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.

1. Show the employee count for each department where employees were born between 1955 – 1960.

```
DROP PROCEDURE IF EXISTS EmpCountByDeptBornBetween1955To1958;
DELIMITER //

CREATE PROCEDURE EmpCountByDeptBornBetween1955To1958()
BEGIN
	DROP TABLE IF EXISTS empCountByDept;
	CREATE TEMPORARY TABLE empCountByDept
	AS (
		select distinct d.dept_name, year(e.birth_date) as birth_year, count(*)
EmployeeCountByDepartment
		from employees e
		join dept_emp de
		on e.emp_no  = de.emp_no
		join departments d
		on d.dept_no = de.dept_no
		where year(e.birth_date) > Year('1954-12-31') and Year(e.birth_date) <
Year(DATE_ADD('1954-12-31', INTERVAL +3 YEAR))
		group by d.dept_name, year(e.birth_date)
		order by year(e.birth_date), d.dept_name asc
	);

	select dept_name, sum(EmployeeCountByDepartment) as
EmpCountByDeptBornBetween1955To1958
	from empCountByDept
	group by dept_name;


END //

DELIMITER ;
```

To call the stored procedure –

```
call EmpCountByDeptBornBetween1955To1958();
```

2. Find out as to which title pays the most within a specified from_date and to_date

```
DROP PROCEDURE IF EXISTS TitleThatEarnedTheMostInAGivenPeriod;

DELIMITER //

CREATE PROCEDURE TitleThatEarnedTheMostInAGivenPeriod(
	IN from_date DATE,
	IN to_date DATE
)
BEGIN
	DROP TABLE IF EXISTS salaryByTitle;

	CREATE TEMPORARY TABLE salaryByTitle
	AS (
		select e.emp_no, t.title, format(avg(s.salary), 1) SalaryBytitle
		from employees e
		join salaries s
		on e.emp_no  = s.emp_no
		join titles t
		on t.emp_no = e.emp_no
		where year(t.from_date) = Year(from_date) and year(t.to_date) =
Year(to_date)
		group by e.emp_no, t.title
		order by t.title, SalaryBytitle asc
	);

	select title, SalaryBytitle as MostSalaryEarnedInAGivenPeriod
	from salaryByTitle
	order by SalaryBytitle desc limit 1;
END //
```

```
DELIMITER ;
```

To call the stored procedure –

```
call TitleThatEarnedTheMostInAGivenPeriod('1994-12-31', '1999-12-31');
```

3. Print out given number of employees along with their salaries

```
DROP PROCEDURE IF EXISTS PrintOutGivenNumberOfEmployeesWithSalaryInfo;

DELIMITER //

CREATE PROCEDURE PrintOutGivenNumberOfEmployeesWithSalaryInfo(
      IN counter INT
)

BEGIN
    Declare myCounter INT default 1;

    -- use the WHILE loop to increment myCounter.
    While counter > myCounter DO
      Set myCounter = myCounter + 1;
   END WHILE;

    DROP TABLE IF EXISTS myEmployeesTable;
    CREATE TEMPORARY TABLE myEmployeesTable
      AS (
            Select * from employees limit myCounter
    );

    DROP TABLE IF EXISTS mySalariesTable;

    -- Get the salaries of all of those employees who are in the mySalariesTable
table
    CREATE TEMPORARY TABLE mySalariesTable
      AS (
            Select emp_no, Avg(salary) as avgSalary
        from salaries where emp_no in (Select emp_no from myEmployeesTable)
        group by emp_no
    );

    DROP TABLE IF EXISTS myOutputTable;
    CREATE TABLE myOutputTable
      (
            emp_no INT,
        first_name varchar(14),
        last_name varchar(16),
        avgSalary float
      );

    SET  myCounter = 0;
    myLoop:  LOOP
         SET  myCounter = myCounter + 1;
         -- Insert one row at a time by joining the two tables such that emp_no is
        -- NOT present in the myOutputTable table.
        Insert into myOutputTable(emp_no, first_name, last_name, avgSalary)
             (Select e.emp_no, e.first_name, e.last_name, s.avgSalary
                    from myEmployeesTable e
             join mySalariesTable s
                    on e.emp_no = s.emp_no
             where e.emp_no not in (Select emp_no from myOutputTable) limit 1);

        IF  (counter > myCounter) THEN
                    ITERATE  myLoop;
             ELSE
                    LEAVE  myLoop;
       END  IF;
   END LOOP;

   Select * from myOutputTable;
END //

DELIMITER ;
```

To call the stored procedure –

```
call PrintOutGivenNumberOfEmployeesWithSalaryInfo(6);
```

4. Show the average salary earned by gender and by Title for a given hire_date

```
DROP PROCEDURE IF EXISTS AverageSalaryForEachTitleBasedOnGenderForASpecifiedHireYear;

DELIMITER //

CREATE PROCEDURE AverageSalaryForEachTitleBasedOnGenderForASpecifiedHireYear (IN
hireYear DATE)
BEGIN
       DROP TABLE IF EXISTS SumOfSalariesByGenderAndTitle;
       CREATE TEMPORARY TABLE SumOfSalariesByGenderAndTitle
       AS (
       Select gender, hire_date, title, sum(employeeAvgSalary) as
SumOfSalaryByGenderAndTitle
       from
             (select e.emp_no, e.gender, year(e.birth_date) as birthyear,
year(e.hire_date) as hire_date,
             t.title, avg(s.salary) as employeeAvgSalary
             from employees e
             join salaries s
             on e.emp_no  = s.emp_no
             join titles t
             on e.emp_no  = t.emp_no
             where year(e.hire_date) = Year(hireYear)
             group by e.emp_no, e.gender, year(e.birth_date), e.hire_date, t.title) a
       group by gender, hire_date, title
       order by title, gender
     );
```

```
	DROP TABLE IF EXISTS CountByGenderAndTitle;
	CREATE TEMPORARY TABLE CountByGenderAndTitle
	AS (
	 Select gender, hire_date, title, count(*) as employeeCountByGenderAndTitle
		from
			(select distinct e.emp_no, e.gender, year(e.hire_date) as
hire_date, t.title
			from employees e
			join salaries s
			on e.emp_no  = s.emp_no
			join titles t
			on e.emp_no  = t.emp_no
			where year(e.hire_date) = Year(hireYear)
			group by e.emp_no, e.gender, year(e.birth_date), e.hire_date,
t.title) a
		group by gender, hire_date, title
		order by title, gender
	);

	select t1.gender, t1.hire_date, t1.title,
format(t1.SumOfSalaryByGenderAndTitle/t2.employeeCountByGenderAndTitle, 1) as
avgSalaryByGenderAndTitle
	from SumOfSalariesByGenderAndTitle t1
	join CountByGenderAndTitle t2
	on t1.gender = t2.gender
	and t1.hire_date = t2.hire_date
	and t1.title = t2.title;
END //

DELIMITER ;
```

To call the stored procedure –

```
call AverageSalaryForEachTitleBasedOnGenderForASpecifiedHireYear ('1986-06-
02');
```

5. Compare two employees first names to find out as to which first name is longer in length.

```
DROP PROCEDURE IF EXISTS FirstNameLengthComparisonBetweenTwoEmployees;

DELIMITER //

CREATE PROCEDURE FirstNameLengthComparisonBetweenTwoEmployees(
IN emp_no1 INT,
IN emp_no2 INT,
OUT empNumber INT,
OUT fName varchar(25),
OUT length INT
)
BEGIN
	DROP TABLE IF EXISTS firstName1;
	CREATE TEMPORARY TABLE firstName1
	AS (
		select first_name, emp_no, length(first_name) as lengthFirstName
```

```
            from employees
            where emp_no = emp_no1
    );

    DROP TABLE IF EXISTS firstName2;
    CREATE TEMPORARY TABLE firstName2
    AS (
            select first_name, emp_no, length(first_name) as lengthFirstName
            from employees
            where emp_no = emp_no2
    );

    SET @len1 = (SELECT  lengthFirstName  from firstName1 );
    SET @len2 = (SELECT  lengthFirstName  from firstName2 );

    IF @len1 > @len2  THEN
            SELECT first_name,  lengthFirstName, emp_no
            into fName, length, empNumber from firstName1;
    ELSEIF @len2 > @len1  THEN
            SELECT first_name,  lengthFirstName, emp_no
            into fName, length, empNumber from firstName2;
    ELSE
            SELECT 'Same length First Name', lengthFirstName, 0000
            INTO fName, length, empNumber from firstName2;
   END IF;

END //

DELIMITER ;
```

To call the stored procedure –

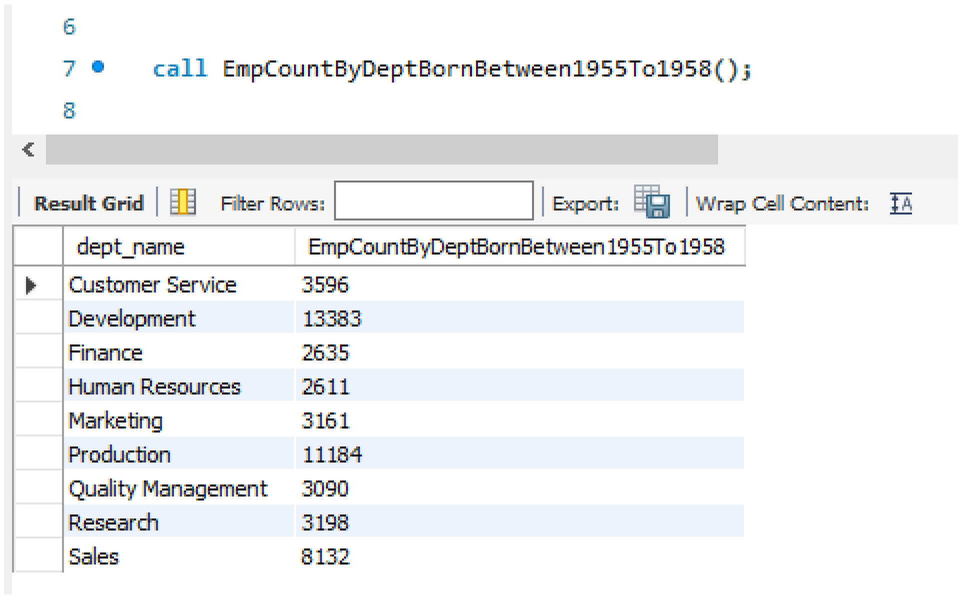
```
call FirstNameLengthComparisonBetweenTwoEmployees (10001, 10002, @empNumber,
@firstName, @length);

Select @empNumber, @firstName, @length;
```

Screenshots:

1. Show the employee count for each department where employees were born between 1955 – 1960.

```
call EmpCountByDeptBornBetween1955To1958();
```



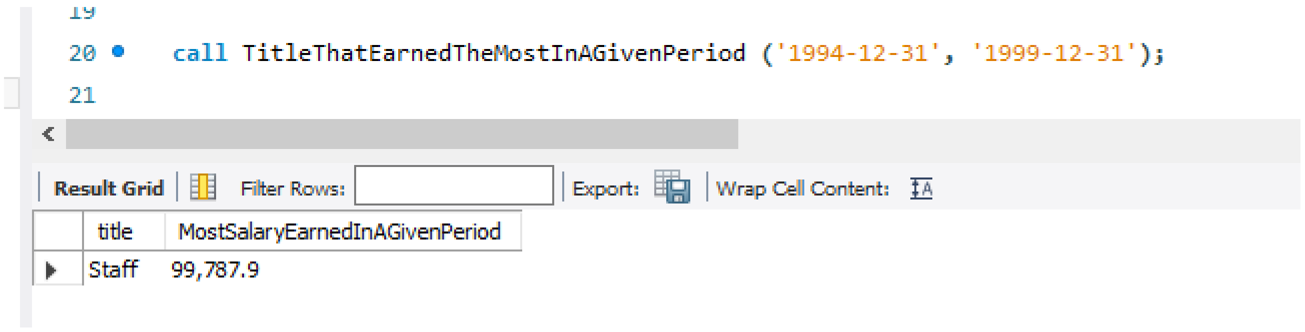
```
call EmpCountByDeptBornBetween1955To1958();
```

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

dept_name	EmpCountByDeptBornBetween1955To1958
Customer Service	3596
Development	13383
Finance	2635
Human Resources	2611
Marketing	3161
Production	11184
Quality Management	3090
Research	3198
Sales	8132

2. Find out as to which title pays the most within a specified from_date and to_date

```
call TitleThatEarnedTheMostInAGivenPeriod('1994-12-31', '1999-12-31');
```



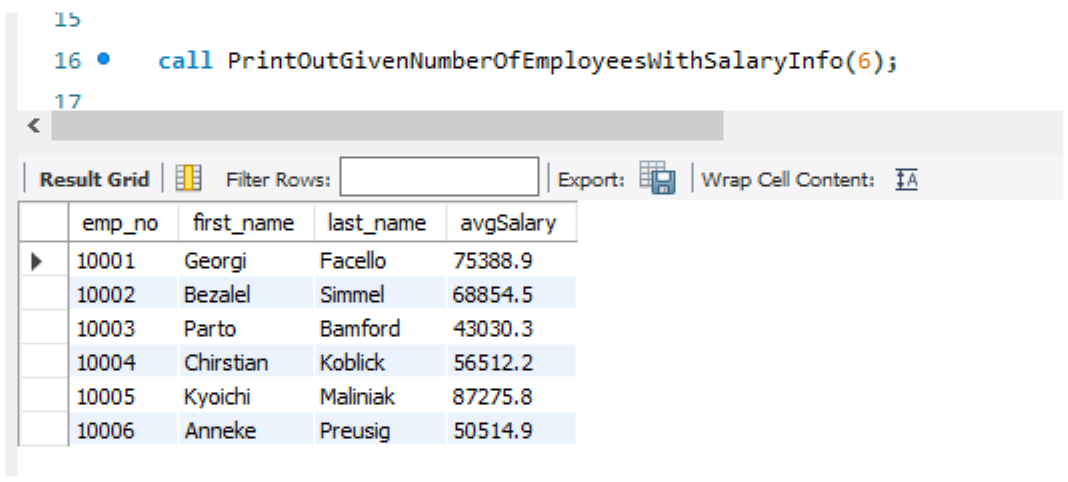
```
call TitleThatEarnedTheMostInAGivenPeriod ('1994-12-31', '1999-12-31');
```

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

title	MostSalaryEarnedInAGivenPeriod
Staff	99,787.9

3. Print out given number of employees along with their salaries

```
call PrintOutGivenNumberOfEmployeesWithSalaryInfo (6);
```



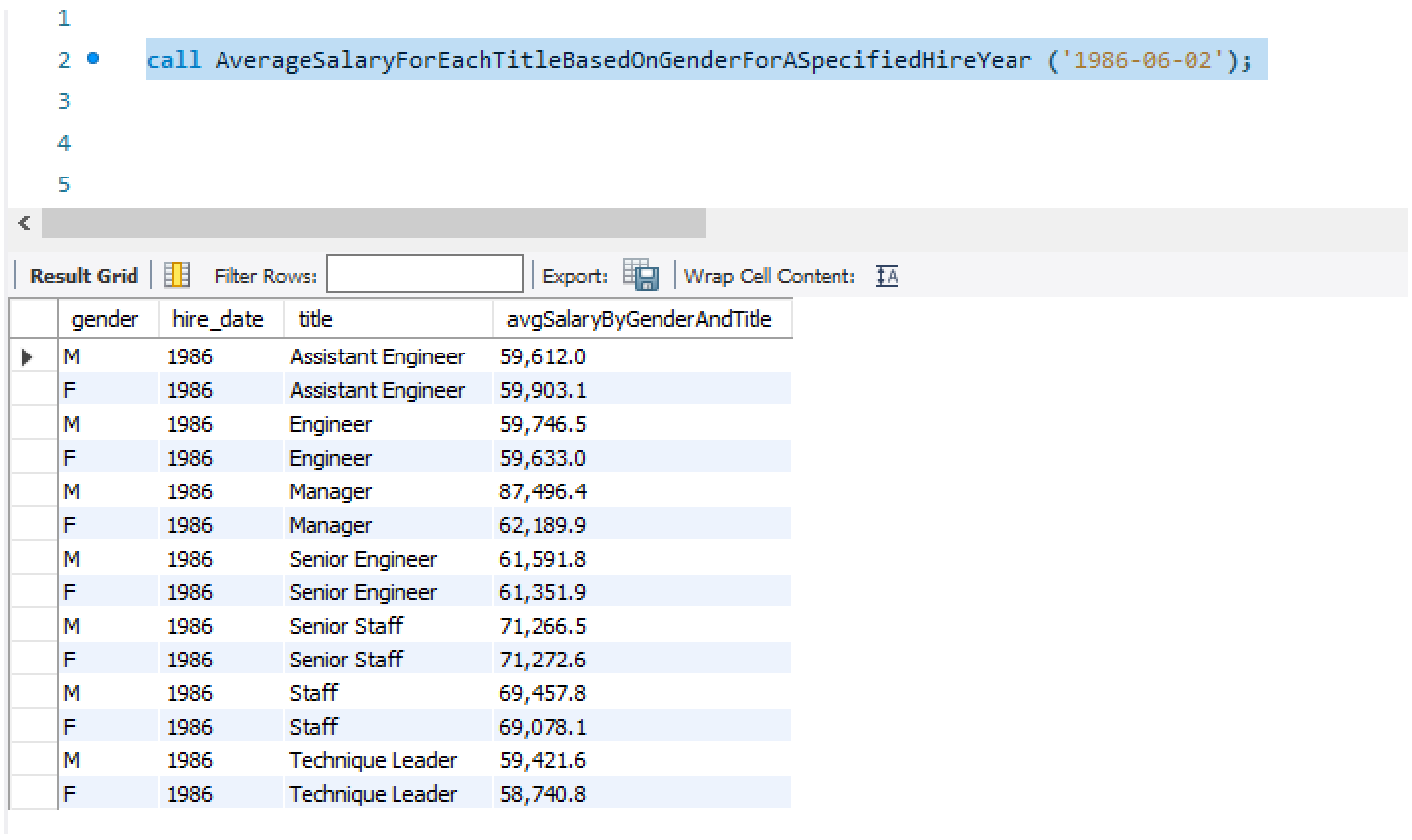
```
call PrintOutGivenNumberOfEmployeesWithSalaryInfo(6);
```

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

emp_no	first_name	last_name	avgSalary
10001	Georgi	Facello	75388.9
10002	Bezalel	Simmel	68854.5
10003	Parto	Bamford	43030.3
10004	Chirstian	Koblick	56512.2
10005	Kyoichi	Maliniak	87275.8
10006	Anneke	Preusig	50514.9

4. **Show the average salary earned by gender and by Title for a given hire_date**

```
call AverageSalaryForEachTitleBasedOnGenderForASpecifiedHireYear ('1986-06-
02');
```



```
call AverageSalaryForEachTitleBasedOnGenderForASpecifiedHireYear ('1986-06-02');
```

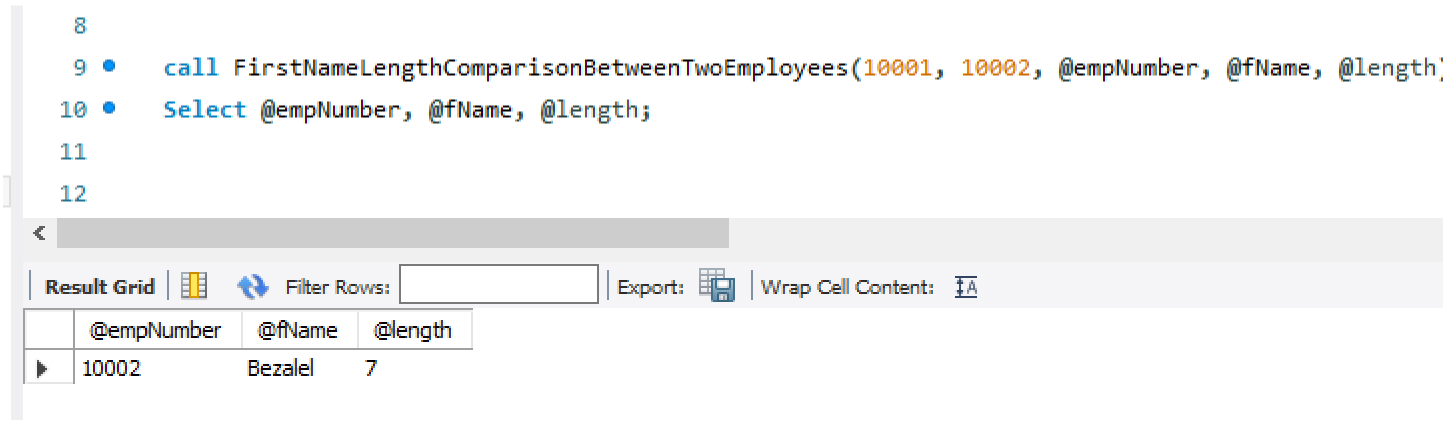
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender	hire_date	title	avgSalaryByGenderAndTitle
M	1986	Assistant Engineer	59,612.0
F	1986	Assistant Engineer	59,903.1
M	1986	Engineer	59,746.5
F	1986	Engineer	59,633.0
M	1986	Manager	87,496.4
F	1986	Manager	62,189.9
M	1986	Senior Engineer	61,591.8
F	1986	Senior Engineer	61,351.9
M	1986	Senior Staff	71,266.5
F	1986	Senior Staff	71,272.6
M	1986	Staff	69,457.8
F	1986	Staff	69,078.1
M	1986	Technique Leader	59,421.6
F	1986	Technique Leader	58,740.8

5. **Compare two employees first names to find out as to which first name is longer in length.**

```
call FirstNameLengthComparisonBetweenTwoEmployees (10001, 10002, @empNumber,
@firstName, @length);
```

```
Select @empNumber, @firstName, @length;
```



```
call FirstNameLengthComparisonBetweenTwoEmployees(10001, 10002, @empNumber, @fName, @length
Select @empNumber, @fName, @length;
```

@empNumber	@fName	@length
10002	Bezalel	7

URL to GitHub Repository: