4. 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:

```
SET @p0='9'; CALL `calculateMinMaxAve`(@p0);
```

Execution results of routine `calculateMinMaxAve`

Average_Assignment_Score	Min_Assignment_Score	Max_Assignment_Score
42.333333333	38	50

5. 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:

```
SET @p0='9'; CALL `show_course_students`(@p0);
```

Execution results of routine `show_course_students`

id	name	email	phone	gender
6	Aline Adams	pede.cras.vulputate@yahoo.edu	1-913-450-2321	female
7	Carlos Mcclain	ante.nunc.mauris@yahoo.couk	1-776-533-9627	male

6. 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):

st_name	c_name	cw_id	cw_course_type	marks
Tatiana Morales	PPOU749	14	project	36
Tatiana Morales	PPOU749	39	project	15
Tatiana Morales	PPOU749	43	assignment	19
Aline Adams	PPOU749	22	project	40
Aline Adams	PPOU749	45	project	16
Aline Adams	PPOU749	47	exam	37
Aline Adams	PPOU749	49	assignment	23
Kelly Murray	PPOU749	6	assignment	14
Kelly Murray	PPOU749	12	project	16
Kelly Murray	PPOU749	16	assignment	43
Kelly Murray	PPOU749	18	assignment	9
Kelly Murray	PPOU749	30	project	38
Kelly Murray	PPOU749	40	project	44
Zoe Fox	PPOU749	14	project	44
Zoe Fox	PPOU749	17	exam	43
Zoe Fox	PPOU749	21	exam	48
Zoe Fox	PPOU749	24	exam	16

7. Add an assignment to a course;

   Code:

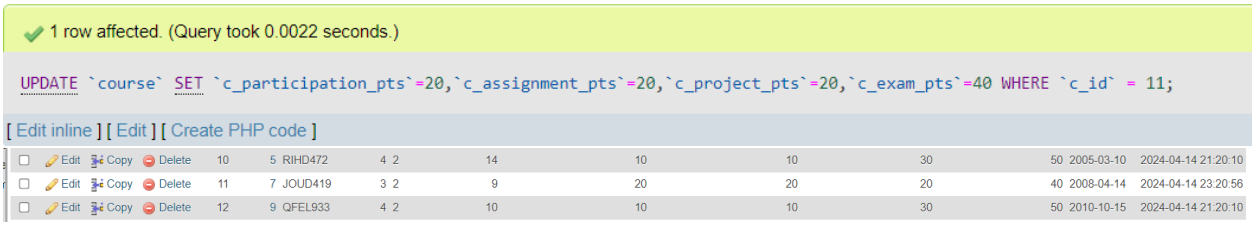
```
INSERT INTO course_coursework
(cw_id,cw_c_id,cw_course_type,cw_content,cw_release_date,cw_due_date,cw_total,created_at)
VALUES(NULL, 10, 'assignment',
'fsdhfihosdifghdfgivibsidbgibidfbgdibibfaifdbidsbfvsibgidnifdniobnifdbgsdiobgdoibgioboibgdiobgdifbg','2024-04-03','2024-04-10',50,'2024-03-21');
```

   Test:

8. 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:



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

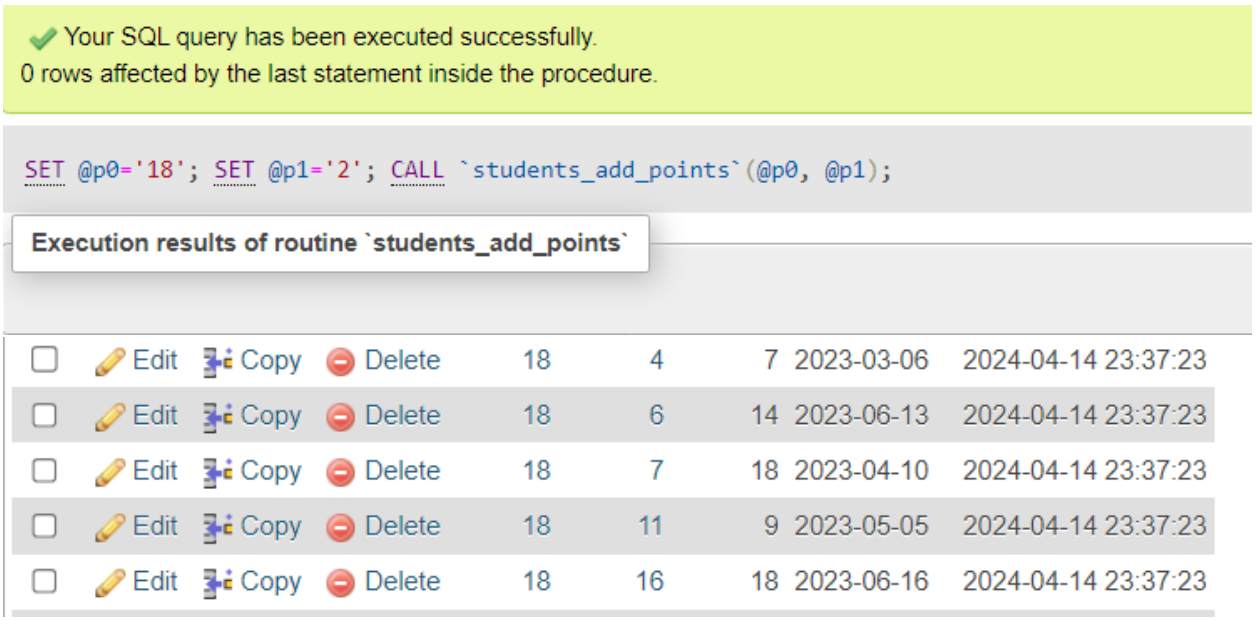
Code:

```
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:



10. 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

☐	Edit	Copy	Delete	18	4	7	2023-03-06	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	6	14	2023-06-13	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	7	18	2023-04-10	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	11	9	2023-05-05	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	16	18	2023-06-16	2024-04-14 23:37:23

Your SQL query has been executed successfully.
0 rows affected by the last statement inside the procedure.

```
SET @p0='18'; SET @p1='2'; CALL `students_add_points_q`(@p0, @p1);
```

Execution results of routine `students_add_points_q`

☐	Edit	Copy	Delete	18	4	7	2023-03-06	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	6	14	2023-06-13	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	7	18	2023-04-10	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	11	9	2023-05-05	2024-04-14 23:37:23
☐	Edit	Copy	Delete	18	16	18	2023-06-16	2024-04-14 23:37:23

11. 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:

```
SET @p0='9'; SET @p1='6'; CALL `students_calculate_grade`(@p0, @p1);
```

Execution results of routine `students_calculate_grade`

Participation	Assignments	Projects	Exams	Student_Grade
7	0.0	0.0	0.0	7.0

(student id: 9 and course id: 6)

12. 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:

```
SET @p0='9'; SET @p1='6'; CALL `students_calculate_grade_minus_min`(@p0, @p1);
```

Execution results of routine `students_calculate_grade_minus_min`

Participation	Assignments	Projects	Exams	Student_Grade
7	0.0	0.0	0.0	7.0