COMP2350/COMP6350 Database Systems

Practical – Week 9

MySQL Procedural Programming

Objectives:

- To create the first stored procedure and understand the building blocks of a stored procedure.
- To formulate some simple stored procedures.
- To understand how to use conditional and iterative control structure in a stored procedure.

Introduction to MySQL stored procedure

1. Creating the very first stored procedure

- Run MySQL Workbench and then open a connection to the database server.
- Open a New Query tab in WorkBench environment and type in the procedure into the query interface as shown in Figure 1:

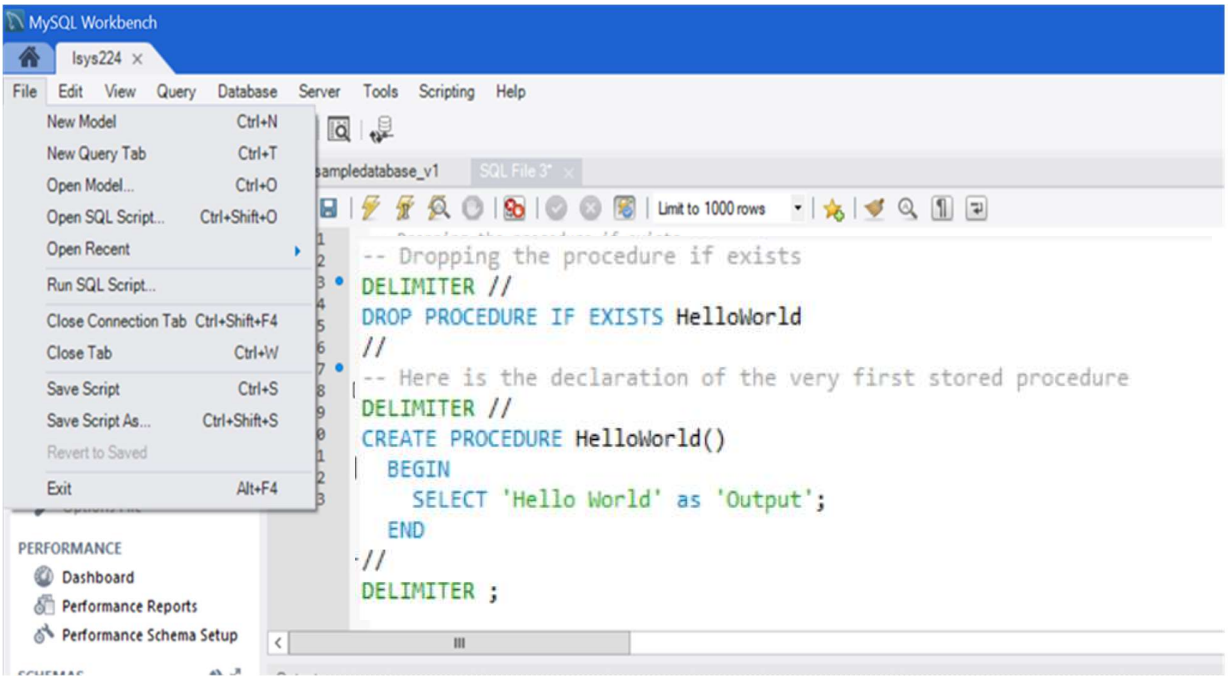

Figure 1: The Environment

- Next, Execute the stored procedure by pressing CTRL+SHIFT+ENTER or clicking on  .

- This will create a stored procedure and it will be stored in the database as shown in Figure 2:

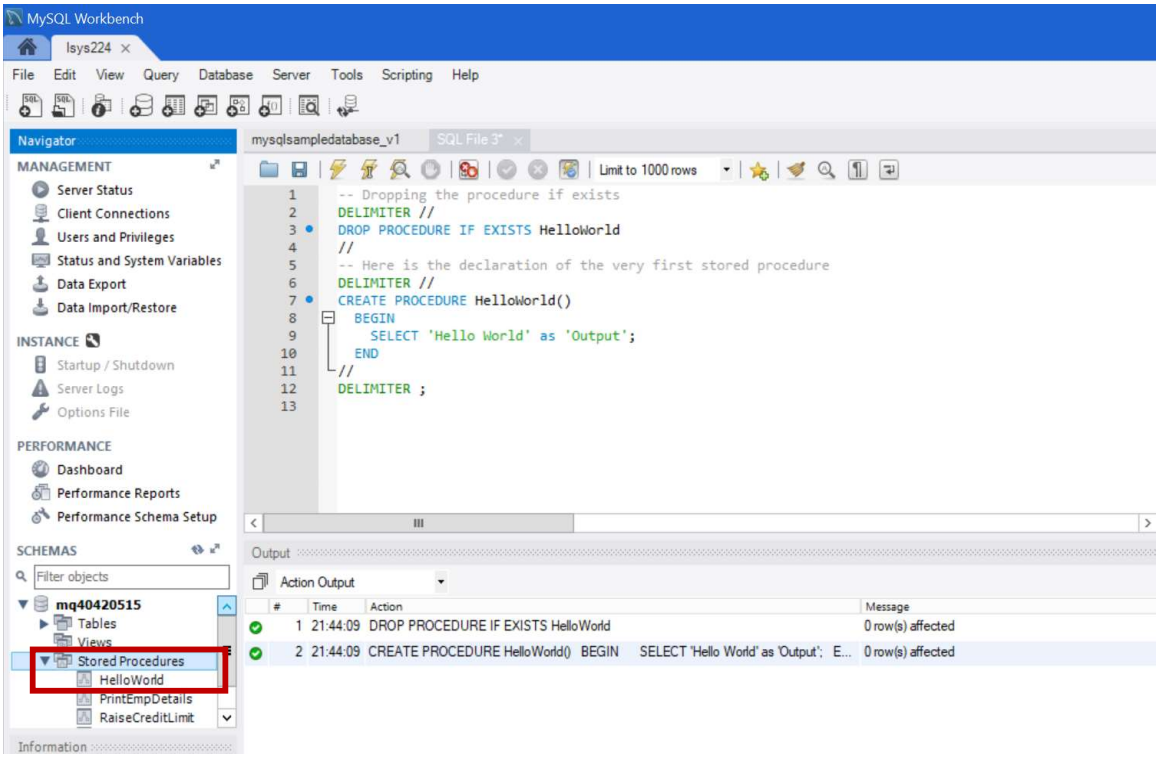

Figure 2: Creating the very first procedure

2. *Calling the stored procedure*

- To call the stored procedure, execute the statement as shown in Figure 3.

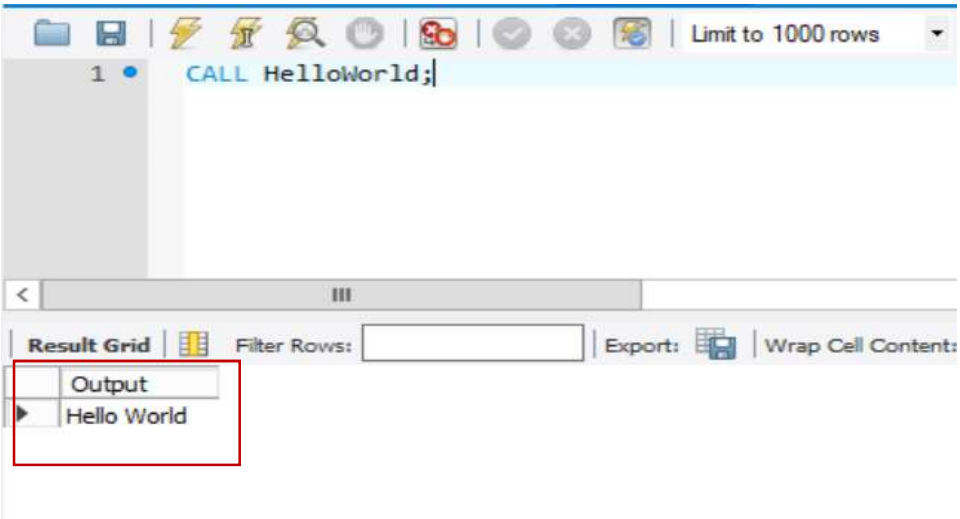

Figure 3: Result of calling the procedure

.

Task Specification

1. Write a stored procedure to displays a welcome message, such as 'Welcome to MySQL Procedural Programming'. Execute the procedure and then call it to display the message. You can use the following template:

```
DELIMITER //
DROP PROCEDURE IF EXISTS …
//
CREATE PROCEDURE …
  BEGIN
     …
  END
  //
DELIMITER ;

CALL …;
```

2. What output would you expect from the following stored procedure?

```
-- Dropping the procedure if exists
DELIMITER //
DROP PROCEDURE IF EXISTS devideNumbers
//

-- Declaring the Procedure
DELIMITER //
CREATE PROCEDURE devideNumbers()
BEGIN

DECLARE firstNumber, secondNumber INT; -- Declaring the variables
DECLARE result NUMERIC(6,2) DEFAULT 0;

SET firstNumber = 82;
SET secondNumber =15;
SET result = firstNumber/secondNumber;

SELECT result as 'Output';

END
//

DELIMITER ;

-- Calling the procedure
CALL devideNumbers();
```

3. Create and call the above procedure (devideNumbers) and verify the output.

4. Run the following code and fix the syntax errors that you encounter:

```
DELIMITER //
 DROP PROCEDURE IF EXISTS CalculateAverage
//

DELIMITER //
```

```
CREATE PROCEDURE CalculateAverage()
BEGIN
        DECLARE lower_limit, upper_limit INT;
        DECLARE num_count, total INT 0 ;
        DECLARE range_average Numeric(5,2) DEFAULT 0 ;
        SET lower_limit = 3;
        SET upper_limit := 15;
        WHILE (lower_limit + num_count)  <= upper_limit DO ;
          SET total = total + lower_limit + num_count;
          SET  num_count = num_count + 1

     range_average = total / num_count ;
        SELECT range_average 'Range Average';
 END
DELIMITER ;


CALL CalculateAverage;
```

5. Write a stored procedure that takes a number as an input and prints whether it is an odd number or an even number. You may use the following procedure template:

```
CREATE PROCEDURE OddOrEven(IN myNumber INT)
 BEGIN
 ....
 END

CALL OddOrEven(123);
```

[**Hint: (n % d) divides n by d, and returns the integer remainder from the operation**]

6. A shop gives a discount according to the code specified on the product tag. Each product has a price and a discount code. The discount code and corresponding discount in percentage are A (20%), B (15%), C (10%) and D (5%). The shop calculates the sale price of a product using this discount code. For example, if a product has a price of 200 dollar and discount code is C then,

   *discount amount = 200*10% = 20;*
   sale price = price – discount amount = 200-20 = 180;

   Write a stored procedure that takes the price and the discount code of the product as inputs and prints the sale price of the product.
   [**HINT: you can use either IF-ELSEIF-ELSE statement or a CASE statement; need to be able to deal with the case of invalid discount codes**]

7. Write a stored procedure that takes two numbers as input and print all odd numbers within that range. For example, if the procedure takes two numbers 4 and 11 as input, it prints 5, 7, 9 and 11 as output.