MySQL Labs

MySQL (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)***

```
DROP FUNCTION IF EXISTS calculate_number_of_students;
DELIMITER $
CREATE FUNCTION calculate_number_of_students(id INTEGER)
RETURNS INT(4)
BEGIN
RETURN (SELECT COUNT(student_id)
            FROM students_courses
            WHERE (course_id = id) AND (grade < 80));
END$
DELIMITER ;

/* Test Function */
SELECT calculate_number_of_students(4);   /* Result = 3 */
```

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

```
DROP PROCEDURE IF EXISTS find_absent_students;
DELIMITER $
CREATE PROCEDURE find_absent_students(id INTEGER)
BEGIN
  SELECT CONCAT(first_name,” “,last_name) AS “Full_Name”
  FROM students s LEFT OUTER JOIN students_courses sc
  ON (s.student_id = sc.student_id)
  WHERE (sc.course_id = id) AND (grade  is NULL);
END$
DELIMITER ;

/* Test Procedure */
CALL find_absent_students(2); /* Result = Ahmed  Ossama*/
```

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

```
DROP PROCEDURE IF EXISTS calculate_average_grade;
DELIMITER $
CREATE PROCEDURE calculate_average_grade (id INTEGER)
BEGIN
  SELECT course_name,avg(grade)
  FROM courses c,students_courses sc
  WHERE (c.course_id = sc.course_id) AND (sc.course_id = id);
END$
DELIMITER ;

/* Test Procedure */
CALL calculate_average_grade (2); /* Result = 94.5000*/
```

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).

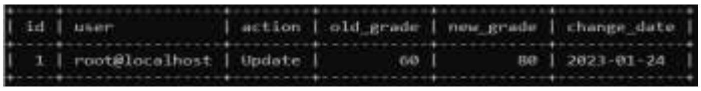
Test the trigger by updating grade int the “Students_courses” table

Confirm that the row is added in the” change_table”

```
/*Create Table changes*/
CREATE TABLE changes(
id INT NOT NULL,
user VARCHAR(30),
action VARCHAR(40),
old_grade INT,
new_grade INT,
change_date DATE,
PRIMARY KEY(id));

/*Create Trigger */
DELIMITER $
CREATE TRIGGER grade_updates_log
AFTER UPDATE ON students_courses
FOR EACH ROW
IF(NEW.grade <> OLD.grade)
THEN
INSERT INTO changes(id,user,action,old_grade,new_grade,change_date)
VALUES(OLD.student_id,CURRENT_USER(),”Update”,OLD.grade,NEW.grade,NOW());
END IF $
DELIMITER ;
```

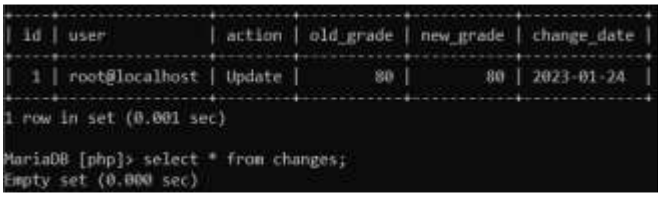
```
/*Test*/
UPDATE students_courses
SET grade = 80
WHERE (course_id = 4) AND (student_id = 1);
```



```
+----+----------------+--------+-----------+-----------+-------------+
| id | user           | action | old_grade | new_grade | change_date |
+----+----------------+--------+-----------+-----------+-------------+
|  1 | root@localhost | Update |        60 |        80 | 2023-01-24  |
+----+----------------+--------+-----------+-----------+-------------+
```

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

```
CREATE EVENT delete_changes
ON SCHEDULE EVERY 5 MINUTE
DO
DELETE FROM changes;
```



```
+----+----------------+--------+-----------+-----------+-------------+
| id | user           | action | old_grade | new_grade | change_date |
+----+----------------+--------+-----------+-----------+-------------+
|  1 | root@localhost | Update |        80 |        80 | 2023-01-24  |
+----+----------------+--------+-----------+-----------+-------------+
1 row in set (0.001 sec)

MariaDB [php]> select * from changes;
Empty set (0.000 sec)
```