Experiment No 9
Aim: Implementation of PL/SQL Functions & Procedures
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72
Performance Date: 18-04-2021

Theory:

Stored Functions:

- A stored function is a special kind stored program that returns a single value. Typically, you use stored functions to encapsulate common formulas or business rules that are reusable among SQL statements or stored programs.
- Different from a stored procedure, you can use a stored function in SQL statements wherever an expression is used. This helps improve the readability and maintainability of the procedural code.
- To create a stored function, you use the CREATE FUNCTION statement.

Syntax:

```
DELIMITER $$
CREATE FUNCTION function_name(
     param1,
     param2,…
)
RETURNS datatype
[NOT] DETERMINISTIC
BEGIN
 -- statements
END $$
DELIMITER;
```

Example:


```
DELIMITER $$
CREATE FUNCTION CustomerLevel(
     credit DECIMAL(10,2)
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
     DECLARE customerLevel VARCHAR(20);
  IF credit > 50000 THEN
          SET customerLevel = 'PLATINUM';
     ELSEIF (credit >= 50000 AND
               credit <= 10000) THEN
     SET customerLevel = 'GOLD';
     ELSEIF credit < 10000 THEN
     SET customerLevel = 'SILVER';
     END IF;
     -- return the customer level
     RETURN (customerLevel);
END$$
```

Stored Procedure:

- A procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database.
- It is a subroutine or a subprogram in the regular computing language.
- A procedure always contains a name, parameter lists, and SQL statements.
- We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc.
- It was first introduced in MySQL version 5. Presently, it can be supported by almost all relational database systems.

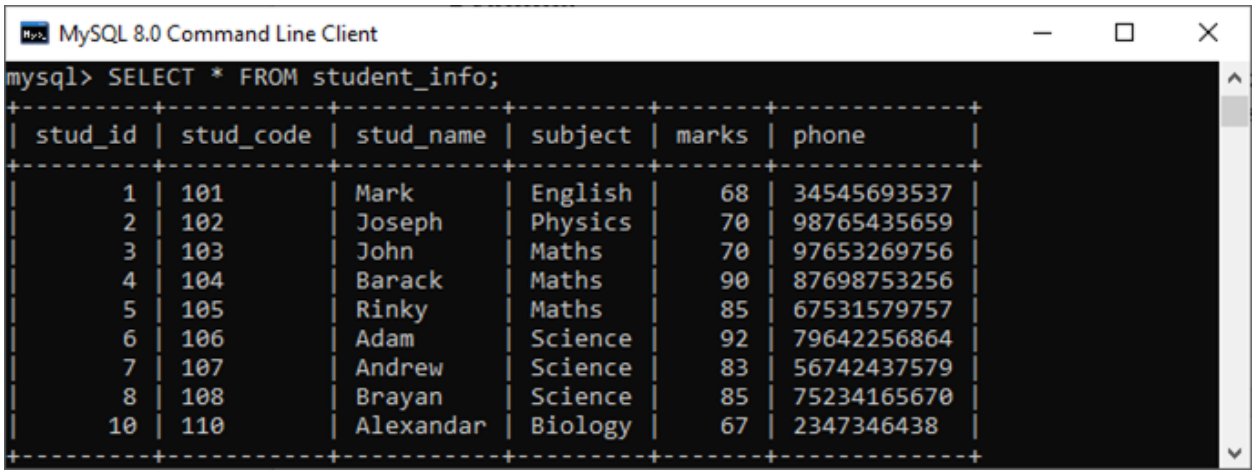
Features:

- Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.
- Stored procedure reduces the traffic between application and database server. Because the application has to send only the stored procedure's name and parameters instead of sending multiple SQL statements.
- Stored procedures are reusable and transparent to any applications.
- A procedure is always secure. The database administrator can grant permissions to applications that access stored procedures in the database without giving any permissions on the database tables.

Syntax:

```
DELIMITER &&
CREATE PROCEDURE procedure_name [[IN | OUT | INOUT]
parameter_name datatype [, parameter datatype]) ]
BEGIN
    Declaration_section
    Executable_section
END &&
DELIMITER ;
```

Example:




```
DELIMITER &&
CREATE PROCEDURE get_merit_student ()
BEGIN
     SELECT * FROM student_info WHERE marks > 70;
     SELECT COUNT(stud_code) AS Total_Student FROM student_info;
END &&
DELIMITER;
```

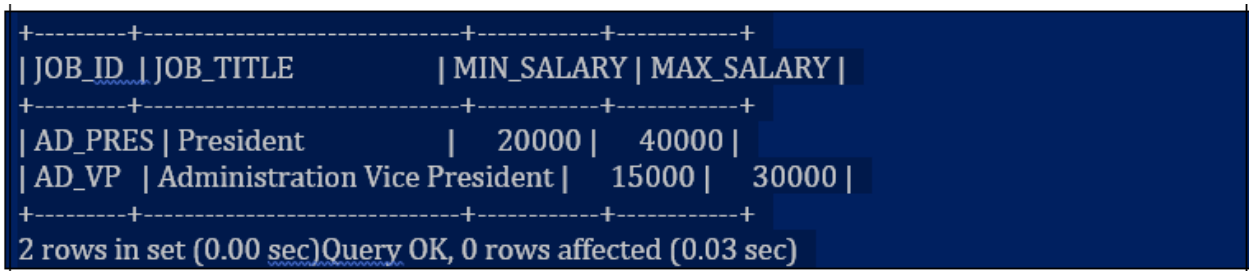
Procedures w/ Parameters:

IN Parameter example:

```
CREATE PROCEDURE my_proc_IN (IN var1 INT)
BEGIN
SELECT * FROM jobs LIMIT var1;
END$$
```

To execute the first 2 rows from the 'jobs' table execute the following command:

```
mysql> CALL my_proc_in(2)$$
```



```
mysql> CALL my_proc_in(5)$$
```

```
+------------+-----------------------------+------------+------------+
| JOB_ID     | JOB_TITLE                   | MIN_SALARY | MAX_SALARY |
+------------+-----------------------------+------------+------------+
| AD_PRES    | President                   |      20000 |      40000 |
| AD_VP      | Administration Vice President |      15000 |      30000 |
| AD_ASST    | Administration Assistant    |       3000 |       6000 |
| FI_MGR     | Finance Manager             |       8200 |      16000 |
| FI_ACCOUNT | Accountant                  |       4200 |       9000 |
+------------+-----------------------------+------------+------------+
5 rows in set (0.00 sec)Query OK, 0 rows affected (0.05 sec)
```

OUT Parameter example:

The following example shows a simple stored procedure that uses an OUT parameter. Within the procedure MySQL MAX() function retrieves maximum salary from MAX_SALARY of jobs table:

```
CREATE PROCEDURE my_proc_OUT (OUT highest_salary INT)
BEGIN
SELECT MAX(MAX_SALARY) INTO highest_salary FROM JOBS;
END$$
```

In the body of the procedure, the parameter will get the highest salary from MAX_SALARY column. After calling the procedure the word OUT tells the DBMS that the value goes out from the procedure. Here highest_salary is the name of the output parameter and we have passed its value to a session variable named @M, in the CALL statement:

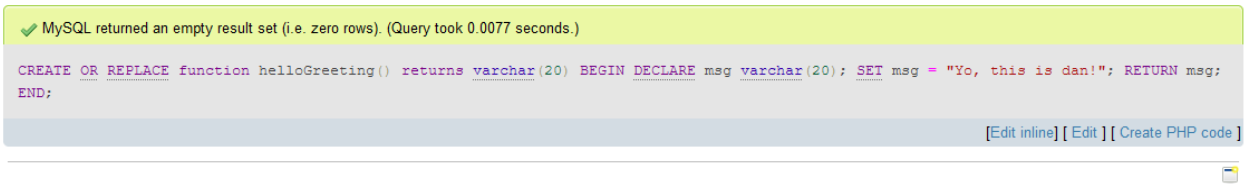
```
mysql> CALL my_proc_OUT(@M)$$
```

```
mysql< SELECT @M$$+-------+
| @M    |
+-------+
| 40000 |
+-------+
1 row in set (0.00 sec)
```

Functions:

Query:

```
DELIMITER $
CREATE OR REPLACE function helloGreeting() returns varchar(20)
BEGIN
     DECLARE msg varchar(20);
     SET msg = “Yo, this is dan!”;
     RETURN msg;
END;
$
```

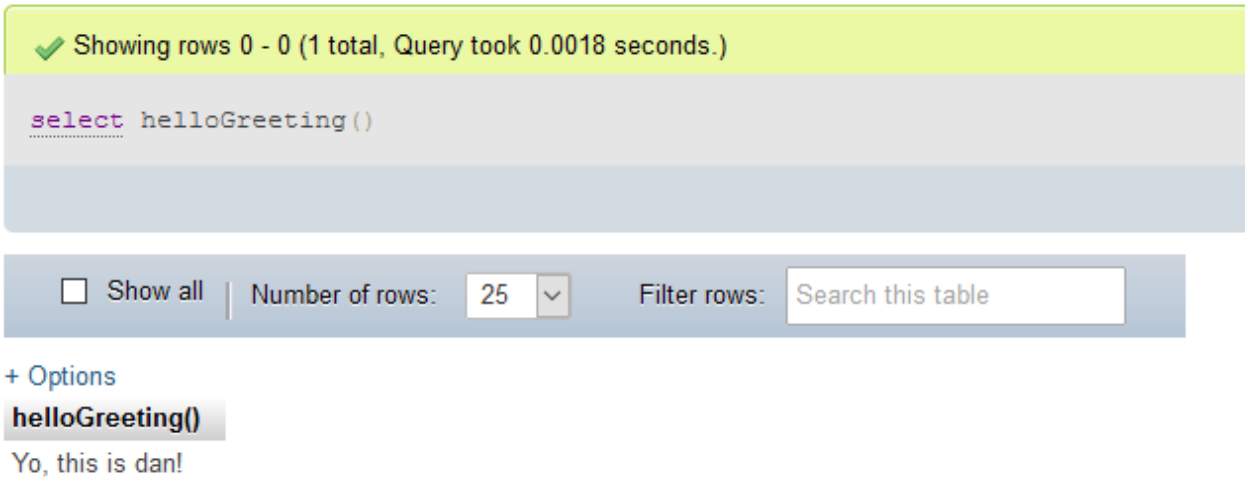


Query:

```
show function status where db = 'exp3'
```

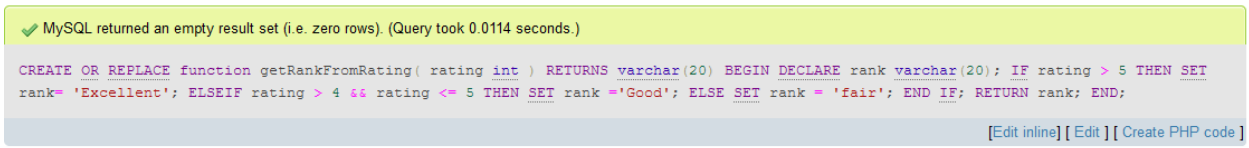
Query:

```
select helloGreeting()
```



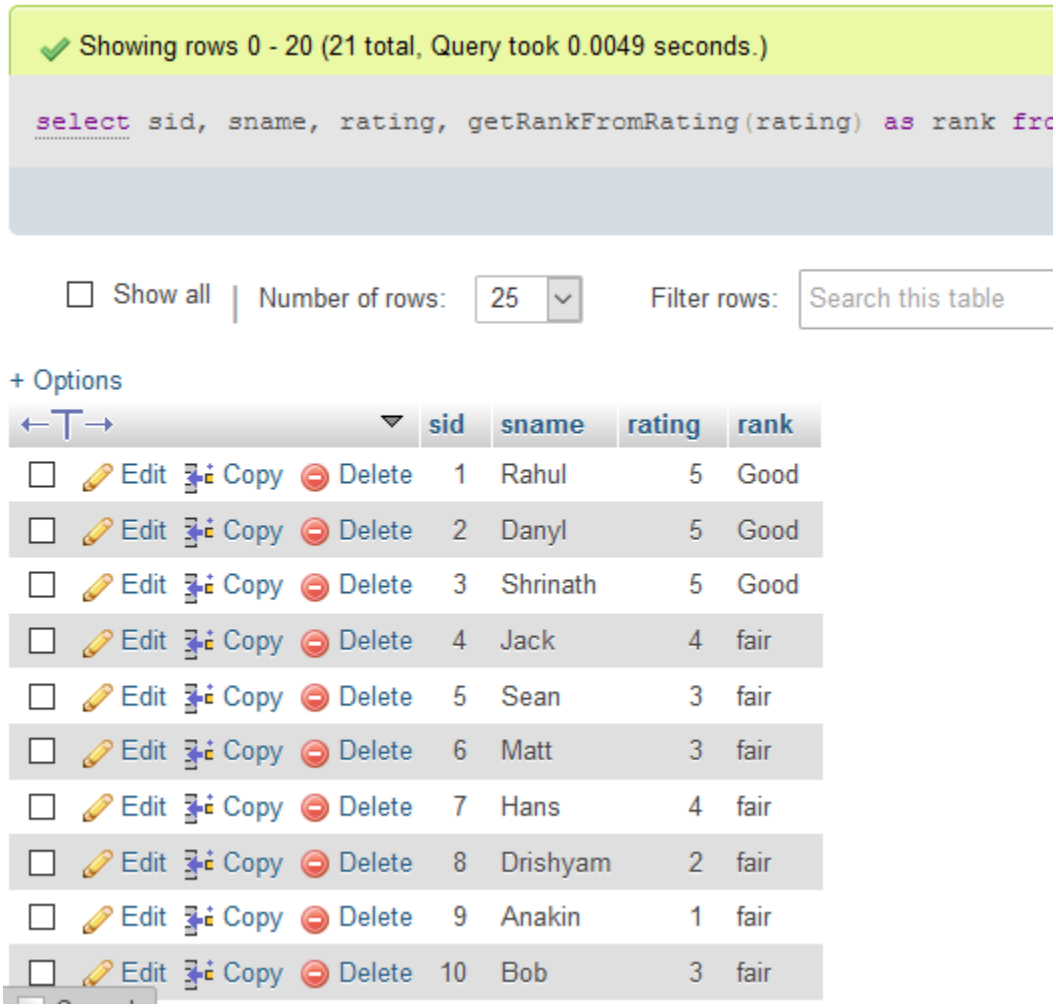
Query:

```
DELIMITER $
CREATE OR REPLACE function getRankFromRating(
     rating int
) RETURNS varchar(20)
BEGIN
     DECLARE rank varchar(20);
     IF rating > 5 THEN
          SET rank= 'Excellent';
     ELSEIF rating > 4 && rating <= 5 THEN
          SET rank ='Good';
     ELSE
          SET rank = 'fair';
     END IF;
     RETURN rank;
END;
$
```



Query:

```
select sid, sname, rating, getRankFromRating(rating)
as rank from sailor
```

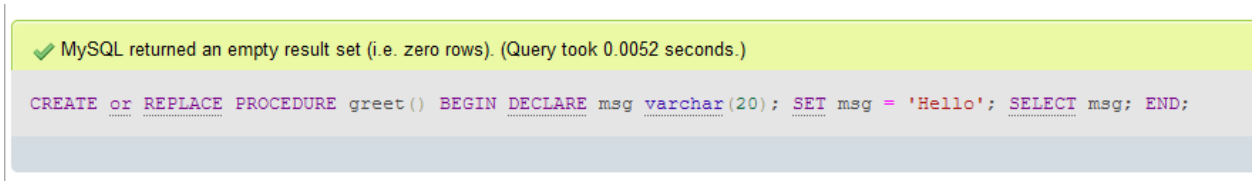


	sid	sname	rating	rank
Edit Copy Delete	1	Rahul	5	Good
Edit Copy Delete	2	Danyl	5	Good
Edit Copy Delete	3	Shrinath	5	Good
Edit Copy Delete	4	Jack	4	fair
Edit Copy Delete	5	Sean	3	fair
Edit Copy Delete	6	Matt	3	fair
Edit Copy Delete	7	Hans	4	fair
Edit Copy Delete	8	Drishyam	2	fair
Edit Copy Delete	9	Anakin	1	fair
Edit Copy Delete	10	Bob	3	fair

Procedures:

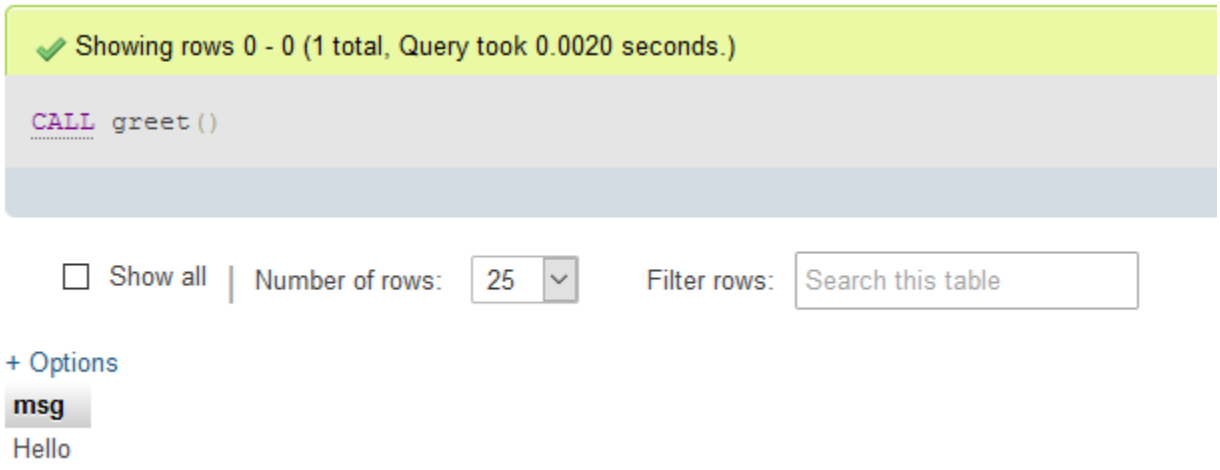
Query:

```
DELIMITER $
CREATE or REPLACE PROCEDURE greet()
BEGIN
     DECLARE msg varchar(20);
     SET msg = 'Hello';
SELECT msg;
END;
$
```



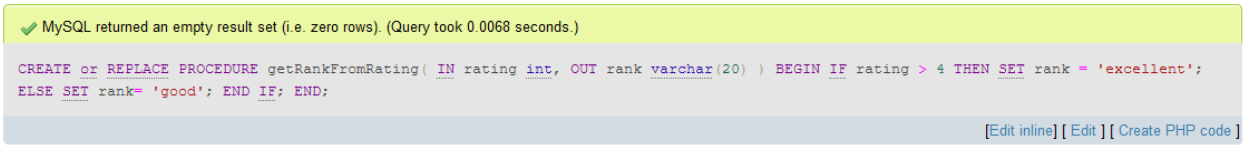
Query:

```
CALL greet();
```



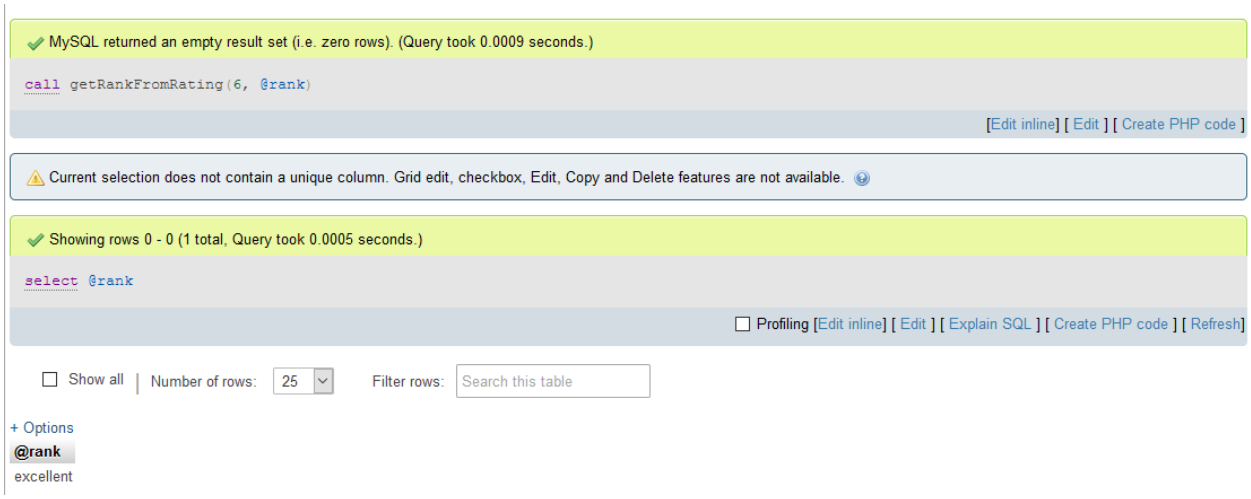
Query:

```
DELIMITER $
CREATE or REPLACE PROCEDURE getRankFromRating(
     IN rating int, OUT rank varchar(20)
)
BEGIN
DECLARE rank varchar(20);
IF rating > 4 THEN
     SET rank = 'excellent';
ELSE
SET rank= 'good';
END IF;
END;
$
```



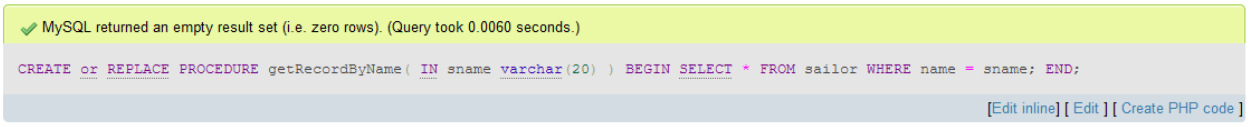
Query:

```
call getRankFromRating(6, @rank);
select @rank
```



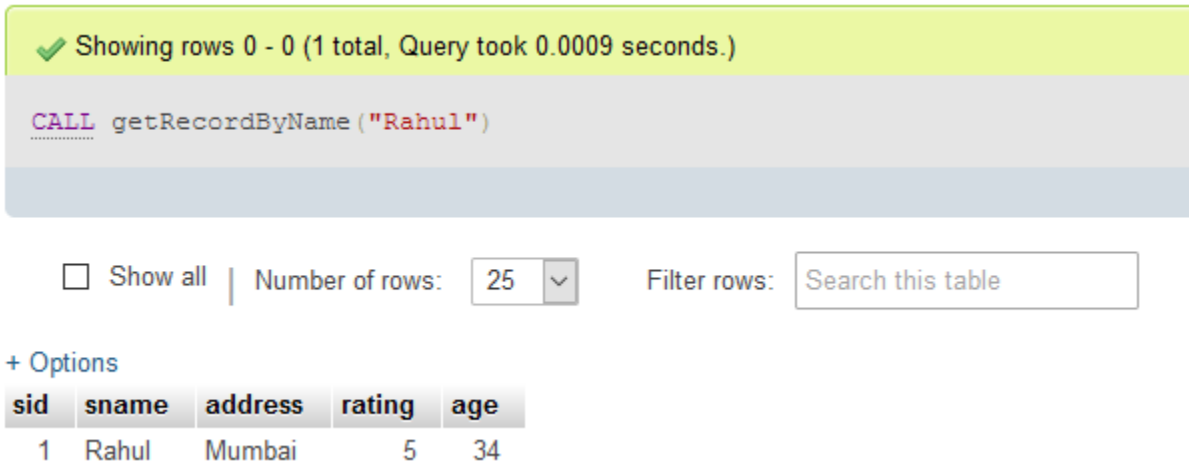
Query:

```
DELIMITER $
CREATE or REPLACE PROCEDURE getRecordByName(
     IN sname varchar(20)
)
BEGIN
     SELECT * FROM sailor WHERE name = sname;
END;
$
```



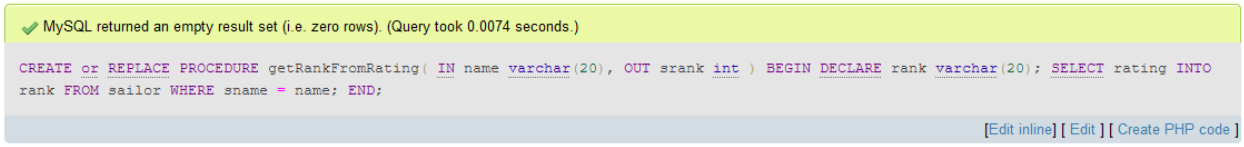
Query:

```
CALL getRecordByName("Rahul")
```



Query:

```
DELIMITER $
CREATE or REPLACE PROCEDURE getRankFromRating(
     IN name varchar(20), OUT srank int
)
BEGIN
     SELECT rating INTO srank FROM sailor WHERE sname = name;
END;
$
```



Query:

```
CALL getRankFromRating(“Shrinath”, @rank);
SELECT @rank
```