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.

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

Screenshots:

```
--
-- MySQL Week 4 Coding Assignment
-- Promineo Tech BESD Coding Bootcamp
--
--  Author:  Lisa Maatta Smith
--

USE employees;

--
-- Coding Assignment #4
-- Requirements:
--      1.  Write 5 stored procedures for the employees database.
--      2.  Write a description of what each stored procedure does and how to use it.
-- NOTE:  Procedures should use constructs you learned about from your
--        research assignment and be more than just queries.
--


--
-- MySQL Week 4 Coding Assignment
-- Procedure #1
-- Get the count of the employees in a particular department.
--          Input parameter:  department_name
--          Output Parameter:  num_of_emp

DROP PROCEDURE IF EXISTS GetEmpCountByDept;

DELIMITER %% ;
CREATE PROCEDURE GetEmpCountByDept(IN department_name VARCHAR(40), INOUT num_of_emp INTEGER)
BEGIN
    DECLARE dept_exists INTEGER DEFAULT 0;

    -- Check that department_name actually exists
    SELECT count(*)
    INTO dept_exists
    FROM departments d
    WHERE d.dept_name = department_name;

    -- If it exists in the departments table, then do a count
    -- Otherwise, just return.

    IF (dept_exists = 1)
    THEN
        SELECT count(*) INTO num_of_emp
        FROM employees e
        INNER JOIN dept_emp de USING (emp_no)
        INNER JOIN departments d USING (dept_no)
        GROUP BY d.dept_name HAVING d.dept_name = department_name;
    ELSE
        SET num_of_emp = 0;
    END IF;
END%%

DELIMITER ; %%
```

```
mysql>
mysql>
mysql> --
mysql> -- Test Procedure #1 with 2 Different Department Names
mysql> --
mysql> --   Procedure:  get_emp_count_by_dept()
mysql> --
mysql>
mysql>
mysql> --  "Marketing", and an output variable var_amount.
mysql> SET @var_name = "Marketing";
Query OK, 0 rows affected (0.00 sec)

mysql> CALL GetEmpCountByDept(@var_name, @var_amount);
Query OK, 1 row affected (0.38 sec)

mysql> SELECT @var_amount AS "Employees Count in",
    ->    @var_name AS "Department";
+--------------------+------------+
| Employees Count in | Department |
+--------------------+------------+
|              20212 | Marketing  |
+--------------------+------------+
1 row in set (0.00 sec)

mysql>
mysql> --  "Janitorial"
mysql> SET @var_name = "Janitorial";
Query OK, 0 rows affected (0.00 sec)

mysql> CALL GetEmpCountByDept(@var_name, @var_amount);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @var_amount AS "Employees Count in",
    ->    @var_name AS "Department";
+--------------------+------------+
| Employees Count in | Department |
+--------------------+------------+
|                  0 | Janitorial |
+--------------------+------------+
1 row in set (0.00 sec)

mysql>
```

```
-- Procedure #2  CalculateRaise()
-- Calculate Raise based on current salary
--   Input parameters:  emp_num, percentageRaise,
--   Output Parameter: newSalary

DROP PROCEDURE IF EXISTS CalculateRaise;

DELIMITER %% ;

CREATE PROCEDURE CalculateRaise (IN emp_num INT, IN percentRaise INT, OUT newSalary INT)
BEGIN
    DECLARE currentSalary DECIMAL(10,2);

    SELECT max(salary)
    INTO currentSalary
    FROM salaries
    WHERE emp_no = emp_num;

    SET newSalary = currentSalary + (currentSalary*(percentRaise/100));
END %%

DELIMITER ; %%
```

```
mysql>    mysql>
mysql> --
mysql> -- Test Procedure #2:  CalculateRaise();
mysql> --
mysql> --  Calculate the New Salary based on a percent and an employee's current max(salary).
mysql> --
mysql>
mysql> SELECT max(salary) FROM salaries
    -> WHERE emp_no = 10058;
+-------------+
| max(salary) |
+-------------+
|       72542 |
+-------------+
1 row in set (0.00 sec)

mysql>
mysql> CALL CalculateRaise (10058, 20, @currentSalary);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT @currentSalary AS "New Salary";
+------------+
| New Salary |
+------------+
|      87050 |
+------------+
1 row in set (0.00 sec)

mysql>
mysql>
```

```
-- Procedure #3a:  AddNewEmployee()
--
--  Add an employee to the employees database.
--   (1) Auto-increment the employee number, to avoid duplicates.
--       Retrieve the max emp_no value, and increment when inserting the new employee.
--   (2) Insert record into employees table with new_emp_no variable & input params.
--   (3) Insert record into dept_emp table with new_emp_no & dept_num input params.
--   (4)
--
--   Input parameter:  birth_date, first_name, last_name, gender, hire_date, dept_num,
--                     salary, title,
--   Output Parameter:  none
--   Local Variables:  max_emp_no -- the current maximum employee number
--                     new_emp_no -- max_emp_no incremented by one
--                     def_to_date -- set to "9999-01-01"
--                     def_from_date -- set to CURDATE()
--


DROP PROCEDURE IF EXISTS AddNewEmployee;

DELIMITER %% ;

CREATE PROCEDURE AddNewEmployee(
    IN birthdate DATE,
    IN f_name VARCHAR(14),
    IN l_name VARCHAR(16),
    IN gender_val ENUM('M','F'),
    IN hiredate DATE,
    IN dept_num CHAR(4),
    IN new_salary INTEGER,
    IN new_title VARCHAR(50),
    OUT error BOOLEAN)
BEGIN
    DECLARE emp_equal_count INT DEFAULT 0;
    DECLARE max_emp_no,new_emp_no INTEGER DEFAULT 0;
    DECLARE def_to_date DATE DEFAULT "9999-01-01";
    DECLARE def_from_date DATE DEFAULT CURDATE();
    SELECT max(emp_no) INTO max_emp_no FROM employees;
    SET new_emp_no = max_emp_no + 1;
    SET error = 0;

    SELECT count(*)
    INTO emp_equal_count
    FROM employees
    WHERE birth_date = birthdate AND first_name = f_name AND last_name = l_name;

    IF emp_equal_count = 0
    THEN
        INSERT INTO employees (emp_no, birth_date, first_name,last_name,gender,hire_date)
            VALUES (new_emp_no, birthdate, f_name, l_name, gender_val, hiredate);
        INSERT INTO salaries (emp_no, from_date, salary, to_date)
            VALUES (new_emp_no, def_from_date, new_salary, def_to_date);
        INSERT INTO titles (emp_no, title, from_date, to_date)
            VALUES (new_emp_no, new_title, def_from_date, def_to_date);
        INSERT INTO dept_emp (emp_no,dept_no,from_date,to_date)
            VALUES (new_emp_no,dept_num,def_from_date,def_to_date);
        SET ERROR = 1;
    ELSE
        SET ERROR = 0;
    END IF;
END%%

DELIMITER ; %%
```

```
-- Procedure #3b:  DeleteEmployee()
--
--  Delete an employee from the employees database.  Because of the cascading
--      deletes, we only need to delete the record from employees, and it will
--      be removed from all of the other tables.
--
--   Input parameter:  emp_num
--   Output Parameter:  error
--
--   Local Variables:  max_emp_no -- the current maximum employee number
--                     new_emp_no -- max_emp_no incremented by one
--                     def_to_date -- set to "9999-01-01"
--                     def_from_date -- set to CURDATE()
--

DROP PROCEDURE IF EXISTS DeleteEmployee;

DELIMITER %% ;
CREATE PROCEDURE DeleteEmployee (IN emp_num INT, OUT error BOOLEAN)
BEGIN
    DECLARE emp_exists INT DEFAULT 0;

    SELECT count(*)
    INTO emp_exists
    FROM employees
    WHERE emp_no = emp_num;

    IF (emp_exists = 1)
    THEN
        DELETE FROM employees WHERE emp_no = emp_num;
        SET error = 1;
    ELSE
        SET error = 0;
    END IF;
END%%

DELIMITER ; %%
```

```
mysql>
mysql>
mysql> --
mysql> -- Test Procedure #3:  AddNewEmployee() & DeleteEmployee()
mysql> --
mysql> --  Add a new employee to the employees database,
mysql> --  with records in:  employees, salaries, titles, and dept_emp tables
mysql> --  NOTE:  to avoid adding the same employee more than once, a check is done
mysql> --  to see if the employee already exists... and is not re-added if so.
mysql> --
mysql>
mysql>
mysql> SELECT max(emp_no) FROM employees;
+-------------+
| max(emp_no) |
+-------------+
|      499999 |
+-------------+
1 row in set (0.00 sec)

mysql>
mysql> --
mysql> -- Test AddEmployee(), inserting the following 6 employees!
mysql> --
mysql>
mysql> CALL AddNewEmployee("2000-01-01", "Mickey", "Mouse", 'M', "2020-01-01", "0005", 85000, "Engineer",@error);
CALL AddNewEmployee("2000-02-03", "Minnie", "Mouse", 'F', "2020-02-03", "0001", 86000, "Staff",@error);
CALL AddNewEmployee("2000-03-03", "Daffy", "Duck", 'M', "2020-03-03", "0002", 87000, "Staff",@error);
CALL AddNewEmployee("2000-04-03", "Daisy", "Duck", 'F', "2020-04-03", "0003", 88000, "Manager",@error);
CALL AddNewEmployee("2000-04-04", "Pluto", "Dog", 'M', "2020-04-04", "0004", 89000, "Technique Leader",@error);
CALL AddNewEmployee("2000-05-05", "Tinkerbell", "Fairy", 'F', "2020-05-05", "0005", 90000, "Senior Engineer",@error);

SELECT max(emp_no) FROM employees;

SELECT * FROM titles WHERE emp_no > 499999;
SELECT * FROM dept_emp WHERE emp_no > 499999;
SELECT * FROM employees WHERE emp_no > 499999;
SELECT * FROM salaries WHERE emp_no > 499999;
Query OK, 1 row affected (0.10 sec)

mysql> CALL AddNewEmployee("2000-02-03", "Minnie", "Mouse", 'F', "2020-02-03", "0001", 86000, "Staff",@error);
Query OK, 1 row affected (0.09 sec)

mysql> CALL AddNewEmployee("2000-03-03", "Daffy", "Duck", 'M', "2020-03-03", "0002", 87000, "Staff",@error);
Query OK, 1 row affected (0.06 sec)

mysql> CALL AddNewEmployee("2000-04-03", "Daisy", "Duck", 'F', "2020-04-03", "0003", 88000, "Manager",@error);
Query OK, 1 row affected (0.06 sec)

mysql> CALL AddNewEmployee("2000-04-04", "Pluto", "Dog", 'M', "2020-04-04", "0004", 89000, "Technique Leader",@error);
Query OK, 1 row affected (0.06 sec)

mysql> CALL AddNewEmployee("2000-05-05", "Tinkerbell", "Fairy", 'F', "2020-05-05", "0005", 90000, "Senior Engineer",@error);
Query OK, 1 row affected (0.06 sec)

mysql>
mysql> SELECT max(emp_no) FROM employees;
+-------------+
| max(emp_no) |
+-------------+
|      500005 |
+-------------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT * FROM titles WHERE emp_no > 499999;
+--------+-----------------+------------+------------+
| emp_no | title           | from_date  | to_date    |
+--------+-----------------+------------+------------+
| 500000 | Engineer        | 2021-01-07 | 9999-01-01 |
| 500001 | Staff           | 2021-01-07 | 9999-01-01 |
| 500002 | Staff           | 2021-01-07 | 9999-01-01 |
| 500003 | Manager         | 2021-01-07 | 9999-01-01 |
| 500004 | Technique Leader | 2021-01-07 | 9999-01-01 |
| 500005 | Senior Engineer | 2021-01-07 | 9999-01-01 |
+--------+-----------------+------------+------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM dept_emp WHERE emp_no > 499999;
+--------+---------+------------+------------+
| emp_no | dept_no | from_date  | to_date    |
+--------+---------+------------+------------+
| 500000 | d005    | 2021-01-07 | 9999-01-01 |
| 500001 | d001    | 2021-01-07 | 9999-01-01 |
| 500002 | d002    | 2021-01-07 | 9999-01-01 |
| 500003 | d003    | 2021-01-07 | 9999-01-01 |
| 500004 | d004    | 2021-01-07 | 9999-01-01 |
| 500005 | d005    | 2021-01-07 | 9999-01-01 |
+--------+---------+------------+------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
| 500005 | 2000-05-05 | Tinkerbell | Fairy     | F      | 2020-05-05 |
+--------+------------+------------+-----------+--------+------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM salaries WHERE emp_no > 499999;
```

```
mysql>
mysql>
mysql> --
mysql> -- Test DeleteEmployee() on emp_no = 500005
mysql> --
mysql>
mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
| 500005 | 2000-05-05 | Tinkerbell | Fairy     | F      | 2020-05-05 |
+--------+------------+------------+-----------+--------+------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM employees WHERE emp_no = 500005;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500005 | 2000-05-05 | Tinkerbell | Fairy     | F      | 2020-05-05 |
+--------+------------+------------+-----------+--------+------------+
1 row in set (0.00 sec)

mysql> CALL DeleteEmployee(500005,@error);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
+--------+------------+------------+-----------+--------+------------+
5 rows in set (0.00 sec)

mysql> SELECT @error AS "Delete Successful (1-TRUE, 0-FALSE)";
+-----------------------------------+
| Delete Successful (1-TRUE, 0-FALSE) |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT * FROM employees WHERE emp_no = 500005;
Empty set (0.00 sec)

mysql> CALL DeleteEmployee(500005,@error);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
+--------+------------+------------+-----------+--------+------------+
5 rows in set (0.00 sec)

mysql> SELECT @error AS "Delete Successful (1-TRUE, 0-FALSE)";
+-----------------------------------+
| Delete Successful (1-TRUE, 0-FALSE) |
+-----------------------------------+
|                                 0 |
+-----------------------------------+
1 row in set (0.00 sec)

mysql>
mysql>
```

```
-- Procedure #4:  SalaryPerCalendarYear()
--                This procedure calculates the calendar year salary total
--                for a particular employee in a particular department.
--
--   Input parameter:  emp_no, cal_year, department_name
--   Output Parameter:  pro_rated_salary
--

-- In the salary table, there is one record per year, per salary, per employee.
-- In the dept_emp, we can correlate which employees worked for which department
-- within a calendar year.
--
-- Since there is a record per year, per salary, per employee, in the salaries table
-- then the salary changes on the from_date to a new salary.

-- Salary in OUR employees database has a value which is in effect for one year,
--       starting at a random date in the year.
--
--      (e.g. emp_no 10058, salary = 53377, from_date = 1989-04-25, to_date = 1990-04-25
--        AND emp_no 10058, salary = 53869, from_date = 1990-04-25, to_date = 1991-04-25)
--
--      What if an employer wants to know calendar year pro-rating, or possibly
--      fiscal year pro-rating?
--             This procedure returns the calendar year pro-rating.


DROP PROCEDURE IF EXISTS SalaryPerCalendarYear;

DELIMITER %% ;

CREATE PROCEDURE SalaryPerCalendarYear (
    IN emp_no INT,
    IN cal_year INT,
    IN department_name VARCHAR(40),
    OUT pro_rated_salary DECIMAL(11,2))
READS SQL DATA
BEGIN
    DECLARE variable1 DECIMAL(11,2) DEFAULT 0.00;
    DECLARE variable2 DECIMAL(11,2) DEFAULT 0.00;
    SET pro_rated_salary = 0.00;

    -- Query 1:  Returns the first record -- for the first part of the salary year
    --           (FROM from_date to 12/31)
    SELECT IF ((EXTRACT(YEAR FROM s.from_date) = cal_year),
               (((s.salary/ DATEDIFF(s.to_date, s.from_date))*
               (DATEDIFF(s.to_date, s.from_date)- DAYOFYEAR(s.from_date)))), 0)
    INTO variable1
    FROM salaries s
    INNER JOIN dept_emp de ON (de.emp_no = s.emp_no)
    INNER JOIN departments d ON (d.dept_no = de.dept_no AND d.dept_name = department_name
                            AND (s.from_date BETWEEN de.from_date AND de.to_date))
    WHERE s.emp_no = emp_no AND (EXTRACT(YEAR FROM s.from_date) = 1990);


    -- Query 2:  Returns the second record -- for the second part of the salary year
    --           (1/1 to the to_date)
    SELECT IF ( (EXTRACT(YEAR FROM s.to_date) = cal_year),
               ((s.salary/ DATEDIFF(s.to_date, s.from_date))* DAYOFYEAR(s.to_date))), 0)
    INTO variable2
    FROM salaries s
    INNER JOIN dept_emp de ON (de.emp_no = s.emp_no)
    INNER JOIN departments d ON (d.dept_no = de.dept_no AND d.dept_name = department_name
                            AND (s.to_date BETWEEN de.from_date AND de.to_date))
    WHERE s.emp_no = emp_no AND (EXTRACT(YEAR FROM s.to_date) = 1990);

    SET pro_rated_salary = variable1 + variable2;

END %%

DELIMITER ; %%
```

```
-- Procedure #5:  UpdateEmploymentRecord()
--                Record that an employee changed departments (getting
--                hired from one department into another department.
--         (1)  to_date in dept_emp with old dept assigned to CURDATE();
--         (2)  add record to dept_emp table with new data:
--             (a) from_date = CURDATE()
--             (b) to_date = "9999-01-01"
--             (c) dept_no = new dept
--             (d) emp_no = emp_no
--   Input parameter: emp_no, old_dept, new_dept, new_start_date,
--   Output Parameter:  error -- If 0, nothing happened!
--

DROP PROCEDURE IF EXISTS UpdateEmploymentRecord;

DELIMITER %% ;
CREATE PROCEDURE UpdateEmploymentRecord(
    IN emp_num INT,
    IN old_dept CHAR(4),
    IN new_dept CHAR(4),
    IN effective_on DATE,
    OUT error INTEGER)
READS SQL DATA
BEGIN
    DECLARE emp_in_old_dept INTEGER DEFAULT 0;

    SELECT count(*)
    INTO emp_in_old_dept
    FROM dept_emp de
    INNER JOIN departments d ON de.dept_no = d.dept_no
    WHERE de.emp_no = emp_num;

    IF (emp_in_old_dept > 0)
    THEN
        UPDATE dept_emp de SET de.to_date = effective_on
        WHERE de.emp_no = emp_num;
        INSERT INTO dept_emp (emp_no,dept_no,from_date,to_date)
            VALUES (emp_num,new_dept,effective_on,"9999-01-01");
    ELSE
        SET emp_in_old_dept = 0;
    END IF;

    SET error = emp_in_old_dept;

END %%

DELIMITER ; %%

SHOW PROCEDURE status;
```

```
mysql>
mysql> --
mysql> -- Test Procedure #4: SalaryPerCalenderYear();
mysql> --
mysql> --  Figure out the salary for a particular calendar year,
mysql> --     pro-rating the salary by the days of the year in that salary.
mysql> --
mysql>
mysql>
mysql> --
mysql> -- Test Example 1:  Employee Id:  10058, Department:  "Marketing", Year:  1990
mysql> --
mysql>
mysql> SET @var_dept_name = "Marketing";
Query OK, 0 rows affected (0.00 sec)

mysql> SET @salary_per_year = 00000.00;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @emp_no = 10058;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @cal_year = 1990;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL SalaryPerCalendarYear(@emp_no,@cal_year,@var_dept_name,@salary_per_year);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT  @emp_no AS "Employee Id:", @cal_year AS "For the year:", @var_dept_name AS "Department Name:", FORMAT(@salary_per_year,2) AS "Pro-rated Salary:";
+------------+--------------+-------------------+--------------------+
| Employee Id: | For the year: | Department Name: | Pro-rated Salary: |
+------------+--------------+-------------------+--------------------+
|      10058 |         1990 | Marketing         | 53,713.99          |
+------------+--------------+-------------------+--------------------+
1 row in set (0.00 sec)

mysql>
mysql>
```

```
mysql>
mysql> --
mysql> -- Test Procedure #5: UpdateEmploymentRecord();
mysql> --
mysql> --  Change an employee from one department to new one, update old & insert new
mysql> --
mysql> --
mysql> -- Change one of these employees to a different department
mysql> --
mysql> --
mysql> -- Test Example #1:   employee #500000 -- Old Dept "d005" to New Dept "d001"
mysql> --
mysql>
mysql> CALL UpdateEmploymentRecord(500000,"d005","d001",CURDATE(),@error);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM dept_emp WHERE emp_no > 499999;
+--------+---------+------------+------------+
| emp_no | dept_no | from_date  | to_date    |
+--------+---------+------------+------------+
| 500000 | d001    | 2021-01-07 | 9999-01-01 |
| 500000 | d005    | 2021-01-07 | 2021-01-07 |
| 500001 | d001    | 2021-01-07 | 9999-01-01 |
| 500002 | d002    | 2021-01-07 | 9999-01-01 |
| 500003 | d003    | 2021-01-07 | 9999-01-01 |
| 500004 | d004    | 2021-01-07 | 9999-01-01 |
| 500005 | d005    | 2021-01-07 | 9999-01-01 |
+--------+---------+------------+------------+
7 rows in set (0.01 sec)

mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
| 500005 | 2000-05-05 | Tinkerbell | Fairy     | F      | 2020-05-05 |
+--------+------------+------------+-----------+--------+------------+
6 rows in set (0.00 sec)

mysql>
mysql> --
mysql> -- Test Example #2  emp_no = 500001, Old Dept "d001" to New Dept "d005"
mysql> --
mysql>
mysql> CALL UpdateEmploymentRecord(500001,"d001","d005",CURDATE(),@error);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM dept_emp WHERE emp_no > 499999;
+--------+---------+------------+------------+
| emp_no | dept_no | from_date  | to_date    |
+--------+---------+------------+------------+
| 500000 | d001    | 2021-01-07 | 9999-01-01 |
| 500000 | d005    | 2021-01-07 | 2021-01-07 |
| 500001 | d001    | 2021-01-07 | 2021-01-07 |
| 500001 | d005    | 2021-01-07 | 9999-01-01 |
| 500002 | d002    | 2021-01-07 | 9999-01-01 |
| 500003 | d003    | 2021-01-07 | 9999-01-01 |
| 500004 | d004    | 2021-01-07 | 9999-01-01 |
| 500005 | d005    | 2021-01-07 | 9999-01-01 |
+--------+---------+------------+------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM employees WHERE emp_no > 499999;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
| 500000 | 2000-01-01 | Mickey     | Mouse     | M      | 2020-01-01 |
| 500001 | 2000-02-03 | Minnie     | Mouse     | F      | 2020-02-03 |
| 500002 | 2000-03-03 | Daffy      | Duck      | M      | 2020-03-03 |
| 500003 | 2000-04-03 | Daisy      | Duck      | F      | 2020-04-03 |
| 500004 | 2000-04-04 | Pluto      | Dog       | M      | 2020-04-04 |
| 500005 | 2000-05-05 | Tinkerbell | Fairy     | F      | 2020-05-05 |
+--------+------------+------------+-----------+--------+------------+
6 rows in set (0.00 sec)

mysql>
```

```sql
--
-- Extra Procedure:  I wanted to test out using a WHILE loop
--                   CountOfEmployeesOverSalaryThreshold()
--
-- Input:   salaryThreshold INTEGER
-- Output:  countOfEmployees INTEGER
-- Purpose:  Find out how many employees make over a certain Threshold
--      I used 100000 in my test example.

DROP PROCEDURE IF EXISTS CountOfEmployeesOverSalaryThreshold;

DELIMITER %% ;

CREATE PROCEDURE CountOfEmployeesOverSalaryThreshold (IN salaryThreshold INTEGER, OUT countOfEmployees INTEGER)
BEGIN
    DECLARE runningTotal,max_emp_no,counter,tempSalary INTEGER DEFAULT 0;
    SELECT max(emp_no) INTO max_emp_no FROM employees;

    WHILE (counter < max_emp_no)
    DO
         SELECT max(salary)
         INTO tempSalary
         FROM salaries WHERE emp_no = counter;
         IF (tempSalary > salaryThreshold)
         THEN
            SET runningTotal = runningTotal + 1;
         END IF;
         SET counter = counter + 1;
         SET tempSalary = 0;
    END WHILE;
    SET countOfEmployees = runningTotal;
END%%

DELIMITER ; %%
```

```
mysql>
mysql>
mysql> --
mysql> -- Test Procedure #EXTRA: (CountOfEmployeesOverSalaryThreshold);
mysql> --
mysql> --  I wanted to try WHILE, and struggled to find a good reason to use it in
mysql> --  a stored procedure.  Here is what I came up with.
mysql> --
mysql>
mysql> CALL CountOfEmployeesOverSalaryThreshold (100000, @countresult);
Query OK, 1 row affected (20.69 sec)

mysql> SELECT @countresult;
+--------------+
| @countresult |
+--------------+
|        19017 |
+--------------+
1 row in set (0.00 sec)

mysql> --
mysql> -- To Test My result, I did the following, and the result is the same!
mysql> --
mysql>
mysql> DROP VIEW IF EXISTS employeesMaxSalaryOver100000;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
mysql> CREATE VIEW employeesMaxSalaryOver100000 AS
    -> SELECT emp_no, count(emp_no),max(salary) FROM salaries
    -> GROUP BY emp_no HAVING max(salary) > 100000;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> SELECT COUNT(*) AS "Final Count" FROM employeesMaxSalaryOver100000;
+-------------+
| Final Count |
+-------------+
|       19017 |
+-------------+
1 row in set (0.65 sec)

mysql>
mysql>
mysql> --
mysql> -- End of MySQL Week 4 Coding Assignment Tests
mysql> --
mysql>
mysql>
```

URL to GitHub Repository: https://github.com/sw-dev-lisa-s-nh/MySQL-week4.git