SQL Lab 3: Database programming
(Trigger, Function and Data stored procedure)

1. Trigger

A trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. (Events include insert, update, and delete)

Example

Every insertion of student information requires another insertion of corresponding insurance information. It means that when the student's information is inserted into a student table, then his/her insurance information will then be automatically inserted into an insurance table as well.

1.1. To continue from the previous lab whose insurance schema is still missing, here we will create the insurance table and populate it as follows.

```
CREATE TABLE IF NOT EXISTS `faculty_insurance` (
`ref_id` char(16) NOT NULL primary key,
`ins_plan` varchar(50) NOT NULL,
`credit_limit` decimal(10,2) DEFAULT NULL,
`duedate` date DEFAULT NULL,
`s_timestamp` datetime DEFAULT NULL,
`status` char(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET= utf8mb4;
```

Initial data from the existing information.

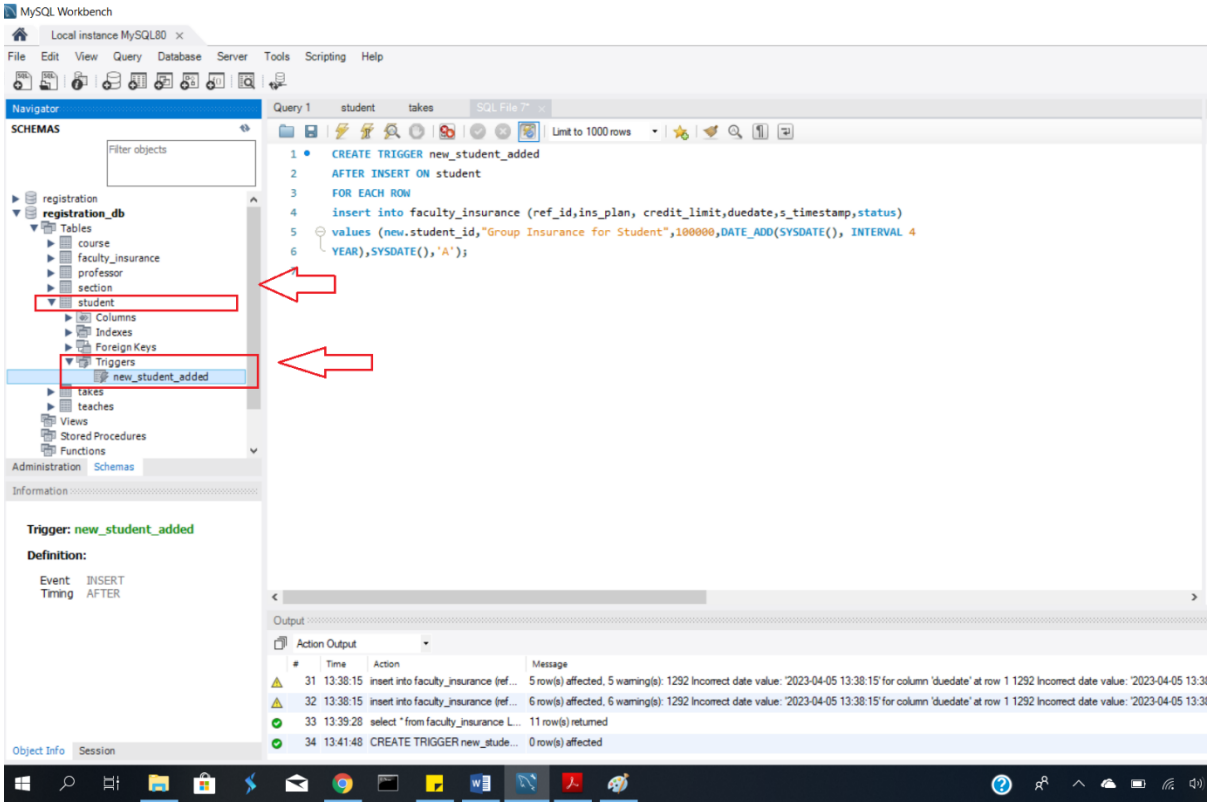
```
insert into faculty_insurance (ref_id, ins_plan, credit_limit, duedate, s_timestamp,status)
select pid,'initial value by system',40000,DATE_ADD(SYSDATE(), INTERVAL 4
YEAR),SYSDATE(),'A' from professor;
insert into faculty_insurance (ref_id,ins_plan, credit_limit,duedate,s_timestamp,status)
select student_id,'initial value by system',20000,DATE_ADD(SYSDATE(), INTERVAL 4
YEAR),SYSDATE(),'A' from student;
```

1.2. To create a trigger named "**new_student_added**", first open a new query, and type the following command to set up a trigger type "**after insert**" (there are various types of trigger, you can see the others at https://www.guru99.com/triggers-pl-sql.html)

```
CREATE TRIGGER new_student_added
AFTER INSERT ON student
```

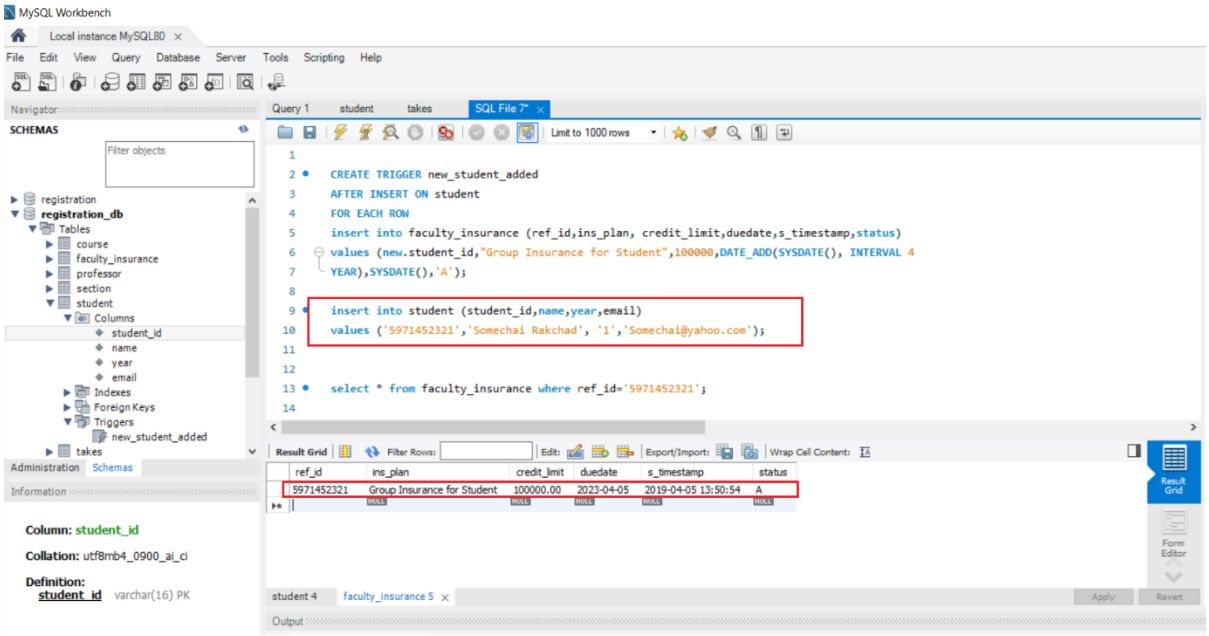
```
FOR EACH ROW
insert into faculty_insurance (ref_id,ins_plan, credit_limit,duedate,s_timestamp,status)
values (new.student_id,"Group Insurance for Student",100000,DATE_ADD(SYSDATE(),
INTERVAL 4
YEAR),SYSDATE(),'A');
```

You can see



1.3. After the trigger is created, you able to test the trigger by trying a query to insert new data record into student table and see the resulting data in the insurance table whose insurance information will be created automatically.

```
insert into student (student_id,name,year,email)
values ('5971452321','Somechai Rakchad', '1','Somechai@yahoo.com');
```



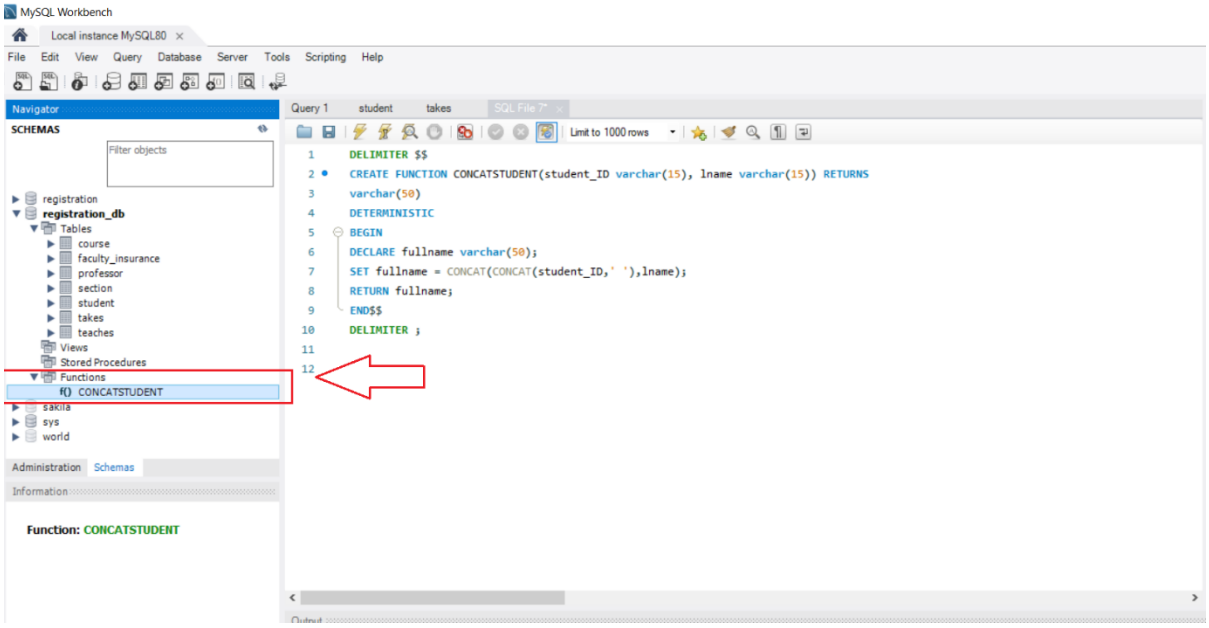
2. Function

We can create a function to support specific requirements. In this example, we create "CONCATSTUDENT()" function for concatenating the student's ID together with student's name. The function is executed along with either select or update command statements.

2.1 Open a new query and type the command as follow.

```
DELIMITER $$
CREATE FUNCTION CONCATSTUDENT(student_ID varchar(15), lname varchar(30))
RETURNS
varchar(50)
DETERMINISTIC
BEGIN
DECLARE fullname varchar(50);
SET fullname = CONCAT(CONCAT(student_ID,' '),lname);
RETURN fullname;
END$$
DELIMITER ;
```

You can see the details of function:



2.2 Test the created function by using a select command with specific inputs.

```
SELECT CONCATSTUDENT('5971463821','Honda Fukumika') as
CONCAT_ID_NAME_RESULT
```

2.3 Test the created function with a select command with populated data in a table.

```
SELECT student_id,CONCATSTUDENT(student_id,name) as CONCAT_ID_NAME_RESULT
,email From student
```

3. Data stored procedure

The use of data stored procedure is in common with that of function. The advantages of data stored procedure are that we can express the data manipulating commands more than one statement within a scope: insert, update, delete and call the function or call stored procedure. Moreover, it also allows us to create the temporary data tables for keeping information during the processing time. We can execute a data stored procedure by using the command statement:

"Call stored_procedure_name (parameter(s))"

Example requirements:

Update all students who get an F with **student_status= 'P'** and his/her insurance contract will be canceled (update **student_insurance.status='N'**), and the system also insert the historical data of changes into the **system_log** table.

3.1 Currently, we have no available column to store the logging information. Thus, we will add a column named "**student_status**" in student table and create the **system_log** table as follows.

```
alter table student add column student_status char(1);
```

```
CREATE TABLE IF NOT EXISTS system_log (
`id` int UNSIGNED AUTO_INCREMENT PRIMARY KEY,
`user_log` varchar(50) default NULL,
`remark` varchar(100) default NULL,
`timestamp` datetime default NULL);
```

3.2 Type command to create a data stored procedure as below.

```
DELIMITER $$
CREATE PROCEDURE Proc_flag_student()
DETERMINISTIC
BEGIN
if(select student_id from takes where grade='F')>0 THEN
CREATE TEMPORARY TABLE IF NOT EXISTS TMP_STUDENT(SID varchar(16));
Truncate table TMP_STUDENT;
insert into TMP_STUDENT (SID) select DISTINCT student_id from takes where grade='F';
update faculty_insurance set status='N' where ref_id in (select SID from TMP_STUDENT);
update student set student_status='P' where student_id in (select SID from TMP_STUDENT);
INSERT INTO system_log (user_log, remark,timestamp)
select SID, 'get F', SYSDATE() from TMP_STUDENT;
select * from student where student_id in (select SID from TMP_STUDENT);
ELSE
select ' F grade is empty';
END IF;
END$$
DELIMITER ;
```

Current data records in "takes" table may not have student who get grade F, thus we will update 1 record for testing the generated data stored procedure accordingly.

```
update takes set grade='F' where student_id=55748896 and cid=201002
```

3.3 Type a command to execute the data stored procedure and trace the effects.

```
Call Proc_flag_student()
```

4. Your task

4.1 Create trigger (2 points)

Each insertion of professor information, the data are inserted into not only professor table but also into faculty_insurance table that credit_limit value is calculated from 300% of his/her salary and ins_plan is "Group Insurance for Instructor".
(**trigger name: new_professor_added)
- Copy sql commands (both creating and testing command)
- Capture the result of the execution (show the result of insurance table)
- Save them in MS word with prob. No. 1

4.2 Create function (2 points)

Convert the number declared in a numerical data type to other currencies using function named "fn_currency(input_number, exchange_rate, currency_name)" and return the result as string.
For example: select fn_currency (70,35.00,USD)
or try to test on professor table using
"select *, fn_currency (70,35.00,USD) from professor"
Expected result of fn_currency must be 2 USD
Copy sql command , capture the result of execution and save them in MS word with prob. No. 2

4.3 Create data stored procedure (6 points)

Update salary of all professors who earns salary less than 30,000 up to 10% and update credit limit of insurance up to 400 % of new salary and also insert log into system_log table that stores the old salary, new salary, old credit limit and new credit limit. Finally, the data stored procedure has to print the name, old salary, new salary and credit limit of all professor information that are updated.
(**procedure name: Proc_cal_professor_upvel)
- Copy sql command
- Capture the results of execution (point out the manipulated data record(s): this data stored procedure will update data in 2 tables, insert data into 1 table, and print out a result)
- Save them in MS word with prob. No. 3

***** Convert your MS word solutions into .pdf file and submit it via MyCourseVille.****