Procedures & triggers and report

Procedure 1:

```
DELIMITER //
CREATE PROCEDURE update_students(
    IN p_student_id INT,
    IN p_f_name VARCHAR(255),
    IN p_l_name VARCHAR(255),
    IN p_email VARCHAR(255),
    IN p_city VARCHAR(255),
    IN p_phone_number VARCHAR(15)
)
BEGIN
    UPDATE students
    SET f_name = p_f_name,
        l_name = p_l_name,
        email = p_email,
        city = p_city,
        phone_number = p_phone_number
    WHERE student_id = p_student_id;
END;
//
DELIMITER ;


CALL update_students(1, 'John', 'Doe', 'john.doe@example.com', 'New York', '123-456-7890');
```

Procedure 2:

```
DELIMITER //

CREATE PROCEDURE update_course(
   IN p_course_id INT,
   IN p_course_name VARCHAR(255))
BEGIN
   UPDATE courses
   SET course_name = p_course_name
   WHERE course_id = p_course_id;
END;
//

DELIMITER ;

CALL update_course(1, 'it');
```

Trigger to calculate avg GPA:

```
DELIMITER //

CREATE TRIGGER calculate_gpa
AFTER INSERT ON Test
FOR EACH ROW
BEGIN
    DECLARE v_total_points DECIMAL(8, 2);
    DECLARE v_total_hours INT;
    DECLARE v_gpa DECIMAL(8, 2);

    -- Calculate total points for all courses taken by the student
    SELECT COALESCE(SUM(Grades.grade * Courses.credit_hours), 0)
    INTO v_total_points
    FROM Test
    INNER JOIN Grades ON Test.grade_code = Grades.grade_code
    INNER JOIN Courses ON Test.course_id = Courses.course_id
    WHERE Test.student_id = NEW.student_id;

    -- Calculate total hours for all courses taken by the student
    SELECT COALESCE(SUM(Courses.credit_hours), 0)
    INTO v_total_hours
    FROM Test
    INNER JOIN Courses ON Test.course_id = Courses.course_id
```

```
    WHERE Test.student_id = NEW.student_id;

    -- Calculate GPA
    IF v_total_hours > 0 THEN
        SET v_gpa = v_total_points / v_total_hours;
    ELSE
        SET v_gpa = 0;
    END IF;

    -- Update GPA in the students table for the same student
    UPDATE students
    SET GPA = v_gpa
    WHERE student_id = NEW.student_id;
END;
//
```

Select statement to make the report table:

```
SELECT
   e.course_id,
   COUNT(DISTINCT e.student_id) AS num_students_enrolled,
   AVG(g.grade) AS average_grade
FROM
   enroll e
JOIN
   test t ON e.student_id = t.student_id AND e.course_id = t.course_id
JOIN
   grades g ON t.grade_code = g.grade_code
GROUP BY
   e.course_id;
```
