Assignment-5

TITLE:

For a given set of relation schemes, create tables and perform the following Simple Queries:

Simple Queries with Aggregate functions, Queries with Aggregate functions (group by and having clause), Queries involving- Date Functions, String Functions , Math Functions

OBJECTIVE:

To understand and apply aggregate functions, group by and having clauses, and built-in functions (Date, String, Math) on relational tables using SQL.

SOFTWARE & HARDWARE REQUIREMENTS:

- **Software:** MySQL / Oracle / PostgreSQL / SQLite
- **Hardware:** Any computer system with SQL-compatible DBMS installed

THEORY:

1. Aggregate Functions:
Used to perform a calculation on a set of values and return a single value. Examples: SUM(), AVG(), COUNT(), MIN(), MAX()

2. GROUP BY Clause:
Used to arrange identical data into groups.

3. HAVING Clause:
Used to filter the results after grouping with the GROUP BY clause.

4. Date Functions:
Used to perform operations on date values. Examples: NOW(), CURDATE(), DATEDIFF(), YEAR(), MONTH(), etc.

5. String Functions:
Used for operations on string data. Examples: UPPER(), LOWER(), LENGTH(), CONCAT(), SUBSTRING(), etc.

6. Math Functions:
Used to perform mathematical calculations. Examples: ROUND(), CEIL(), FLOOR(), POWER(), SQRT(), etc.

RELATION SCHEMA USED:

Employee

EmpID	EmpName	Department	Salary	DOJ (Date of Joining)
101	Alice	HR	45000	2020-03-15
102	Bob	IT	60000	2019-07-10
103	Charlie	IT	55000	2021-01-25
104	David	HR	48000	2022-10-12
105	Eva	Sales	52000	2023-01-01

SQL QUERIES AND OUTPUTS:

1. Simple Queries with Aggregate Functions

-- Total number of employees

```
SELECT COUNT(*) AS Total_Employees FROM Employee;
```

-- Average salary

```
SELECT AVG(Salary) AS Average_Salary FROM Employee;
```

-- Maximum and Minimum salary

```
SELECT MAX(Salary) AS Highest_Salary, MIN(Salary) AS Lowest_Salary FROM Employee;
```

2. Queries with GROUP BY and HAVING Clause

-- Average salary by department

```
SELECT Department, AVG(Salary) AS Avg_Dept_Salary
FROM Employee
GROUP BY Department;
```

-- Departments with average salary greater than 50000

```
SELECT Department, AVG(Salary) AS Avg_Dept_Salary
FROM Employee
GROUP BY Department
```

HAVING AVG(Salary) > 50000;

3. Queries Involving Date Functions

-- Display employee name and year of joining

SELECT EmpName, YEAR(DOJ) AS Year_Joined FROM Employee;

-- Find employees who joined after 01-Jan-2021

SELECT EmpName, DOJ FROM Employee

WHERE DOJ > '2021-01-01';

-- Find how many days each employee has been working

SELECT EmpName, DATEDIFF(CURDATE(), DOJ) AS Days_Worked FROM Employee;

4. Queries Involving String Functions

-- Display names in uppercase

SELECT UPPER(EmpName) AS Upper_Name FROM Employee;

-- Concatenate name with department

SELECT CONCAT(EmpName, ' - ', Department) AS Full_Detail FROM Employee;

-- Find length of each employee's name

SELECT EmpName, LENGTH(EmpName) AS Name_Length FROM Employee;

5. Queries Involving Math Functions

-- Round salary to nearest thousand

SELECT EmpName, ROUND(Salary, -3) AS Rounded_Salary FROM Employee;

-- Display square root of salary

SELECT EmpName, SQRT(Salary) AS Salary_Sqrt FROM Employee;

-- Display ceiling and floor of salary divided by 1000

SELECT EmpName, CEIL(Salary/1000) AS Ceil_Val, FLOOR(Salary/1000) AS Floor_Val FROM Employee;

RESULT:

The above queries were executed successfully on the Employee table. The output demonstrated correct usage of aggregate functions, group by and having clauses, and built-in Date, String, and Math functions.

CONCLUSION:

Through this practical, we gained hands-on experience with various SQL functions that are essential for data analysis and reporting. We learned how to aggregate data, group results, apply conditions to groups, and use in-built functions to manipulate date, string, and numeric values effectively.