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. Lastly, in the Learning Management System, click the “Add Submission” button and paste the URL to your GitHub repository.

**Coding Steps:**

Write 5 stored procedures for the employees database.

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

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

**Screenshots (stored procedures listed below):**

Purpose: displays how long an employee has been with the company.

DELIMITER $$

CREATE PROCEDURE years\_in\_job(

IN employee\_number int,

OUT years\_in\_job int

)

BEGIN

DECLARE start\_year int;

DECLARE end\_year int;

SELECT year (de.from\_date), year (de.to\_date)

INTO start\_year, end\_year

FROM dept\_emp de

WHERE de.emp\_no = employee.numer

LIMIT 1;

IF end\_year = 9999

THEN

SET end\_year = year (now());

END IF;

SELECT end\_year = start\_year INTO years\_in\_job;

END$$

CALL years\_in\_job;

Purpose: rates the average employee salary

DELIMITER $$

CREATE PROCEDURE SalaryScore(

IN amount INT

)

BEGIN

SELECT avg (\*)

INTO SalaryAverage

FROM salaries

WHERE salary = SalaryAverage;

IF SalaryAverage > 95000 THEN

SET salary\_rank = 10;

ELSE

SET salary\_rank = 1;

END IF;

END$$

CALL SalaryScore();

Purpose: determines the company rank based on the calculated total number of salaries of the given input

DELIMITER $$

CREATE PROCEDURE GetCountBySalaryAmount(

IN amount INT

)

BEGIN

SELECT count (\*)

INTO total

FROM salaries

WHERE salary = amount;

IF total > 70 THEN

SET company\_rank = 10;

ELSE

SET company\_rank = 100;

END IF;

END$$

CALL GetCountBySalaryAmount();

Purpose: takes the user entered ‘average salary’ and based on that information the query gives feedback

DELIMITER $$

CREATE PROCEDURE SalaryScore(

IN amount INT

)

BEGIN

SELECT avg (\*)

INTO SalaryAverage

FROM salaries

WHERE salary = SalaryAverage;

IF SalaryAverage > 100000 THEN

SELECT concat( ‘The salary average is ‘, SalaryAverage, ‘. That’s a pretty penny.’);

ELSE

SELECT concat( ‘the salary average is ‘, SalaryAverage, ‘.. I don’t know what to say about that .’);

END IF;

END$$

CALL SalaryScore();

Purpose: Karen counter

DELIMITER $$

CREATE PROCEDURE KarenCount()

BEGIN

SELECT count (\*)

INTO KarenCount

FROM employees

WHERE name = KarenCount;

IF KarenCount >= 1 THEN

SELECT concat( ‘There are‘, KarenCount, ‘ Karens.’);

ELSE

SELECT concat( ‘I couldn’t locate any Karens.’);

END IF;

END$$

CALL KarenCount;