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:

For this procedure we are inputting a date and we will be returned with a count of employees who have that date as their birth date. We use it by calling the procedure and entering a date in the parentheses, we can enter any date we want.

```
1   DELIMITER //
2
3 • CREATE PROCEDURE GetCountByBirthDate(
4   IN birthDate DATE
5   )
6   BEGIN
7   SELECT count(*) AS count
8   FROM employees
9   WHERE birth_date = birthDate;
10  END//
11
12  DELIMITER ;
```

```
2
3 • CALL GetCountByBirthDate('1961-01-01');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count
71

In this procedure we used an IN and OUT parameter where we take in a title name and are returned with the count of employees that fall under that title. The count is stored INTO the new variable we created in this case is called total. Once we call the procedure and it runs successfully we can SELECT the new variable name to see the count.

```
14  DELIMITER //
15  CREATE PROCEDURE GetCountByTitle(
16  IN title_name VARCHAR(50),
17  OUT total INTEGER
18  )
19  BEGIN
20  SELECT count(*)
21  INTO total
22  FROM titles
23  WHERE title = title_name;
24  END //
25
26  DELIMITER ;
27
```

```
4  CALL GetCountByTitle('Staff', @total);
5  SELECT @total;
```

ult Grid | Filter Rows: | Export: | Wrap Cell Content:

@total
107388

This procedure uses INOUT parameter to set an age demographic. It also uses an if then statement to see if the total count is greater than a number inserted in this case 10000. So, I entered the birth date of 1989-01-01 and if the count of employees born after that date is higher than 10000 then the age demographic is younger if else the age demographic will be set to older.

```
28   DELIMITER //
29
30 • CREATE PROCEDURE GetCompanyAgeDemo(
31   IN birthDate DATE,
32   OUT total INTEGER,
33   INOUT ageDemo VARCHAR (20)
34   )
35   BEGIN
36   SELECT count(*)
37   INTO total
38   FROM employees
39   WHERE birth_date > 1989-01-01;
40   IF total > 10000 THEN
41   SET ageDemo = 'younger';
42   ELSE
43   SET ageDemo = 'older';
44   END IF;
45   END//
46
47   DELIMITER ;
```

```
7 • CALL GetCompanyAgeDemo('1989-01-01', @total, @age_demo);
8 • SELECT @total;
9 • SELECT @age_demo;
```

ult Grid | Filter Rows: | Export: | Wrap Cell Content:

@age_demo
younger

This procedure calculates the employee count based on the gender entered. To call the procedure you enter M for male or F for female and a count will be returned.

```
1   DELIMITER //
2
3 • CREATE PROCEDURE GetCompanyGenderCount(
4   IN gender_type ENUM ('M','F'),
5   OUT total INTEGER
6   )
7   BEGIN
8   SELECT count(*)
9   INTO total
10  FROM employees
11  WHERE gender = gender_type;
12  END//
13
14  DELIMITER ;
```

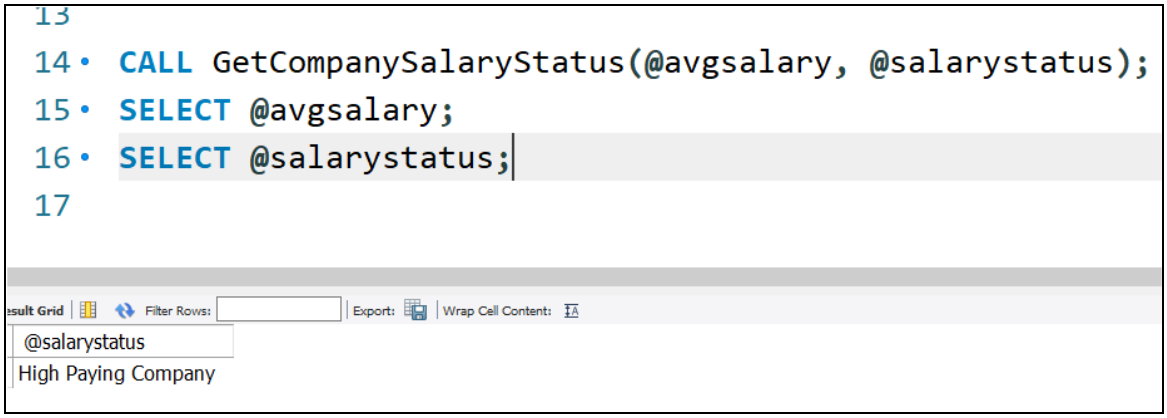
```
10
11 • CALL GetCompanyGenderCount('M', @total_gender_count);
12 • SELECT @total_gender_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

@total_gender_count
179970

This procedure calculates the average salary of all employees then classifies if the company is a high or low paying company based on the if then statement I added. In this case because the average salary is above 60000 and the number being compared to is 50000 it determined it was a low paying company once the procedure is called.

```
16  DELIMITER //
17
18 • CREATE PROCEDURE GetCompanySalaryStatus(
19  OUT avgsalary INTEGER,
20  INOUT salaryStatus VARCHAR (40)
21  )
22  BEGIN
23
24  SELECT avg(salary)
25  INTO avgsalary
26  FROM salaries;
27  IF avgsalary > 50000 THEN
28  SET salaryStatus = 'High Paying Company';
29  ELSE
30  SET salaryStatus = 'Low Paying Company';
31  END IF;
32  END//
33
34  DELIMITER ;
```



URL to GitHub Repository:

https://github.com/lcuevas6/week-4-sql-assigment.git