Day3

insert into students_courses
values
(1,4,60,NULL),
(2,1,NULL,NULL),
(2,4,75,NULL),
(3,1,NULL,NULL),
(3,2,NULL,NULL),
(3,3,75,NULL);

1 ***Create function to calculate the number of students who get grade less than 80 in a certain exam (course id will be sent as a parameter)***

```
 CREATE VIEW count_students_view AS
   -> SELECT course_id, COUNT(*) AS student_count
   -> FROM student_courses
   -> WHERE grade < 80
   -> GROUP BY course_id;

SELECT student_count
   -> FROM count_students_view
   -> WHERE course_id = course_id;


###(course_id= 1,2,3 =1)
```

2 ***Create stored procedure to display the names of the absence students of a certain courses.(Absent means has no grades)***

```
DELIMITER // ‘
CREATE PROCEDURE getAbsentStudents(IN courseID INT)
   -> BEGIN
   ->     SELECT CONCAT(s.first_name, ' ', s.last_name) AS student_name
   ->     FROM students s
   ->     LEFT JOIN student_courses sc ON s.student_id = sc.student_id
   ->     WHERE sc.course_id = courseID AND sc.grade IS NULL;
   -> END //
   ->  DELIMITER ;
 CALL getAbsentStudents(3);
```

3 ***Create stored procedure to calculate the average grades for certain course.***

```
DELIMITER //
CREATE PROCEDURE calculateAverageGrade(IN courseID INT)
   -> BEGIN
   ->     DECLARE totalGrades INT;
   ->     DECLARE totalStudents INT;
   ->     DECLARE averageGrade DECIMAL(5, 2);
   ->
   ->     SELECT COUNT(*) INTO totalStudents
   ->     FROM student_courses
   ->     WHERE course_id = courseID;
   ->
   ->     SELECT SUM(grade) INTO totalGrades
   ->     FROM student_courses
   ->     WHERE course_id = courseID;
   ->
   ->     SET averageGrade = totalGrades / totalStudents;
   ->
   ->     SELECT CONCAT(c.first_name, ' ', c.last_name) AS student_name,
sc.grade
   ->     FROM students c
   ->     INNER JOIN student_courses sc ON c.student_id = sc.student_id
   ->     WHERE sc.course_id = courseID;
   ->
   ->     SELECT averageGrade;
   -> END //
 DELIMITER ;
CALL calculateAverageGrade(3);
CALL calculateAverageGrade(1);
CALL calculateAverageGrade(2);
```

4

Create trigger to keep track the changes(updates) of the grades in the studnets_courses table
(create <u>changes table</u> with the following fields:
id int primary key ,
user varchar(30),
action varchar(40),
old_grade int,
new_grade int,
change_date date).

Test the trigger by updating grade int the "Students_courses" table

Confirm that the row is added in the" change_table"

```
  CREATE TABLE change_table (
    ->   id INT PRIMARY KEY AUTO_INCREMENT,
    ->   user VARCHAR(30),
    ->   action VARCHAR(40),
    ->   old_grade INT,
    ->   new_grade INT,
    ->   change_date DATE
    -> );
 DELIMITER //
 CREATE TRIGGER track_grade_changes
    -> AFTER UPDATE ON student_courses
    -> FOR EACH ROW
    -> BEGIN
    ->   INSERT INTO change_table (user, action,
old_grade, new_grade, change_date)
    ->   VALUES (USER(), 'Grade Update', OLD.grade,
NEW.grade, CURDATE());
    -> END //
 DELIMITER ;
 UPDATE student_courses
    -> SET grade = 90
    -> WHERE course_id = 2 AND student_id = 1;
select * from student_courses;
```

5 *Create event to delete the changes tables every 5 minute*

```
DELIMITER //
> CREATE EVENT delete_changes_event
-> ON SCHEDULE EVERY 5 MINUTE
-> DO
 -> BEGIN
 ->     DELETE FROM change_table;
 -> END//
DELIMITER ;
SHOW EVENTS;
```