

# Aggregate Functions and Grouping in SQL

---

*Submitted by: Shalini T*

---

## Introduction

Aggregate functions and grouping in SQL are essential tools for summarizing and analyzing large sets of data. They allow us to perform calculations on multiple rows of a table's column and return a single summarized value. Grouping, on the other hand, organizes data into meaningful sets, enabling deeper analysis and insights.

## Objective of the Task

The objective of this task is to understand and practice the use of aggregate functions such as COUNT, SUM, AVG, MAX, MIN, along with the GROUP BY and HAVING clauses to summarize and categorize data effectively.

## Tools Used

- MySQL Workbench for executing SQL queries
- Company Database containing the Employees table

## Dataset Overview

The dataset used for this task is the Employees table with the following structure:

Column Name	Data Type	Description
<b>emp_id</b>	INT	Employee ID (Primary Key)
<b>name</b>	VARCHAR(50)	Employee Name
<b>department</b>	VARCHAR(50)	Department Name
<b>salary</b>	DECIMAL(10,2)	Employee Salary
<b>email</b>	VARCHAR(100)	Unique Email Address
<b>joining_date</b>	DATE	Date of Joining

## Aggregate Functions in SQL

Aggregate functions in SQL perform a calculation on a set of values and return a single value. Some of the commonly used aggregate functions are:

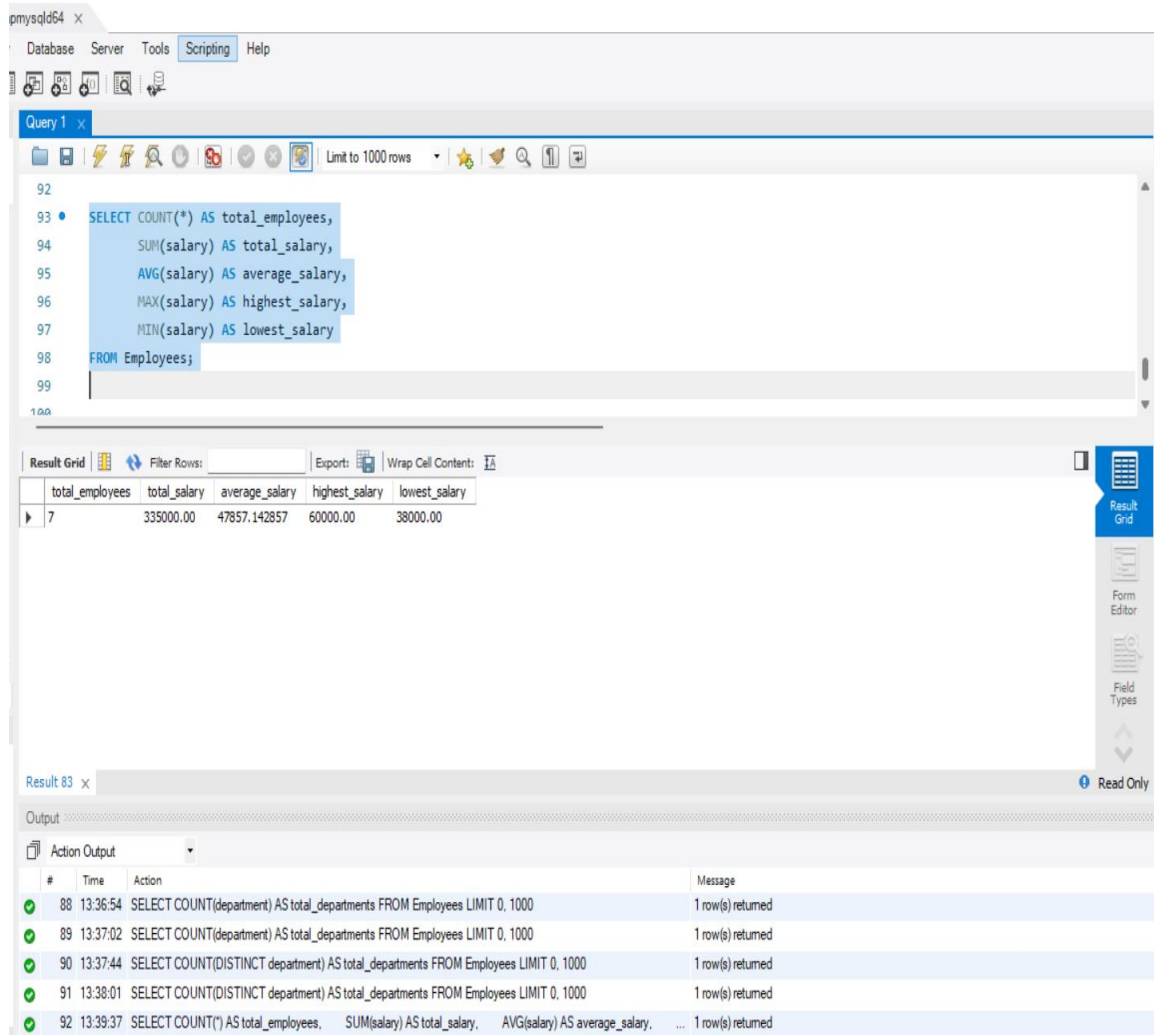
COUNT: Counts the number of rows.

SUM: Calculates the total sum of a numeric column.

AVG: Returns the average value of a numeric column.

MAX: Finds the highest value in a column.

MIN: Finds the lowest value in a column.



The screenshot shows a SQL IDE window titled 'pmysqld64'. The 'Scripting' tab is active, displaying a SQL query in the 'Query 1' editor. The query is as follows:

```
92  
93 • SELECT COUNT(*) AS total_employees,  
94       SUM(salary) AS total_salary,  
95       AVG(salary) AS average_salary,  
96       MAX(salary) AS highest_salary,  
97       MIN(salary) AS lowest_salary  
98 FROM Employees;  
99
```

Below the query editor, the 'Result Grid' is visible, showing a single row of results:

	total_employees	total_salary	average_salary	highest_salary	lowest_salary
7	7	335000.00	47857.142857	60000.00	38000.00

At the bottom, the 'Output' pane shows a log of actions and messages:

#	Time	Action	Message
88	13:36:54	SELECT COUNT(department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
89	13:37:02	SELECT COUNT(department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
90	13:37:44	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
91	13:38:01	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
92	13:39:37	SELECT COUNT(*) AS total_employees, SUM(salary) AS total_salary, AVG(salary) AS average_salary, ...	1 row(s) returned

## GROUP BY Clause

The GROUP BY clause groups rows that have the same values in specified columns into aggregated data. It is often used with aggregate functions to produce summarized results.

The screenshot shows the npmysql64 application interface. The main window displays a SQL query in the editor:

```

47
48
49 • SELECT department, COUNT(emp_id)
50 FROM Employees
51 GROUP BY department;
52
53
54
--

```

Below the query editor, the 'Result Grid' is visible, showing the results of the query:

department	COUNT(emp_id)
HR	2
Finance	2
IT	2
Sales	1

On the right side of the application, a help message is displayed:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

At the bottom, the 'Output' pane shows a log of actions and their results:

#	Time	Action	Message	Duration / Fetch
6	11:02:49	SELECT COUNT(*) FROM Employees LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
7	11:05:27	select * from employess LIMIT 0, 1000	Error Code: 1146. Table 'employee_management.emplo...	0.000 sec
8	11:05:49	select * from employees LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
9	11:12:13	SELECT department, COUNT(*) FROM Employees G...	4 row(s) returned	0.000 sec / 0.000 sec
10	11:12:41	SELECT department, COUNT(emp_id) FROM Employee...	4 row(s) returned	0.000 sec / 0.000 sec

## HAVING Clause

The HAVING clause is used to filter records after the GROUP BY operation has been applied. It is similar to the WHERE clause, but WHERE filters rows before grouping, whereas HAVING filters after grouping.

The screenshot shows the MySQL Workbench interface with the following components:

- Query Editor:** Contains two SQL queries. The first query is `select * from employees order by salary desc limit 1;`. The second query is `select department,sum(salary) from employees group by department having sum(salary)>90000;`.
- Result Grid:** Displays the results of the second query. It shows two rows: Finance with a sum(salary) of 97000.00, and IT with a sum(salary) of 115000.00.
- Output Panel:** Shows the execution log with the following entries:
 

#	Time	Action	Message	Duration / Fetch
51	11:19:33	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP...	0.000 sec
52	11:19:38	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP...	0.000 sec
53	11:20:10	select max(salary) from employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
54	13:18:49	select * from employees order by salary desc limit 1	1 row(s) returned	0.078 sec / 0.000 sec
55	13:22:58	select department,sum(salary) from employees group by...	2 row(s) returned	0.000 sec / 0.000 sec

## Count Total Employees

**SELECT COUNT(\*) AS total\_employees FROM Employees;**

Counts the total number of employees in the company.

# Aggregate Functions & Grouping in SQL

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL commands:

```
37
38
39 • SHOW TABLES;
40 • SELECT COUNT(*) FROM Employees;
41
42 • SELECT DISTINCT department FROM Employees;
43
44 • SELECT COUNT(*) FROM Employees;
```

The result grid shows the output of the last query:

COUNT(*)
7

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
2	10:58:32	use employee_management	0 row(s) affected	0.000 sec
3	10:58:37	show tables	3 row(s) returned	0.015 sec / 0.000 sec
4	10:58:47	desc employees	6 row(s) returned	0.000 sec / 0.016 sec
5	10:58:54	select * from employees LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
6	11:02:49	SELECT COUNT(*) FROM Employees LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec

## Employees Per Department

**SELECT department, COUNT(\*) AS total\_employees FROM Employees GROUP BY department;**

Shows how many employees are there in each department.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL commands:

```
68 • SELECT department, COUNT(*) AS total_employees
69 FROM Employees
70 GROUP BY department;
71
72
73 • SELECT department, joining_date, COUNT(*) AS total_employees
74 FROM Employees
75 GROUP BY department, joining_date;
```

The result grid shows the output of the first query:

department	total_employees
HR	2
Finance	2
IT	2
Sales	1

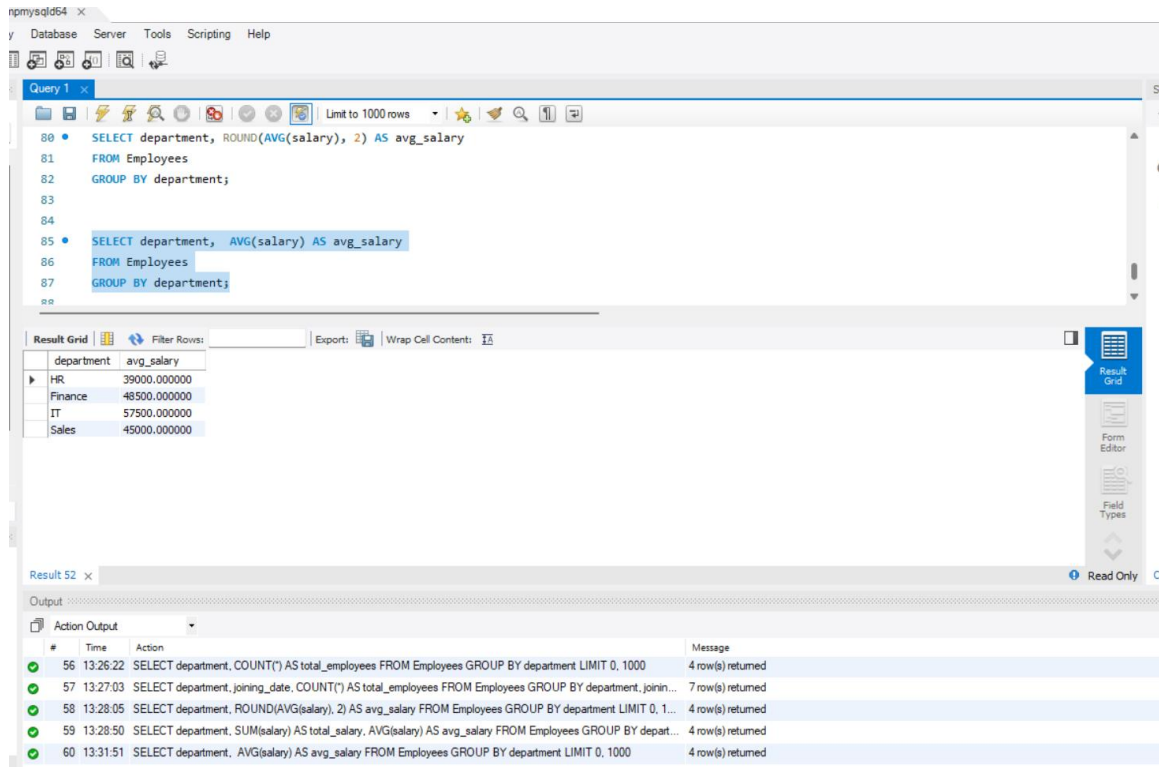
The output pane shows the execution log:

#	Time	Action	Message
90	13:37:44	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
91	13:38:01	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
92	13:39:37	SELECT COUNT(*) AS total_employees, SUM(salary) AS total_salary, AVG(salary) AS average_salary, ...	1 row(s) returned
93	16:02:53	select department,sum(salary) from employees group by department LIMIT 0, 1000	4 row(s) returned
94	16:07:08	SELECT department, COUNT(*) AS total_employees FROM Employees GROUP BY department LIMIT 0, 1000	4 row(s) returned

## Average Salary by Department

**SELECT department, AVG(salary) AS avg\_salary FROM Employees GROUP BY department;**

Calculates the average salary for each department.



The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```

80 • SELECT department, ROUND(AVG(salary), 2) AS avg_salary
81 FROM Employees
82 GROUP BY department;
83
84
85 • SELECT department, AVG(salary) AS avg_salary
86 FROM Employees
87 GROUP BY department;
88

```

The results pane displays the output of the query, showing the average salary for each department:

department	avg_salary
HR	39000.000000
Finance	48500.000000
IT	57500.000000
Sales	45000.000000

Below the results pane, the 'Action Output' pane shows the execution log:

#	Time	Action	Message
56	13:26:22	SELECT department, COUNT(*) AS total_employees FROM Employees GROUP BY department LIMIT 0, 1000	4 row(s) returned
57	13:27:03	SELECT department, joining_date, COUNT(*) AS total_employees FROM Employees GROUP BY department, joinin...	7 row(s) returned
58	13:28:05	SELECT department, ROUND(AVG(salary), 2) AS avg_salary FROM Employees GROUP BY department LIMIT 0, 1...	4 row(s) returned
59	13:28:50	SELECT department, SUM(salary) AS total_salary, AVG(salary) AS avg_salary FROM Employees GROUP BY depart...	4 row(s) returned
60	13:31:51	SELECT department, AVG(salary) AS avg_salary FROM Employees GROUP BY department LIMIT 0, 1000	4 row(s) returned

## Departments with More Than One Employee

**SELECT department, COUNT(\*) AS total FROM Employees GROUP BY department HAVING total > 1;**

Displays only those departments having more than one employee.

## Highest Salary Employee

**SELECT name, salary FROM Employees WHERE salary = (SELECT MAX(salary) FROM Employees);**

Finds the employee(s) with the highest salary.

pmysqld64 x

Database Server Tools Scripting Help

Query 1 x

SQLAdditions

Limit to 1000 rows

Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

53
54 • select max(salary)
55   from employees;
56
57
58 • select * from employees
59   order by salary desc
60   limit 1;

```

Result Grid

	emp_id	name	department	salary	email	joining_date
▶	3	Bob Johnson	IT	60000.00	bob@example.com	2024-05-01
•	NULL	NULL	NULL	NULL	NULL	NULL

Form Editor

Field Types

employees 46 x

Apply Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
50	11:19:26	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP...	0.000 sec
51	11:19:33	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP...	0.000 sec
52	11:19:38	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP...	0.000 sec
53	11:20:10	select max(salary) from employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
54	13:18:49	select * from employees order by salary desc limit 1	1 row(s) returned	0.078 sec / 0.000 sec

### Total Salary by Department

**SELECT department, SUM(salary) FROM Employees GROUP BY department;**

Calculates the total salary expenditure for each department.



# Aggregate Functions & Grouping in SQL

The screenshot shows the SQL Developer interface with a query window titled 'Query 1'. The query is as follows:

```
101 • SELECT COUNT(department) AS total_departments
102 FROM Employees;
103
104
105 • select department,sum(salary)
106 from employees
107 group by department;
108
```

The 'Result Grid' shows the results of the second query:

department	sum(salary)
HR	78000.00
Finance	97000.00
IT	115000.00
Sales	45000.00

The 'Output' window shows the execution log with the following messages:

#	Time	Action	Message
89	13:37:02	SELECT COUNT(department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
90	13:37:44	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
91	13:38:01	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
92	13:39:37	SELECT COUNT(*) AS total_employees, SUM(salary) AS total_salary, AVG(salary) AS average_salary, ...	1 row(s) returned
93	16:02:53	select department,sum(salary) from employees group by department LIMIT 0, 1000	4 row(s) returned

## MAX()

- \*\*Query to find highest salary:\*\*

**SELECT name, salary FROM Employees WHERE salary = (SELECT MAX(salary) FROM Employees);**

The screenshot shows the SQL Developer interface with a query window titled 'Query 1'. The query is as follows:

```
50 FROM Employees
51 GROUP BY department;
52
53
54 • select max(salary)
55 from employees;
56
57
```

The 'Result Grid' shows the results of the second query:

max(salary)
60000.00

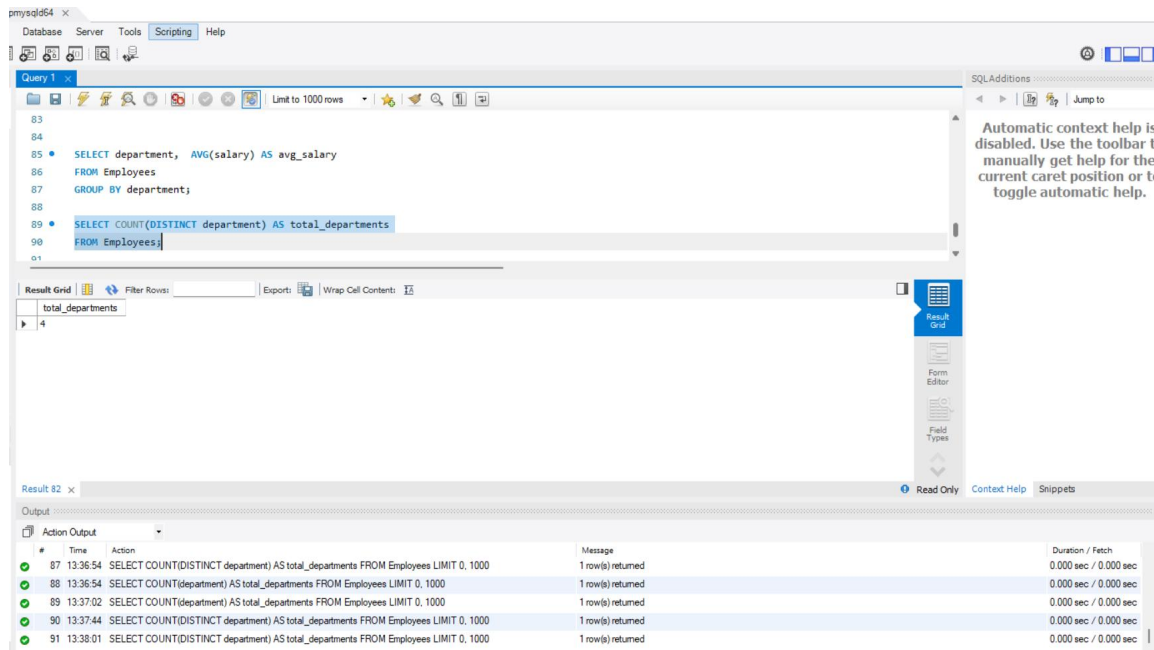
The 'Output' window shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
49	11:19:26	SELECT department, COUNT(emp_id) FROM Employees GROUP BY department LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
50	11:19:26	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated ...	0.000 sec
51	11:19:33	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated ...	0.000 sec
52	11:19:38	select emp_id,max(salary) from employees LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated ...	0.000 sec
53	11:20:10	select max(salary) from employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

## Count Distinct Departments

**SELECT COUNT(DISTINCT department) AS unique\_departments FROM Employees;**

Counts the total number of distinct departments in the company.



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

83
84
85 • SELECT department, AVG(salary) AS avg_salary
86 FROM Employees
87 GROUP BY department;
88
89 • SELECT COUNT(DISTINCT department) AS total_departments
90 FROM Employees;
91

```

The Results tab is active, showing the output of the query. The result set is titled "total\_departments" and contains one row with the value 4.

The Output tab is also visible, showing the execution log. The log indicates that the query was executed successfully and returned 1 row(s).

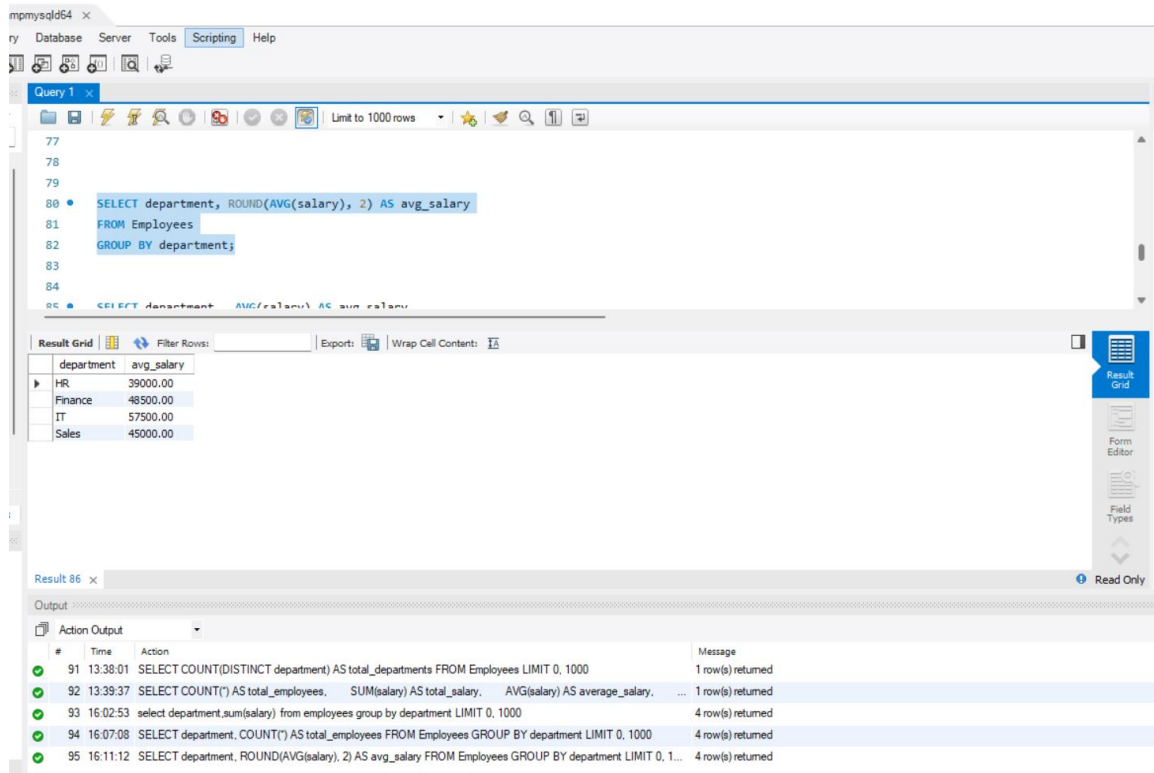
#	Time	Action	Message	Duration / Fetch
87	13:36:54	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
88	13:36:54	SELECT COUNT(department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
89	13:37:02	SELECT COUNT(department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
90	13:37:44	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
91	13:38:01	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

## Rounded Average Salary

**SELECT department, ROUND(AVG(salary), 2) AS rounded\_avg FROM Employees  
GROUP BY department;**

Rounds the average salary to 2 decimal places for better readability.

# Aggregate Functions & Grouping in SQL



The screenshot shows a MySQL IDE window with a SQL query and its results. The query is:

```
SELECT department, ROUND(AVG(salary), 2) AS avg_salary
FROM Employees
GROUP BY department;
```

The results are displayed in a table with two columns: department and avg\_salary.

department	avg_salary
HR	39000.00
Finance	48500.00
IT	57500.00
Sales	45000.00

The bottom panel shows the output of the query, including the time taken and the number of rows returned for each statement.

#	Time	Action	Message
91	13:38:01	SELECT COUNT(DISTINCT department) AS total_departments FROM Employees LIMIT 0, 1000	1 row(s) returned
92	13:39:37	SELECT COUNT(*) AS total_employees, SUM(salary) AS total_salary, AVG(salary) AS average_salary, ...	1 row(s) returned
93	16:02:53	select department_sum(salary) from employees group by department LIMIT 0, 1000	4 row(s) returned
94	16:07:08	SELECT department, COUNT(*) AS total_employees FROM Employees GROUP BY department LIMIT 0, 1000	4 row(s) returned
95	16:11:12	SELECT department, ROUND(AVG(salary), 2) AS avg_salary FROM Employees GROUP BY department LIMIT 0, 1...	4 row(s) returned

## Conclusion

Through this task, I learned how to use aggregate functions along with GROUP BY and HAVING clauses to summarize and analyze data effectively. These concepts are essential for generating meaningful insights from large datasets in real-world applications.