Ex no: 8	**Querying Using GROUP BY, HAVING**
Date:	

Aim:

To write SQL queries using GROUP BY and HAVING clauses to group and filter data.

Procedure:

1. Use the GROUP BY clause to group rows that have the same values in specified columns.
2. Use aggregate functions with GROUP BY to calculate grouped data.
3. Use the HAVING clause to filter groups based on a specified condition.
4. Document the SQL queries used for GROUP BY and HAVING.
5. Execute the queries in the SQL environment.
6. Validate the results by checking the grouped and filtered data.
7. Combine GROUP BY with different aggregate functions.
8. Ensure the queries perform as expected with various data sets.
9. Optimize the queries for better performance.
10. Test the queries with different group by columns.

Queries:

1. Use the GROUP BY clause to group rows that have the same values in specified columns.

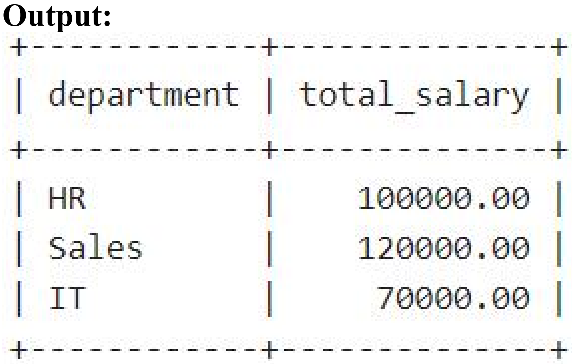
Query;

```
-- Create Employees table
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    role VARCHAR(50),
    salary DECIMAL(10, 2)
);
-- Insert data into Employees table
INSERT INTO employees (employee_id, name, department, role, salary) VALUES
(1, 'Alice', 'HR', 'Manager', 60000),
(2, 'Bob', 'HR', 'Assistant', 40000),
(3, 'Charlie', 'Sales', 'Manager', 75000),
(4, 'David', 'Sales', 'Representative', 45000),
(5, 'Eve', 'IT', 'Developer', 70000);

-- using GROUP BY

SELECT department, SUM(salary) AS total_salary
FROM employees
GROUP BY department;
```

Output:



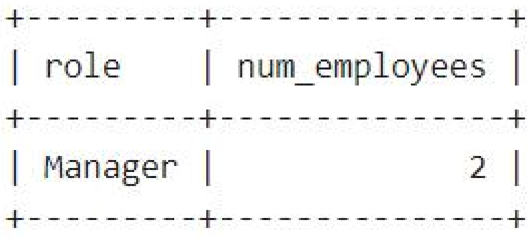
```
+------------+--------------+
| department | total_salary |
+------------+--------------+
| HR         |    100000.00 |
| Sales      |    120000.00 |
| IT         |     70000.00 |
+------------+--------------+
```

2. Use aggregate functions with GROUP BY to calculate grouped data.

Query:

```
SELECT role, COUNT(*) AS num_employees
FROM employees
GROUP BY role
HAVING COUNT(*) > 1;
```

Output:



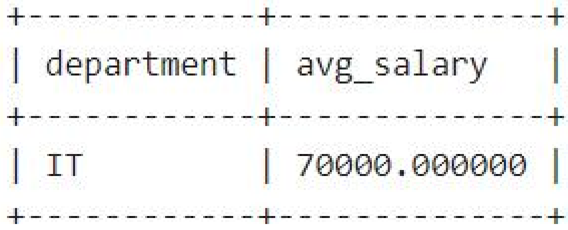
```
+---------+---------------+
| role    | num_employees |
+---------+---------------+
| Manager |             2 |
+---------+---------------+
```

3. Use the HAVING clause to filter groups based on a specified condition.

Query:

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 60000;
```

Output:



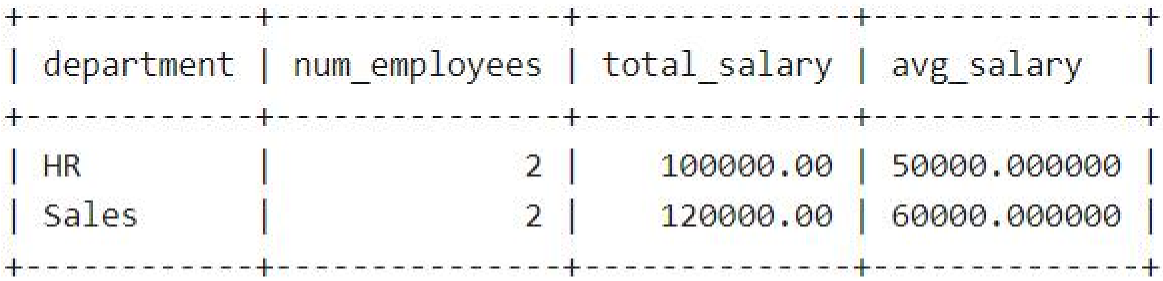
```
+------------+--------------+
| department | avg_salary   |
+------------+--------------+
| IT         | 70000.000000 |
+------------+--------------+
```

4. Document the SQL queries used for GROUP BY and HAVING.

Query:

```
            SELECT department, COUNT(*) AS num_employees, SUM(salary) AS total_salary,
      AVG(salary)   AS avg_salary
        FROM employees
        GROUP BY department
        HAVING COUNT(*) > 1;
```

Output:



```
+------------+---------------+--------------+--------------+
| department | num_employees | total_salary | avg_salary   |
+------------+---------------+--------------+--------------+
| HR         |             2 |    100000.00 | 50000.000000 |
| Sales      |             2 |    120000.00 | 60000.000000 |
+------------+---------------+--------------+--------------+
```

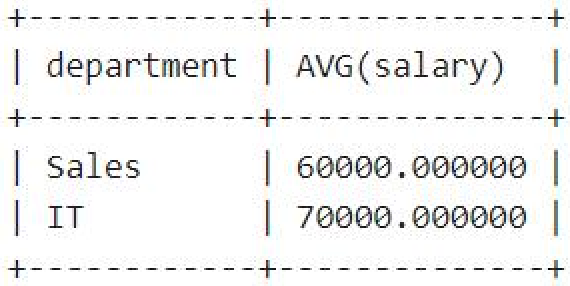
5. Combine GROUP BY with different aggregate functions.

Query:

```
 SELECT department, AVG(salary)
FROM employees
```

```
GROUP BY department
HAVING AVG(salary) > 50000;
```

Output:



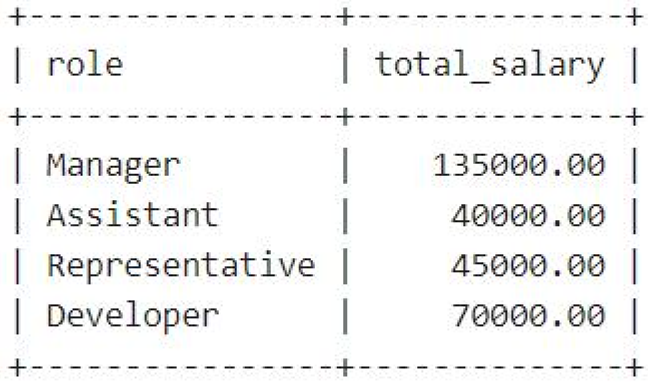
```
+------------+--------------+
| department | AVG(salary)  |
+------------+--------------+
| Sales      | 60000.000000 |
| IT         | 70000.000000 |
+------------+--------------+
```

6. Test the queries with different group by columns.

Query:

```
SELECT role, SUM(salary) AS total_salary
FROM employees
GROUP BY role;
```

Output:



```
+----------------+--------------+
| role           | total_salary |
+----------------+--------------+
| Manager        |    135000.00 |
| Assistant      |     40000.00 |
| Representative |     45000.00 |
| Developer      |     70000.00 |
+----------------+--------------+
```

Algorithm	**15**	
Program	**30**	
Execution	**30**	
Output & Result	**15**	
Viva / Record	**10**	
Total	**100**	

Result:

Thus, we created SQL queries for querying using Aggregate Functions (GROUP BY and HAVING) successfully and the output is verified