**SQL Queries Using Aggregate Functions, GROUP BY, HAVING, and Views**

**Example 1:**

**Aim:**

To understand and execute SQL queries using **aggregate functions**, **GROUP BY**, and **HAVING** clauses, and to learn how to **create and drop views** in a relational database.

**Steps**

1. Create sample tables with appropriate data.
2. Insert sample data into the tables.
3. Write SQL queries using aggregate functions (SUM(), AVG(), COUNT(), MAX(), MIN()).
4. Use the GROUP BY clause to group records based on one or more columns.
5. Apply the HAVING clause to filter grouped records.
6. Create a **View** using the CREATE VIEW statement.
7. Drop a **View** using the DROP VIEW statement.

**Tables**

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

Name VARCHAR(50),

Department VARCHAR(50),

Marks INT

);

INSERT INTO Student VALUES

(1, 'Raj', 'CSE', 85),

(2, 'Priya', 'ECE', 92),

(3, 'Amit', 'CSE', 75),

(4, 'Sara', 'ECE', 88),

(5, 'John', 'EEE', 67),

(6, 'Neha', 'CSE', 95);

**Aggregate Function Queries:**

1. Write a SQL query to find the total number of students

SELECT COUNT(\*) AS Total\_Students FROM Student;

2. Write a SQL query to find Average marks of all students

SELECT AVG(Marks) AS Average\_Marks FROM Student;

3. Write a SQL query to find Highest marks

SELECT MAX(Marks) AS Highest\_Marks FROM Student;

4. Write a SQL query to find Lowest marks

SELECT MIN(Marks) AS Lowest\_Marks FROM Student;

5. Write a SQL query to find Sum of all marks

SELECT SUM(Marks) AS Total\_Marks FROM Student;

**Queries Using GROUP BY**

6. Write a SQL query to Count of students in each department

SELECT Department, COUNT(\*) AS No\_of\_Students

FROM Student

GROUP BY Department;

7. Write a SQL query to find Average marks department-wise

SELECT Department, AVG(Marks) AS Average\_Marks

FROM Student

GROUP BY Department;

**Queries Using HAVING**

1. Write a SQL query to find Departments having more than 1 student

SELECT Department, COUNT(\*) AS No\_of\_Students

FROM Student

GROUP BY Department

HAVING COUNT(\*) > 1;

9. Write a SQL query to find Departments with average marks above 80

SELECT Department, AVG(Marks) AS Avg\_Marks

FROM Student

GROUP BY Department

HAVING AVG(Marks) > 80;

**Creating and Dropping Views**

10. Create a view of high scoring students (marks > 80)

CREATE VIEW HighScorers AS

SELECT StudentID, Name, Marks

FROM Student

WHERE Marks > 80;

11. Select data from the view

SELECT \* FROM HighScorers;

12. Drop the view

DROP VIEW HighScorers;

**Conclusion**

Using **aggregate functions** along with GROUP BY and HAVING enables us to perform complex analytical queries in SQL.

**Views** allow for simplified access to complex queries and enhance security and modularity in database systems.

Example 2:

Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.

Step 1: Retrieve data from instructor table

Step 2: Write the following aggregate queries using aggregate functions(max,min,avg,count,sum)

Write SQL Query to find the highest(maximum) salary of instructor(max

Write SQL Query to find the details highest(maximum) salary of instructor

Write SQL Query to find the second highest(maximum) salary of instructor

Write SQL Query to find the details second highest(maximum) salary of instructor

Write SQL Query to find the least(minimum) salary of instructor

Write SQL Query to find the details least(minimum) salary of instructor

Write SQL Query to find the average salary of instructor

Write SQL Query to find the sum of salaries of all the instructors

Write a SQL query to find the number of tuples/records/rows in the instructor table( relation)

Or   
 Write a SQL query to find the number of instructors in the instructor table( relation)

Write a SQL query to find the average salary of instructors in each department

Write a SQL Query to find the average salary of instructors in the Computer Science department

Write a SQL Query to find the names and average salaries of all departments whose average

salary is greater than 42000.

Step 4: Create view

Step 5: drop view

Experiment: Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.

Aim: To implement aggregate functions, Group by, Having and creation and drop of views

Aggregate functions operate on the multiset of values of a column of a relation, and return a value.

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the **GROUP BY** clause of the **SELECT** statement.

The **GROUP BY** clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

The most commonly used SQL aggregate functions

* **MIN( )** - returns the smallest value within the selected column
* MAX( ) - returns the largest value within the selected column
* COUNT( ) - returns the number of rows in a set
* SUM( ) - returns the total sum of a numerical column
* AVG( ) - returns the average value of a numerical column

Aggregate functions ignore null values (except for COUNT()).

Schema - instructor(ID, name, dept\_ name, salary)

Write a SQL query to find the all details of instructors :  
mysql> select \* from instructor;

+----------+---------------+-----------------+--------------+

| ID | name | dept\_name | salary |

+----------+---------------+-----------------+-------------+

| 10101 | Srinivasan | Comp.Sci | 65000.00 |

| 12121 | Wu | Finance | 90000.00 |

| 15151 | Mozart | Music | 40000.00 |

| 22222 | Einstein | Physics | 95000.00 |

| 32343 | El Said | History | 60000.00 |

| 33456 | Gold | Physics | 87000.00 |

| 45565 | Katz | Comp.Sci | 75000.00 |

| 58583 | Califieri | History | 62000.00 |

| 76543 | Singh | Finance | 80000.00 |

| 76766 | Crick | Biology | 72000.00 |

| 83821 | Brandt | Comp.Sci | 92000.00 |

| 98345 | Kim | Elec.Eng | 80000.00 |

+----------+--------------+-----------------+--------------+

12 rows in set (0.02 sec)

Write SQL Query to find the highest(maximum) salary of instructor

mysql> select max(salary) from instructor;

+----------------+

| max(salary) |

+----------------+

| 95000.00 |

+----------------+

1 row in set (0.00 sec)

Write SQL Query to find the details highest(maximum) salary of instructor

mysql> select \* from instructor where salary=(select max(salary) from instructor);

+---------+-----------+-----------------+--------------+

| ID | name | dept\_name | salary |

+---------+-----------+-----------------+--------------+

| 22222 | Einstein | Physics | 95000.00 |

+---------+------------+----------------+--------------+

1 row in set (0.00 sec)

Write SQL Query to find the second highest(maximum) salary of instructor

mysql> select max(salary) from instructor where salary not in (select max(salary) from instructor);

+----------------+

| max(salary) |

+----------------+

| 92000.00 |

+---------------+

1 row in set (0.00 sec)

OR

Write SQL Query to find the second highest(maximum) salary of instructor

mysql> select max(salary) from instructor where salary <>(select max(salary) from instructor);

+-----------------+

| max(salary) |

+-----------------+

| 92000.00- |

+-----------------+

1 row in set (0.00 sec)

Write SQL Query to find the details second highest(maximum) salary of instructor

mysql> select \* from instructor where salary=(select max(salary) from instructor where salary <>(select max(salary) from instructor));

+----------+----------+-------------------+---------------+

| ID | name | dept\_name | salary |

+----------+----------+------------------+----------------+

| 83821 | Brandt | Comp.Sci | 92000.00 |

+----------+---------+------------------+-----------------+

1 row in set (0.00 sec)

(General Query N th Highest salary of instructor (N=1) **Alternative Query** )

Write SQL Query to find the details highest(maximum) salary of instructor

mysql> SELECT \* from instructor in1

where N-1 = (SELECT COUNT(DISTINCT salary) from instructor in2

where in2.salary > in1.salary) ;

mysql> SELECT \* from instructor in1 where 1-1 = (SELECT COUNT(DISTINCT salary)

from instructor in2 where in2.salary > in1.salary) ;

+----------+------------+----------------+-------------+

| ID | name | dept\_name | salary |

+----------+------------+----------------+-------------+

| 22222 | Einstein | Physics | 95000.00 |

+----------+------------+---------------+---------------+

1 row in set (0.04 sec)

mysql>

Write SQL Query to find the details highest(maximum) salary of instructor (N=2)

( **Alternative Query** )

mysql> SELECT name,salary from instructor in1 where 2-1 = (SELECT COUNT(DISTINCT salary) from instructor in2 where in2.salary > in1.salary) ;

+----------+--------------+

| name | salary |

+----------+--------------+

| Brandt | 92000.00 |

+----------+--------------+

1 row in set (0.00 sec)

Write SQL Query to find the least(minimum) salary of instructor

mysql> select min(salary) from instructor;

+-------------+

| min(salary) |

+-------------+

| 40000.00 |

+-------------+

1 row in set (0.00 sec)

Write SQL Query to find the details least(minimum) salary of instructor

mysql> select \* from instructor where salary=(select min(salary) from instructor);

+---------+-----------+-----------------+-------------+

| ID | name | dept\_name | salary |

+---------+-----------+----------------+--------------+

| 15151 | Mozart | Music | 40000.00 |

+---------+-----------+----------------+--------------+

1 row in set (0.00 sec)

mysql>

Write SQL Query to find the average salary of instructor

mysql> select avg(salary) from instructor;

+--------------------+

| avg(salary) |

+--------------------+

| 74833.333333 |

+-------------------- +

1 row in set (0.00 sec)

Write SQL Query to find the sum of salaries of all the instructors

mysql> select sum(salary) from instructor;

+-------------+

| sum(salary) |

+-------------+

| 898000.00 |

+-------------+

1 row in set (0.00 sec)

Write a SQL query to find the number of tuples/records/rows in the instructor table( relation)

Or   
Write a SQL query to find the number of instructors in the instructor table( relation)

mysql> select count(ID) from instructor;

+-----------+

| count(ID) |

+-----------+

| 12 |

+-----------+

1 row in set (0.01 sec)

mysql>

OR

mysql> select count(\*) from instructor;

+----------+

| count(\*) |

+----------+

| 12 |

+----------+

1 row in set (0.00 sec)

mysql>

Find the average salary of instructors in each department

mysql> select dept\_name, avg (salary) as avg\_salary from instructor group by dept\_name;

+-----------+--------------+

| dept\_name | avg\_salary |

+-----------+--------------+

| Biology | 72000.000000 |

| Comp.Sci | 77333.333333 |

| Elec.Eng | 80000.000000 |

| Finance | 85000.000000 |

| History | 61000.000000 |

| Music | 40000.000000 |

| Physics | 91000.000000 |

+-----------+--------------+

7 rows in set (0.00 sec)

mysql>

write a SQL Query to find the average salary of instructors in the Computer Science department  
mysql> select avg (salary) from instructor where dept\_name= 'Comp.Sci';

+--------------+

| avg (salary) |

+--------------+

| 77333.333333 |

+--------------+

1 row in set (0.00 sec)

mysql>

Write a SQL Query to find the names and average salaries of all departments whose average salary is greater than 42000.

mysql> select dept\_name, avg (salary) as avg\_salary from instructor  
 group by dept\_name having avg (salary) > 42000

Note: predicates in the **having** clause are applied after the formation of **groups**

whereas predicates in the **where** clause are applied before forming **groups**

mysql> select dept\_name, avg (salary) as avg\_salary from instructor

group by dept\_name having avg (salary) > 42000;

+-----------+--------------+

| dept\_name | avg\_salary |

+-----------+--------------+

| Biology | 72000.000000 |

| Comp.Sci | 77333.333333 |

| Elec.Eng | 80000.000000 |

| Finance | 85000.000000 |

| History | 61000.000000 |

| Physics | 91000.000000 |

+-----------+--------------+

6 rows in set (0.00 sec)

In **SQL**, **a view** is a virtual **table** based on the result-set of an **SQL** statement.

**A view** contains rows and columns, just like a real **table.**

A view is defined using the **create view** statement which has the form

**create view** *v* **as** *<* query expression >  
where <query expression> is any legal SQL expression.

The view name is represented by *v.*

**Create view**

A view of instructors without their salary

*mysql>create view faculty as select ID, name, dept\_name from instructor;*

*Query OK, 0 rows affected (0.05 sec)*

mysql> desc faculty ;

+-----------------+-------------+-------+------+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-----------------+------------+-------+------+---------+-------+

| ID | char(5) | NO | | NULL | |

| name | char(20) | NO | | NULL | |

| dept\_name | char(20) | YES | | NULL | |

+----------------+------------+-------+------+---------+-------+

3 rows in set (0.02 sec)

mysql>

Write a SQL to find all instructors in the Biology department (view faculty)

**mysql>select** *name* **from** *faculty* **where** *dept\_name =* 'Biology'

mysql> select name from faculty where dept\_name = 'Biology';

+---------+

| name |

+---------+

| Crick |

+---------+

1 row in set (0.00 sec)

mysql>

Create a view of department salary totals

**create view** *departments\_total\_salary(dept\_name, total\_salary)* **as select** *dept\_name*, **sum** (*salary*) **from** *instructor* **group by** *dept\_name*;

mysql> create view departments\_total\_salary(dept\_name, total\_salary) as select dept\_name, sum(salary) from instructor group by dept\_name;

Query OK, 0 rows affected (0.02 sec)

mysql> desc departments\_total\_salary;

+--------------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------------+---------------+------+-----+---------+-------+

| dept\_name | char(20) | YES | | NULL | |

| total\_salary | decimal(30,2) | YES | | NULL | |

+--------------+---------------+------+-----+---------+-------+

2 rows in set (0.01 sec)

mysql>

mysql> select \* from departments\_total\_salary;

+-----------+--------------+

| dept\_name | total\_salary |

+-----------+--------------+

| Biology | 72000.00 |

| Comp.Sci | 232000.00 |

| Elec.Eng | 80000.00 |

| Finance | 170000.00 |

| History | 122000.00 |

| Music | 40000.00 |

| Physics | 182000.00 |

+-----------+--------------+

7 rows in set (0.00 sec)

mysql>

mysql> drop view faculty;

Query OK, 0 rows affected (0.02 sec)

mysql>