Relational Databases with MySQL Week 4 Coding Assignment

**Points possible:** 70

|  |  |  |
| --- | --- | --- |
| Category | Criteria | % of Grade |
| Functionality | Does the code work? | 25 |
| Organization | Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear. | 25 |
| Creativity | Student solved the problems presented in the assignment using creativity and out of the box thinking. | 25 |
| Completeness | All requirements of the assignment are complete. | 25 |

**Instructions:** Using a text editor of your choice, write the queries that accomplishes the objectives listed below. Take screenshots of the queries and results and paste them in this document where instructed below. Create a new repository on GitHub for this week’s assignments and push this document, with your Java project code, to the repository. Add the URL for this week’s repository to this document where instructed and submit this document to your instructor when complete.

**Coding Steps:**

Write 5 stored procedures for the employees database.

Write a description of what each stored procedure does and how to use it.

1. This stored procedure can help get the total salary for all current employees combined.

DELIMITER //

DROP PROCEDURE IF EXISTS get\_total\_salary;

CREATE PROCEDURE get\_total\_salary()

BEGIN

SET @salary\_total = (SELECT SUM(s.salary) from departments d

INNER JOIN dept\_emp de ON de.dept\_no = d.dept\_no

INNER JOIN salaries s ON s.emp\_no = de.emp\_no);

SET @current\_emp\_count = (SELECT COUNT(emp\_no) FROM dept\_emp WHERE to\_date > NOW());

SELECT @current\_emp\_count AS "Current Employees", FORMAT(@salary\_total, 'C') AS "Total Salary";

END//

CALL get\_total\_salary()//

DELIMITER ; //

1. This stored procedure can help publish the lowest salary and the highest salary earned among all salaries of the employees.

DELIMITER //

DROP PROCEDURE IF EXISTS p\_salaryrange;

CREATE PROCEDURE p\_salaryrange()

BEGIN

SELECT MIN(s.salary) AS 'Lowest Salary', MAX(s.salary) AS 'Highest Salary' from salaries s;

END//

CALL p\_salaryrange()//

DELIMITER ; //

1. This stored procedure can take an input parameter to query the employee database and return rows(20) from the table.

DELIMITER //

DROP PROCEDURE IF EXISTS p\_employeeList;

CREATE PROCEDURE p\_employeeList(

IN firstName VARCHAR(20)

)

BEGIN

SELECT \*

FROM employees

WHERE first\_name LIKE firstName

LIMIT 20;

END //

CALL p\_employeeList('g%')//

DELIMITER ; //

1. This stored procedure can take an input parameter to query the employee database and return result from the table as a parameter which can then be displayed.

DELIMITER // ;

DROP PROCEDURE IF EXISTS p\_employees\_by\_age;

CREATE PROCEDURE p\_employees\_by\_age(IN bornbefore DATE, OUT employees\_bornbefore FLOAT)

BEGIN

SELECT COUNT(\*) INTO employees\_bornbefore

FROM employees

WHERE birth\_date < bornbefore;

END //

DELIMITER ; //

CALL p\_employees\_by\_age('1960-01-01', @employees\_bornbefore);

Select @employees\_bornbefore;

1. This stored procedure uses BEGIN…END and a WHILE … END WHILE. It takes an input parameter to query the employee database and return result into a table that is created to show the results in a nice format.

DELIMITER ;

DROP TABLE IF EXISTS interim;

CREATE TABLE interim (

employee\_id INT (11),

employee\_fname varchar (45)

);

DELIMITER // ;

DROP PROCEDURE IF EXISTS p\_show\_ten\_employees;

CREATE PROCEDURE p\_show\_ten\_employees(IN input\_emp\_no INT)

BEGIN

DECLARE A INT;

DECLARE input\_emp\_fname CHAR(45);

DECLARE input\_emp\_lname CHAR(20);

SET A = 1;

WHILE A <=10 DO

SELECT CONCAT(first\_name, " ", last\_name)INTO input\_emp\_fname

FROM employees

WHERE emp\_no = input\_emp\_no;

SET A = A + 1;

INSERT INTO interim VALUES (input\_emp\_no, input\_emp\_fname);

SET input\_emp\_no = (input\_emp\_no + 1);

END WHILE;

SELECT employee\_id AS "Employee ID", employee\_fname AS "Employee Name" FROM interim;

END//

DELIMITER ; //

CALL p\_show\_ten\_employees(10050);

Procedures should use constructs you learned about from your research assignment and be more than just queries.

In Closing: Dropping all created stored procedures that were created for this assignment:

DROP PROCEDURE IF EXISTS get\_total\_salary;

DROP PROCEDURE IF EXISTS p\_salaryrange;

DROP PROCEDURE IF EXISTS p\_employeeList;

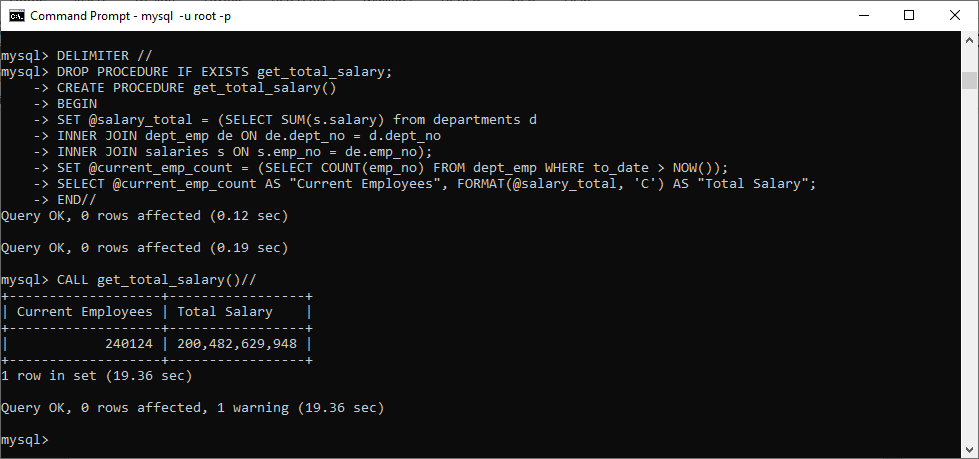
DROP PROCEDURE IF EXISTS p\_employees\_by\_age;

DROP PROCEDURE IF EXISTS p\_show\_ten\_employees;

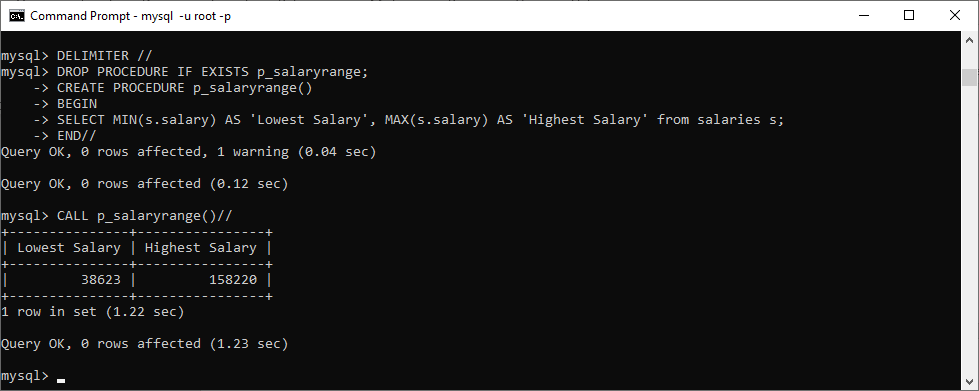
DROP TABLE IF EXISTS interim;

**Screenshots:**

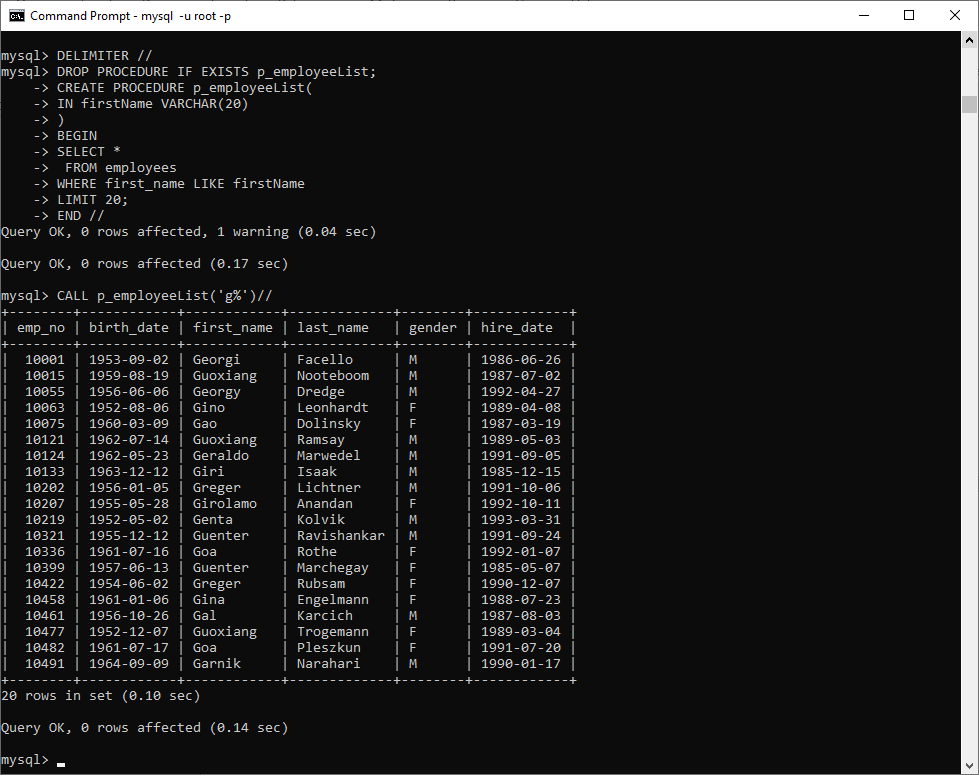
1. This stored procedure can help get the total salary for all current employees combined.



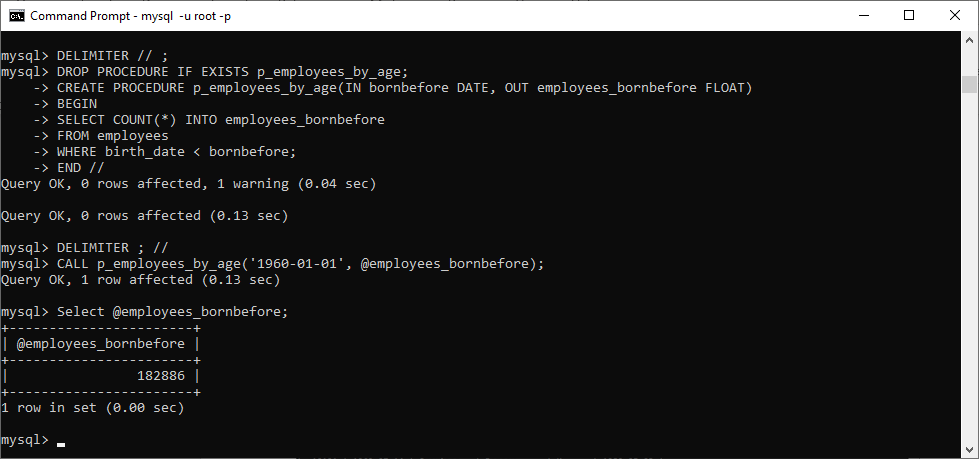
1. This stored procedure can help publish the lowest salary and the highest salary earned among all salaries of the employees.



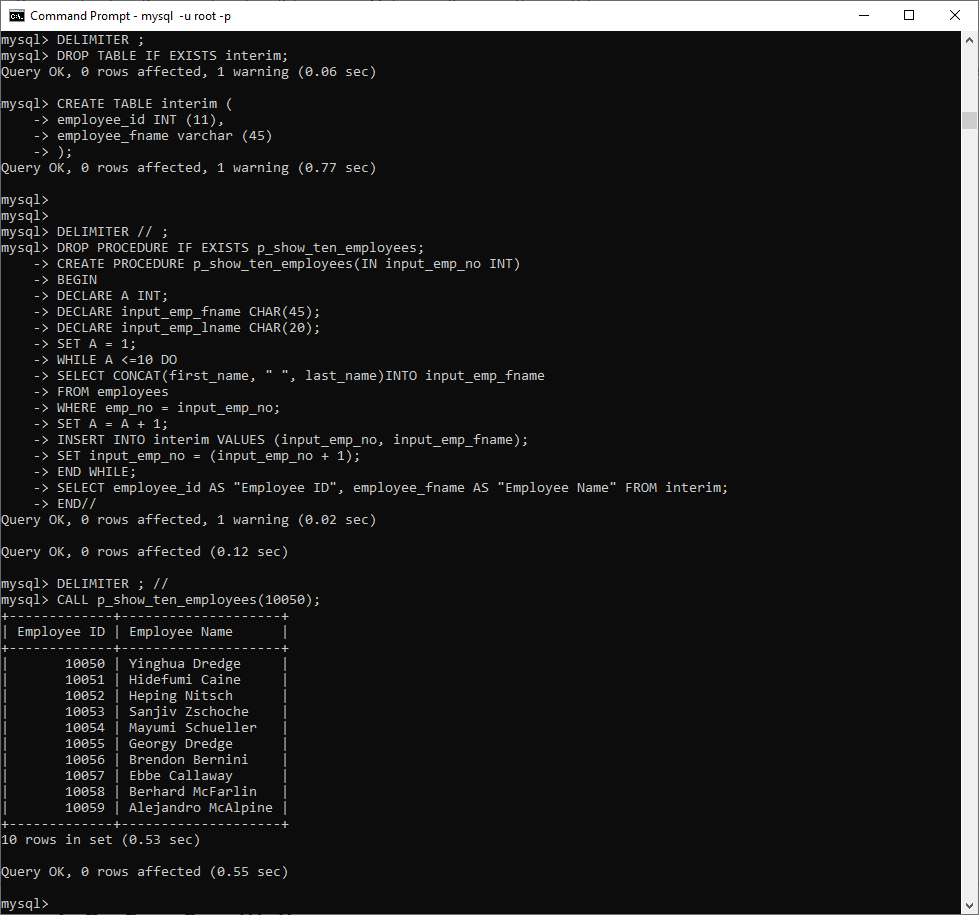
1. This stored procedure can take an input parameter to query the employee database and return rows (20) from the table.



1. This stored procedure can take an input parameter to query the employee database and return result from the table as a parameter which can then be displayed.



1. This stored procedure uses BEGIN…END and a WHILE … END WHILE. It takes an input parameter to query the employee database and return result into a table that is created to show the results in a nice format.



**URL to GitHub Repository:**

<https://github.com/srikripa/MSQL-Repos-W4.git>