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)***
	delimiter // create function cnt_below80(course_id int) -> returns int -> BEGIN -> declare cnt int; -> select count(*) into cnt from student_courses -> where course_id = student_courses.course_id and grade < 80; -> return cnt; -> end// delimiter ; call it by: select cnt_below80(3) as num_students_below_80;
2	***Create stored procedure to display the names of the absence students of a certain courses.(Absent means has no grades)***
	delimiter // MariaDB [grades]> CREATE PROCEDURE getabsent(IN course_id INT) -> begin -> SELECT s.first_name, s.last_name -> FROM students s -> INNER JOIN student_courses sc ON s.student_id = sc.student_id -> WHERE sc.course_id = course_id AND sc.grade IS NULL; -> end; -> // *delimiter ;*
3	***Create stored procedure to calculate the average grades for certain course.***

	DELIMITER // MariaDB [grades]> CREATE PROCEDURE avgcourse(IN course_id INT, OUT average_grade DECIMAL(5, 2)) -> begin -> SELECT AVG(grade) INTO average_grade -> FROM student_courses -> WHERE course_id = course_id; -> end; -> // delimiter ; ■ To call call avgcourse(1,@avg); -> select @avg;
4	***Create trigger to keep track the changes(updates) of the grades in the studnets_courses table*** ***(create changes table 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 changes (-> id INT AUTO_INCREMENT PRIMARY KEY, -> user VARCHAR(30), -> action VARCHAR(40), -> old_grade INT, -> new_grade INT, -> change_date DATE ->); Delimiter // CREATE TRIGGER after_update_student_grade -> AFTER UPDATE ON student_courses -> FOR EACH ROW -> BEGIN -> IF NEW.grade != OLD.grade THEN -> INSERT INTO changes (user, action, old_grade, new_grade, change_date) -> VALUES (CURRENT_USER(), 'Update Grade', OLD.grade, NEW.grade, CURDATE()); -> END IF; -> END; -> // DELIMITER ;

	■ update UPDATE student_courses -> SET grade = 9 -> WHERE course_id = 2 AND student_id = 1; -- to view SELECT * FROM changes;
5	***Create event to delete the changes tables every 5 minute***

	SET GLOBAL event_scheduler = ON;
	DELIMITER //
	
	CREATE EVENT delete_changes_data
	ON SCHEDULE EVERY 5 MINUTE
	DO
	BEGIN
	DELETE FROM changes;
	END;
	//
	
	DELIMITER ;
	
	To show:
	SHOW EVENTS;