EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING
AND
INFORMATION TECHNOLOGY

Week 4 Practice Assignment – Quarter 2, 2025-2026
Method: LAB 2 Practice 3
Course Name: Advanced Databases
Course Code: CSW 436
Student's Full Name: Phạm Trần Gia Hưng
Student ID: 2331200153

Important Instructions –

- Implement each question using MySQL Workbench.
- Document all question Output/results properly by capturing the screenshots of the output/results and SQL code for every question in a Word document and save it.
- After completing all questions, upload the document to Moodle.

Topic: Stored procedures and functions

1. Create a stored procedure named `getTodayDate` that:
   - Uses no IN parameters
   - Uses one OUT parameter to return the current system date

```
7   DELIMITER $$
8   CREATE PROCEDURE GetTodayDate(OUT Current_date_param DATE)
9   BEGIN
10      SET Current_date_param = CURDATE();
11  END $$
12  DELIMITER ;
13
14  CALL GetTodayDate(@today);
15  SELECT @today AS Today_Date;
16
17
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Today_Date
2026-01-19

2. Create a stored procedure named checkEvenOdd that:
   - Takes one IN parameter (an integer number)
   - Uses one OUT parameter to return:
     - "Even" if the number is even
     - "Odd" if the number is odd

```
22      -- o    "Odd" if the number is odd
23      DELIMITER $$
24      CREATE PROCEDURE CheckEvenOdd(IN num INT, OUT Result VARCHAR(10))
25      BEGIN
26          IF num % 2 = 0 THEN
27              SET Result = 'Even';
28          ELSE
29              SET Result = 'Odd';
30          END IF;
31      END $$
32      DELIMITER ;
33
34      CALL checkEvenOdd(10, @res);
35      SELECT @res;
36      CALL checkEvenOdd(7, @res);
37      SELECT @res;
38
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

@res
Even

```
34      CALL checkEvenOdd(10, @res);
35      SELECT @res;
36      CALL checkEvenOdd(7, @res);
37      SELECT @res;
38
```

Result Grid | Filter Rows:

@res
Odd

3. Write an SQL statement to create a table named students with the following columns:

- student_id (INT, Primary Key, Auto Increment)
- name (VARCHAR(50))
- marks (INT).

```sql
CREATE TABLE IF NOT EXISTS Student (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    marks INT NOT NULL
);
INSERT INTO Student (name, marks) VALUES
('wulfgard', 90),
('Pogranichnik', 50),
('Mifu', 80),
('Tangtang', 60);
select * from Student;

-- 4.   Write a stored procedure named addStudent
```

Result Grid | Filter Rows: | Edit:

student_id	name	marks
1	wulfgard	90
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
NULL	NULL	NULL

4. Write a stored procedure named addStudent that:

- Accepts IN parameters for name and marks.
- Inserts a new student record into the students table.

```sql
-- drop PROCEDURE addStudent;
DELIMITER $$
CREATE PROCEDURE addStudent(IN studentName VARCHAR(50), IN studentMarks INT)
BEGIN
    INSERT INTO Student VALUES (null,studentName, studentMarks);
END $$
DELIMITER ;

CALL addStudent('huichieh', 100);
select * from Student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	name	marks
1	wulfgard	90
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
5	huichieh	100
NULL	NULL	NULL

5. Write a stored procedure named getStudentById that:

   - Accepts one IN parameter (student_id).
   - Displays the name and marks of the student with the given ID

```
74    DELIMITER $$
75    CREATE PROCEDURE getStudentById(IN id INT)
76    BEGIN
77        SELECT name, marks
78        FROM Student
79        WHERE student_id = id;
80    END $$
81    DELIMITER ;
82
83    CALL getStudentById(2);
84
85
```

Result Grid | Filter Rows: | Export: | Wrap Ce

name	marks
Pogranichnik	50

6. Write a stored procedure named getTotalStudents that:

   - Uses one OUT parameter.
   - Returns the total number of students present in the students table.

```
DELIMITER $$
CREATE PROCEDURE getTotalStudents(OUT totalStudents INT)
BEGIN
    SELECT COUNT(*) INTO totalStudents
    FROM Student;
END $$
DELIMITER ;

CALL getTotalStudents(@total);
SELECT @total AS Total_Students;
```

7. Write a stored procedure named getResultStatus that:

   - Accepts one IN parameter (student_id).
   - Uses one OUT parameter to return: "Pass" if marks ≥ 40 and "Fail" if marks < 40.

```
106  DELIMITER $$
107  CREATE PROCEDURE getResultStatus(IN id INT, OUT resultStatus VARCHAR(10))
108  BEGIN
109      DECLARE studentMarks INT;
110
111      SELECT marks INTO studentMarks
112      FROM Student
113      WHERE student_id = id;
114
115      IF studentMarks >= 40 THEN
116          SET resultStatus = 'Pass';
117      ELSE
118          SET resultStatus = 'Fail';
119      END IF;
120  END $$
121  DELIMITER ;
122
123  CALL getResultStatus(1, @status);
124  SELECT @status AS Result;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Result
Pass

Student id 1 have 90 so pass

With fail condition:

```
126  CALL addStudent('pramanix', 39);
127  select * from Student;
```

Result Grid | Filter Rows: | Edit:

student_id	name	marks
1	wulfgard	90
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
5	huichieh	100
6	pramanix	39
NULL	NULL	NULL

```
126 •   CALL addStudent('pramanix', 39);
127 •   select * from Student;
128 •   CALL getResultStatus(6, @status);
129 •   SELECT @status AS Result;
130
```

Result Grid | Filter Rows: | Ex

Result
Fail

8. Write a stored procedure named updateMarks that:

   - Accepts IN parameters (student_id, new_marks).
   - Updates the marks of the specified student.

   Before:

student_id	name	marks
1	wulfgard	90

   After:

```
133     -- •    Updates the marks of the specified student.
134     DELIMITER $$
135 •   CREATE PROCEDURE updateMarks(IN id INT, IN newMark INT)
136     BEGIN
137         UPDATE Student
138         SET marks = newMark
139         WHERE student_id = id;
140     END $$
141     DELIMITER ;
142
143 •   CALL updateMarks(1, 75);
144 •   SELECT * FROM Student WHERE student_id = 1;
145
```

Result Grid | Filter Rows: | Edit: | Export/Import:

student_id	name	marks
1	wulfgard	75
NULL	NULL	NULL

9. Write a stored procedure named deleteStudent that:

   - Accepts one IN parameter (student_id).
   - Deletes the student record with the given ID from the table.

Before:

student_id	name	marks
1	wulfgard	75
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
5	huichieh	100
6	pramanix	39
NULL	NULL	NULL

After:

```
150  DELIMITER $$
151  CREATE PROCEDURE deleteStudent(IN id INT)
152  BEGIN
153      DELETE FROM Student
154      WHERE student_id = id;
155  END $$
156  DELIMITER ;
157
158  CALL deleteStudent(5);
159  SELECT * FROM Student;
160
161
162
```

Result Grid | Filter Rows: | Edit:

student_id	name	marks
1	wulfgard	75
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
6	pramanix	39
NULL	NULL	NULL

10. Write a function to calculate grade that:

- Accepts marks as input
- Returns:
  - 'A' if marks ≥ 80
  - 'B' if marks ≥ 60
  - 'C' if marks ≥ 40
  - ‘F' if marks < 40

If grade is 85

```
168     -- o    'F' if marks < 40
169     DELIMITER $$
170 •   CREATE FUNCTION calculateGrade(marks INT)
171     RETURNS CHAR(1) DETERMINISTIC
172     BEGIN
173         DECLARE grade CHAR(1);
174         IF marks >= 80 THEN
175             SET grade = 'A';
176         ELSEIF marks >= 60 THEN
177             SET grade = 'B';
178         ELSEIF marks >= 40 THEN
179             SET grade = 'C';
180         ELSE
181             SET grade = 'F';
182         END IF;
183         RETURN grade;
184     END $$
185     DELIMITER ;
186
187 •   SELECT calculateGrade(85) AS Grade;   -- A
```

Result Grid | Filter Rows: | Export: | Wrap

Grade
A

If grade is 65

```
184         RETURN grade;
185     END $$
186     DELIMITER ;
187
188 •   SELECT calculateGrade(85) AS Grade;
189 •   SELECT calculateGrade(65) AS Grade;
190 •   SELECT calculateGrade(45) AS Grade;
191 •   SELECT calculateGrade(30) AS Grade;
```

Result Grid | Filter Rows: | Expo

Grade
B

If grade is 45

```
189 •   SELECT calculateGrade(65) AS Grade;   -- B
190 •   SELECT calculateGrade(45) AS Grade;   -- C
191 •   SELECT calculateGrade(30) AS Grade;   -- F
```

Result Grid | Filter Rows: | Export:

Grade
C

If grade is 30

```
190 •   SELECT calculateGrade(45) AS Grade;
191 •   SELECT calculateGrade(30) AS Grade;
```

Result Grid | Filter Rows: | Export

Grade
F

11. Write a function named checkPassFail that used to return Pass or Fail
    - Accepts student marks as input
    - Returns 'Pass' if marks ≥ 40
    - Returns 'Fail' if marks < 40

Student mark: (id 1 updated to 75 after question 8)

student_id	name	marks
1	wulfgard	75
2	Pogranichnik	50
3	Mifu	80
4	Tangtang	60
6	pramanix	39
NULL	NULL	NULL

```
214    DELIMITER $$
215    CREATE FUNCTION checkPassFailById(id INT)
216    RETURNS VARCHAR(10) DETERMINISTIC
217    BEGIN
218        DECLARE studentMarks INT;
219        SELECT marks
220        INTO studentMarks
221        FROM Student
222        WHERE student_id = id;
223        IF studentMarks >= 40 THEN
224            RETURN 'Pass';
225        ELSE
226            RETURN 'Fail';
227        END IF;
228    END $$
229    DELIMITER ;
230
231    SELECT student_id, name, marks, checkPassFailById(student_id) AS Result
232    FROM Student;
233
234
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	name	marks	Result
1	wulfgard	75	Pass
2	Pogranichnik	50	Pass
3	Mifu	80	Pass
4	Tangtang	60	Pass
6	pramanix	39	Fail

12. Write a function named totalStudents that:
    - Returns the total number of records present in the students table

EIU

```
218    DELIMITER $$
219    CREATE FUNCTION totalStudents()
220    RETURNS INT DETERMINISTIC
221    BEGIN
222        DECLARE total INT;
223        SELECT COUNT(*) INTO total
224        FROM Student;
225        RETURN total;
226    END $$
227    DELIMITER ;
228
229    SELECT totalStudents() AS Total_Students;
```

Result Grid | Filter Rows: | Export:

Total_Students
5