COMP2350/COMP6350 Database Systems

Practical – Week 10

MySQL Procedural Programming

Objectives:

- To define and execute stored procedures interacting with a database.
- To create the first stored function and understand the building blocks of a stored function.
- To understand how to use a stored function within a SQL statement or a stored procedure.
- To formulate some stored procedures and stored functions.

Sample Database schema:

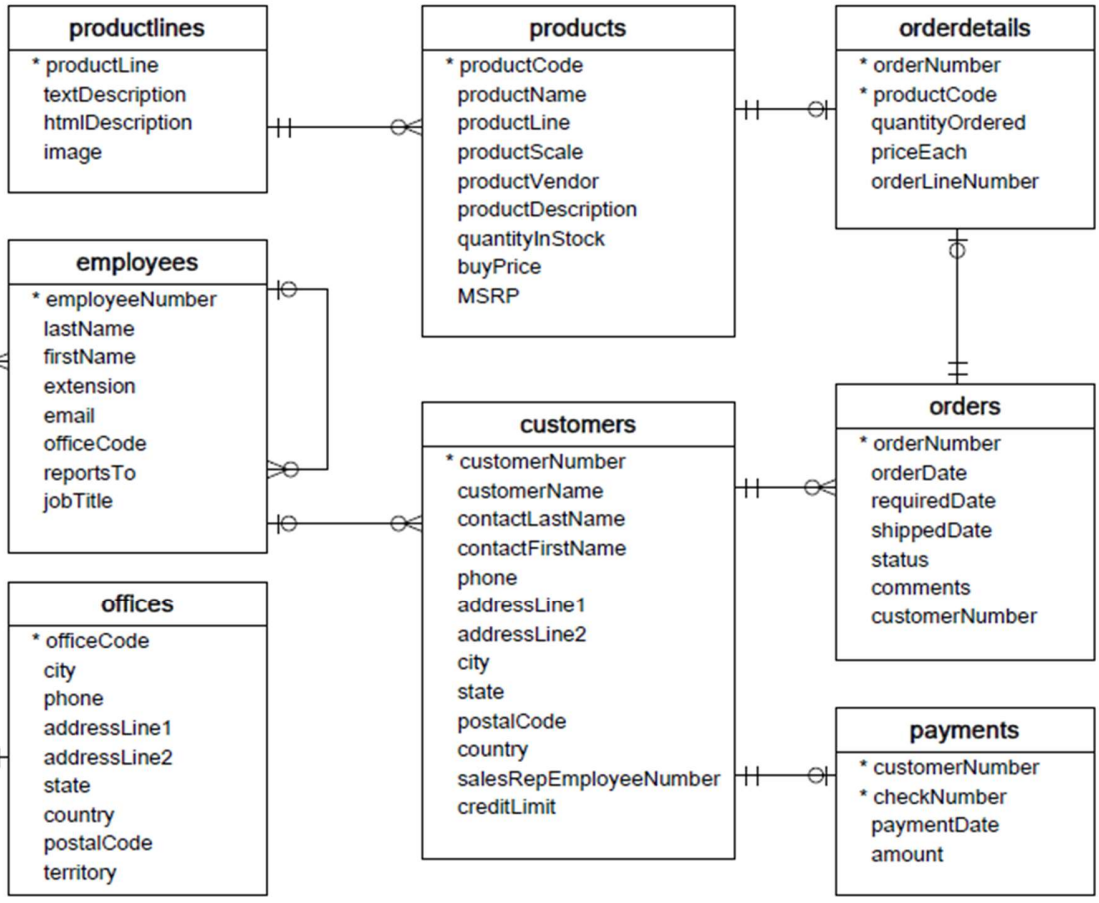

Figure 1: Sample Database Schema Diagram

MySQL Stored Function

1. Creating the first stored function

- Run MySQL Workbench and then open a connection to the database server.
- Open a New Query tab in WorkBench environment and type in the function into the query interface as shown in Figure 2.

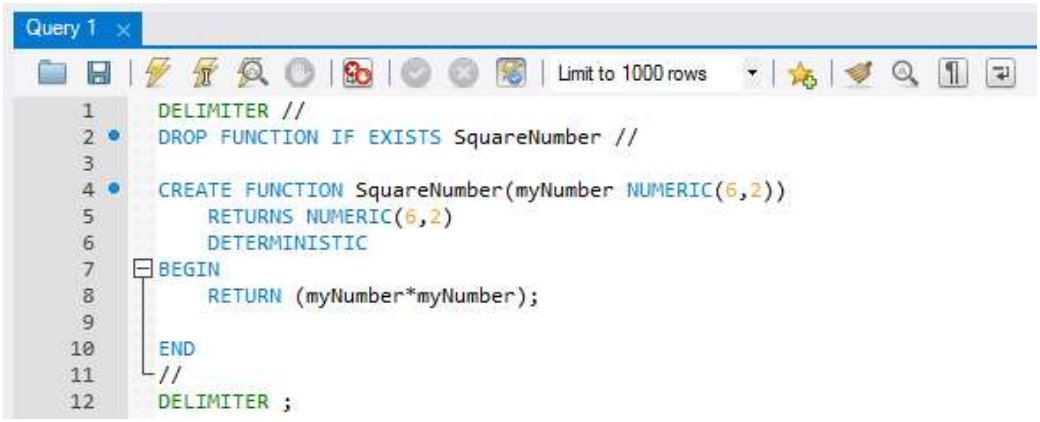

Figure 2: SquareNumber Function Definition

- Next, Execute the stored procedure by pressing CTRL+SHIFT+ENTER or clicking on  .
- This will create the stored function. 'SquareNumber', and it will be stored in the database as shown in Figure 3.

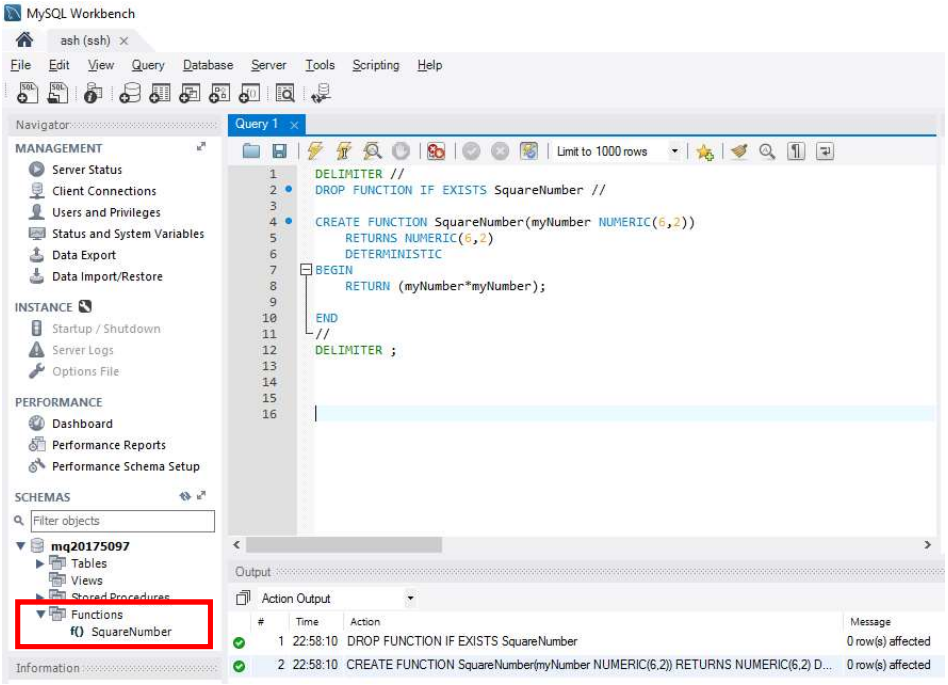

Figure 3: Function created and stored within database

2. Executing the first stored function

- **Unlike a stored procedure, stored function cannot be executed using CALL.**
- Stored functions can be used within a SQL statement and within a stored procedure.
- Execute the SQL statement to call SquareNumber function as shown in Figure 4.

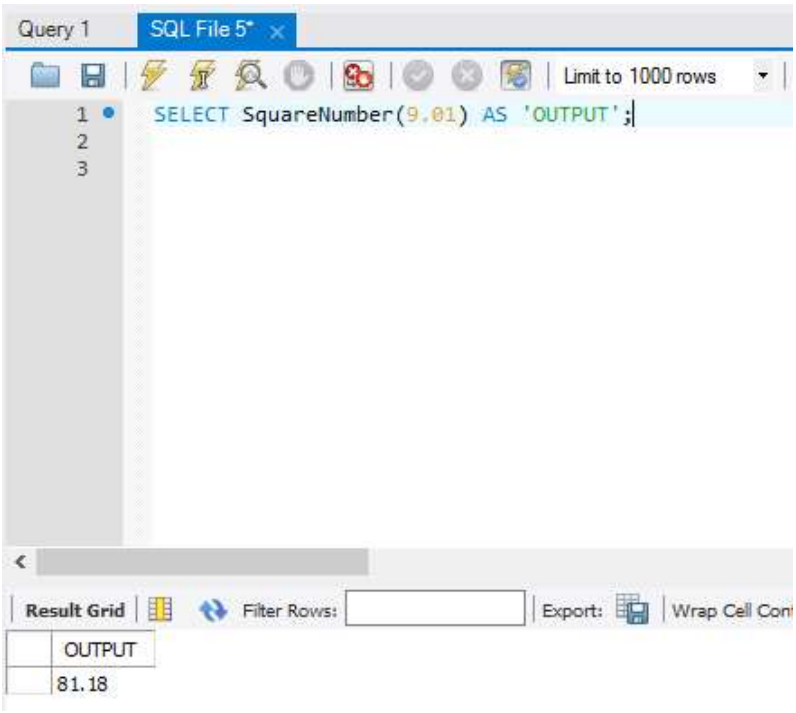

Figure 4: Result of calling a stored function within a SQL statement

Creating the Sample Database

Create a sample database by running the SQL script (mysqlsampledatabase_v1) provided in the "Practical" Folder of week 10.

Task Specifications:

1. Given the following stored procedure.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS ASequence $$
CREATE PROCEDURE ASequence (IN n INT)
BEGIN
    DECLARE a, b, i INT DEFAULT 1;
    DECLARE temp int;
    DECLARE str TEXT DEFAULT '';
    SET str = concat (str, a, ' ');
    SET str = concat (str, b, ' ');
    WHILE i<= n-2 DO
        SET temp = a + b;
        SET str = concat (str, temp, ' ');
        SET a = b;
        SET b = temp;
        SET i = i+1;
    END WHILE;
    SELECT str AS 'The Sequence';
END
$$
DELIMITER ;
CALL ASequence (10);
```

Figure 5: Stored procedure for Task 1

   a) What output would you expect by calling the procedure with the following statement?

      CALL ASequence(10);

   b) Verify your answer by executing and calling the procedure.

2. Write a stored procedure that returns the total amount payable for a given order; where the total amount payable for an order is calculated as:

   *sum(quantityOrdered*priceEach)*

3. Test your procedure by printing the total amount payable for order number 10100.

4. Can we rewrite Task 2 using a stored function? What will be the return type of the function? Write the stored function to calculate the total amount payable for a given order.

5. Write a SQL statement to display all the orders and the amount payable for the orders. *[Hint: can we use the function from Task 4?]*

6. Write a stored procedure that takes a country name as input and returns the total number of customers living in that given country.

7. Test your procedure by printing the total number of customers from Australia.

8. Write a stored procedure that takes a customer number as input and computes bonus credit amount based on the his/her current credit limit. The bonus is calculated as per the following table. The procedure also updates the database with the new credit limit for each eligible customer.

Credit Limit	Bonus Credit
>= 200000	Increase current credit limit by 50000
>= 150000 and < 200000	Increase current credit limit by 30000
>= 100000 and < 150000	Increase current credit limit by 15000
>= 50000 and < 100000	Increase current credit limit by 5000
< 50000	No bonus