

Relational Databases with MySQL Week 8 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: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) 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:

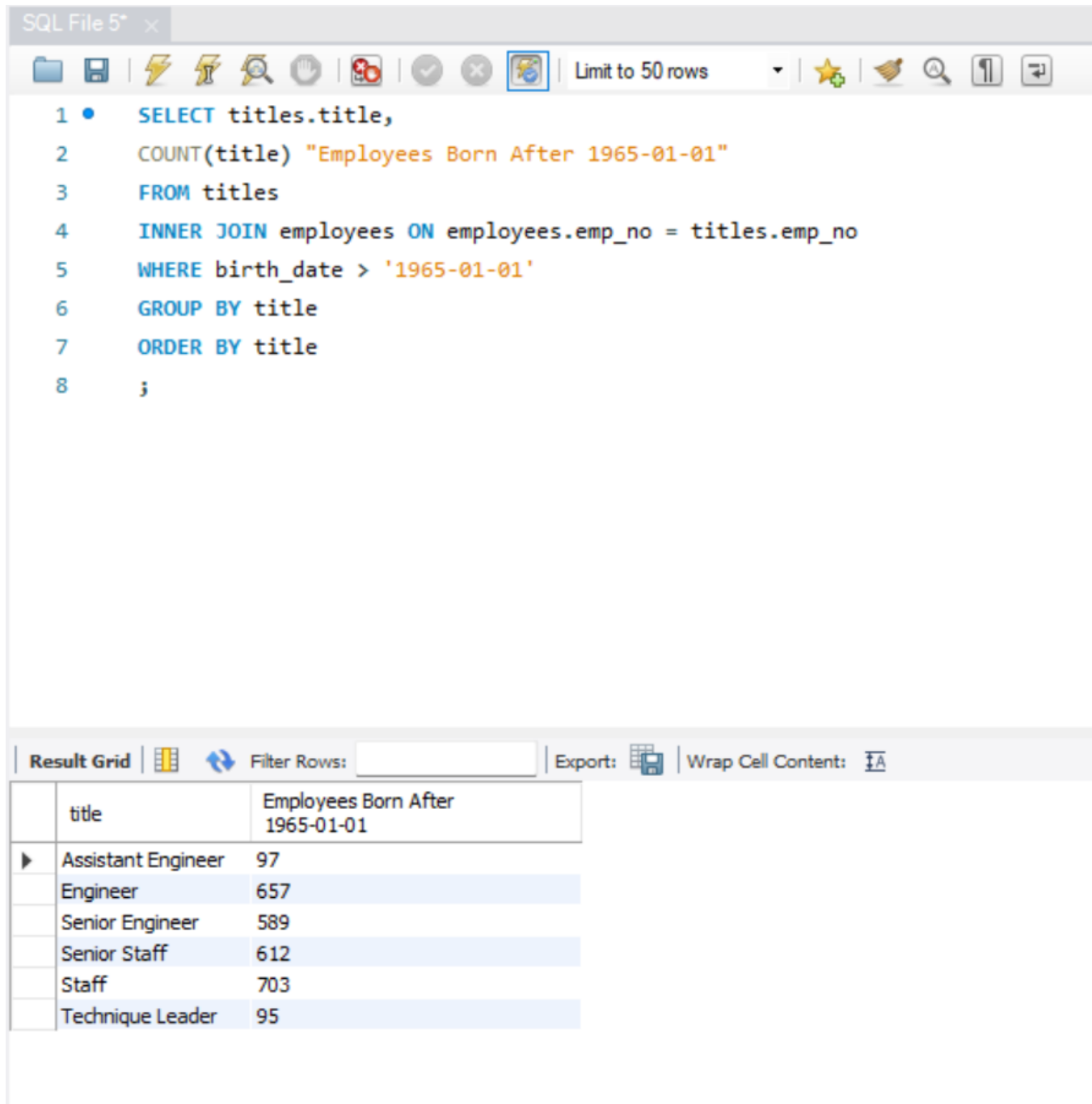
Write queries to address the following business needs.

1. I want to know how many employees with each title were born after 1965-01-01.
2. I want to know the average salary per title.
3. How much money was spent on salary for the marketing department between the years 1990 and 1992?

Screenshots of Queries with Query Results:

I want to know how many employees with each title were born after 1965-01-01.

(note that these query results do not include those born on 1965-01-01, as the request was for birth dates after 1965-01-01 and not on or after 1965-01-01)



The screenshot displays a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 50 rows' dropdown. The SQL editor contains the following query:

```
1 • SELECT titles.title,  
2     COUNT(title) "Employees Born After 1965-01-01"  
3     FROM titles  
4     INNER JOIN employees ON employees.emp_no = titles.emp_no  
5     WHERE birth_date > '1965-01-01'  
6     GROUP BY title  
7     ORDER BY title  
8     ;
```

Below the editor, the 'Result Grid' tab is active, showing the query results in a table. The table has two columns: 'title' and 'Employees Born After 1965-01-01'. The results are as follows:

title	Employees Born After 1965-01-01
Assistant Engineer	97
Engineer	657
Senior Engineer	589
Senior Staff	612
Staff	703
Technique Leader	95

I want to know the average salary per title.

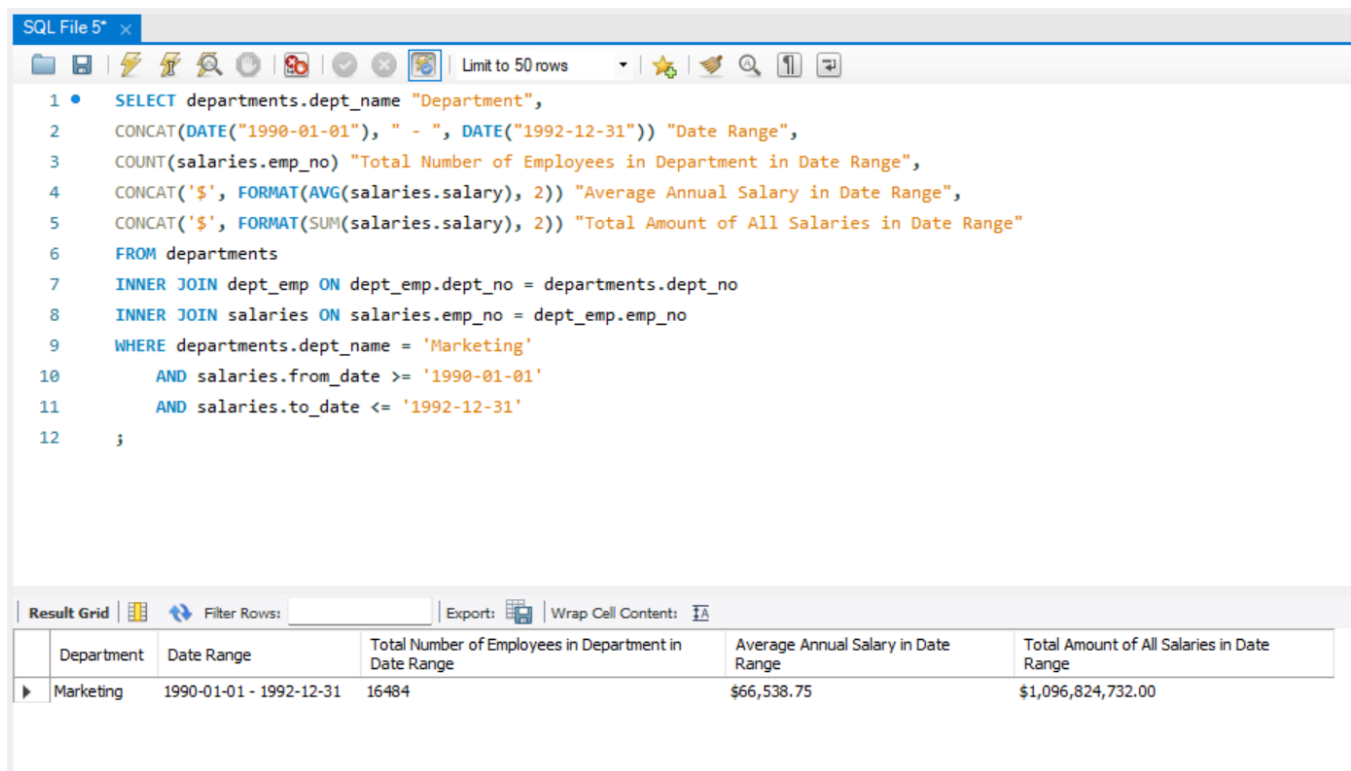
The screenshot shows a SQL IDE window titled "SQL File 5* x". The query editor contains the following SQL code:

```
1 • SELECT titles.title "Job Title",
2     CONCAT ('$ ', FORMAT(AVG(salary), 2)) "Average Employee Salary"
3     FROM salaries
4     INNER JOIN titles ON titles.emp_no = salaries.emp_no
5     GROUP BY title
6     ORDER BY title
7     ;
```

The results are displayed in a "Result Grid" below the query editor. The grid has two columns: "Job Title" and "Average Employee Salary". The results are as follows:

Job Title	Average Employee Salary
Assistant Engineer	\$59,304.99
Engineer	\$59,508.17
Manager	\$66,924.27
Senior Engineer	\$60,543.27
Senior Staff	\$70,470.29
Staff	\$69,308.47
Technique Leader	\$59,294.37

How much money was spent on salary for the marketing department between the years 1990 and 1992?



The screenshot shows an SQL IDE window titled "SQL File 5*". The query editor contains the following SQL code:

```
1 • SELECT departments.dept_name "Department",
2   CONCAT(DATE("1990-01-01"), " - ", DATE("1992-12-31")) "Date Range",
3   COUNT(salaries.emp_no) "Total Number of Employees in Department in Date Range",
4   CONCAT('$', FORMAT(AVG(salaries.salary), 2)) "Average Annual Salary in Date Range",
5   CONCAT('$', FORMAT(SUM(salaries.salary), 2)) "Total Amount of All Salaries in Date Range"
6 FROM departments
7 INNER JOIN dept_emp ON dept_emp.dept_no = departments.dept_no
8 INNER JOIN salaries ON salaries.emp_no = dept_emp.emp_no
9 WHERE departments.dept_name = 'Marketing'
10    AND salaries.from_date >= '1990-01-01'
11    AND salaries.to_date <= '1992-12-31'
12 ;
```

Below the query editor, the "Result Grid" tab is active, displaying the results of the query in a table. The table has five columns: Department, Date Range, Total Number of Employees in Department in Date Range, Average Annual Salary in Date Range, and Total Amount of All Salaries in Date Range. The results show data for the Marketing department for the date range 1990-01-01 to 1992-12-31.

Department	Date Range	Total Number of Employees in Department in Date Range	Average Annual Salary in Date Range	Total Amount of All Salaries in Date Range
Marketing	1990-01-01 - 1992-12-31	16484	\$66,538.75	\$1,096,824,732.00

URL to GitHub Repository:

<https://github.com/jprengaman/BESD-Week-8>