment\_pts FROM course WHERE c\_id = course\_id);

SET project\_pct = (SELECT c\_project\_pts FROM course WHERE c\_id = course\_id);

SET exam\_pct = (SELECT c\_exam\_pts FROM course WHERE c\_id = course\_id);

SET participation\_present = (SELECT COUNT(cp\_id) FROM course\_participation

JOIN student\_courses ON course\_participation.cp\_stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

JOIN course ON student\_courses.c\_id = course.c\_id

WHERE student.st\_id = student\_id AND course.c\_id = course\_id AND cp\_present = TRUE);

SET @participation\_g\_total = ((participation\_present/participation\_count)\*100)\*(participation\_pct/100);

SET @assignment\_grade = (calculate\_category\_grade\_minus\_min(student\_id, course\_id, 'assignment', assignment\_pct));

SET @project\_grade = (calculate\_category\_grade\_minus\_min(student\_id, course\_id, 'project', project\_pct));

SET @exam\_grade = (calculate\_category\_grade\_minus\_min(student\_id, course\_id, 'exam', exam\_pct));

SET @student\_grade = (@participation\_g\_total + @assignment\_grade + @project\_grade + @exam\_grade);

SELECT @participation\_g\_total AS Participation, @assignment\_grade AS Assignments, @project\_grade AS Projects, @exam\_grade AS Exams, @student\_grade AS Student\_Grade;

END$$

DELIMITER ;

calculate\_category\_grade\_minus\_min function:

DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `calculate\_category\_grade\_minus\_min`(`course\_id` INT, `type` VARCHAR(50), `student\_id` INT, `grade\_pct` INT) RETURNS decimal(10,1)

BEGIN

DECLARE done integer DEFAULT FALSE;

DECLARE cw\_total integer;

DECLARE cw\_id integer;

DECLARE cw\_marks integer;

DECLARE cw\_grade decimal(10,1);

DECLARE min\_score decimal(10,1);

DECLARE grade\_score decimal(10,1);

DECLARE cur1 CURSOR FOR SELECT cw\_id, cw\_total FROM course\_coursework WHERE cw\_c\_id = course\_id AND cw\_course\_type = type;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET grade\_score = 0.0;

SET min\_score = 0.0;

OPEN cur1;

cw\_loop: LOOP

FETCH cur1 INTO cw\_id, cw\_total;

IF done THEN

LEAVE cw\_loop;

END IF;

SET cw\_marks = (SELECT marks FROM student\_course\_coursework

JOIN student\_courses ON student\_course\_coursework.stc\_id = student\_courses.stc\_id

JOIN student ON student\_courses.st\_id = student.st\_id

WHERE student\_course\_coursework.cw\_id = cw\_id AND student.st\_id = student\_id);

IF cw\_marks IS NOT NULL THEN

SET cw\_grade = ((cw\_marks/cw\_total)\*100)\*(grade\_pct/100);

IF min\_score = 0.0 THEN

SET min\_score = cw\_grade;

SET grade\_score = grade\_score + cw\_grade;

ELSEIF min\_score > cw\_grade THEN

SET min\_score = cw\_grade;

END IF;

SET grade\_score = grade\_score + cw\_grade;

END IF;

END LOOP;

CLOSE cur1;

SET grade\_score = grade\_score - min\_score;

RETURN grade\_score;

END$$

DELIMITER ;

Test:

