SQL Aggregate Functions

Aggregate functions in SQL are functions that perform calculations on sets of values and return a single summarized value. These functions are particularly useful when we need to summarize, analyze, or group large datasets in SQL databases.

Key Features of SQL Aggregate Functions

1-Set-based operations: Work on multiple rows at once

2-Single return value: Always return one result per group

3-Null handling: Typically ignore NULL values in calculations

Common Used SQL Aggregate Functions

1. Count()

A- The COUNT() function returns the number of rows that match a given condition or are present in a column.

B- COUNT(*): Counts all rows.

C- COUNT(column_name) : Counts non-NULL values in the specified column.

D- COUNT (DISTINCT column_name): Counts unique non-NULL values in the column.

Examples:

```
-- Total number of records in the table
SELECT COUNT(*) AS TotalRecords FROM Employee;

-- Count of non-NULL salaries
SELECT COUNT(Salary) AS NonNullSalaries FROM Employee;

-- Count of unique non-NULL salaries
SELECT COUNT(DISTINCT Salary) AS UniqueSalaries FROM Employee;
```

2. SUM( )

A- The SUM() function calculates the total sum of a numeric column**.**

B- SUM(column_name): Returns the total sum of all non-NULL values in a column.

Examples:

```
-- Calculate the total salary
SELECT SUM(Salary) AS TotalSalary FROM Employee;

-- Calculate the sum of unique salaries
SELECT SUM(DISTINCT Salary) AS DistinctSalarySum FROM Employee;
```

3. AVG()

A-The AVG() function calculates the average of a numeric column. It divides the sum of the column by the number of non-NULL rows.

B-AVG(column_name): Returns the average of the non-NULL values in the column.

```
-- Calculate the average salary
SELECT AVG(Salary) AS AverageSalary FROM Employee;

-- Average of distinct salaries
SELECT AVG(DISTINCT Salary) AS DistinctAvgSalary FROM Employee;
```

4. MIN() and MAX()

A-The MIN() and MAX() functions return the smallest and largest values, respectively, from a column.

B-MIN(column_name): Returns the minimum value.

C-MAX(column_name): Returns the maximum value.

```
-- Find the highest salary
SELECT MAX(Salary) AS HighestSalary FROM Employee;

-- Find the lowest salary
SELECT MIN(Salary) AS LowestSalary FROM Employee;
```

Examples of SQL Aggregate Functions

Let's consider a demo Employee table to demonstrate SQL aggregate functions . This table contains employee details such as their ID, Name, and Salary.

d	Name	Salary
1	A	802
2	B	403
3	C	604
4	D	705
5	E	606
6	F	NULL

1. Count the Total Number of Employees

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

Output:

TotalEmployees
6

2. Calculate the Total Salary

```
SELECT SUM(Salary) AS TotalSalary FROM Employee;
```

Output:

TotalSalary
3120

3. Find the Average Salary:

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

Output:

AverageSalary
624

4. Find the Highest and Lowest Salary

```
SELECT MAX(Salary) AS HighestSalary FROM Employee;
```

Output:

HighestSalary	LowestSalary
802	403

What is GROUP BY in SQL?

The **GROUP BY** statement in SQL is used to arrange identical data into groups based on specified columns.

A- GROUP BY clause is used with the SELECT statement.

B-In the query, the GROUP BY clause is placed after the WHERE.

C-In the query, the GROUP BY clause is placed before the ORDER BY if used.

D-In the query, the GROUP BY clause is placed before the Having.

E-Place condition in the having clause.

Syntax

```
SELECT column1, function_name(column2)
FROM table_name
GROUP BY column1, column2
```

SQL GROUP BY Examples

Let's assume that we have two tables Employee and Student

Emp

emp_no	name	sal	age
1	Aarav	50000	25
2	Aditi	60000.5	30
3	Aarav	75000.75	35
4	Anjali	45000.25	28
5	Chetan	80000	32
6	Divya	65000	27
7	Gaurav	55000.5	29
8	Divya	72000.75	31
9	Gaurav	48000.25	26
10	Divya	83000	33

Student

name	year	subject
Alice	1	Mathematics
Bob	2	English
Charlie	3	Science
David	1	Mathematics
Emily	2	English
Frank	3	Science

Example 1 : Group By Single Column

Group By single column means, placing all the rows with the same value of only that particular column in one group. Consider the query for Calculating the Total Salary of each Employee by their name in the emp table

```
SELECT name, SUM(sal) FROM emp
GROUP BY name;
```

Output

name	SUM(sal)
Aarav	125000.75
Aditi	60000.5
Anjali	45000.25
Chetan	80000
Divya	220000.75
Gaurav	103000.75

As you can see in the above output, the rows with duplicate NAMEs are grouped under the same NAME and their corresponding SALARY is the sum of the SALARY of duplicate rows. The SUM() function of SQL is used here to calculate the sum. The NAMES that are added are Aarav, Divya and Gaurav.

Example 2 : Group By Multiple Columns

Group by multiple columns is say, for example, **GROUP BY column1, column2**. This means placing all the rows with the same values of columns **column 1** and **column 2** in one group. This SQL query **groups student records by both SUBJECT and YEAR** and then **counts the number of records** (i.e., students) in each of those groups.

Query:

```
SELECT SUBJECT, YEAR, Count(*)
FROM Student
GROUP BY SUBJECT, YEAR;
```

Output:

Output

subject	year	Count(*)
English	2	2
Mathematics	1	2
Science	3	2

As we can see in the above output the students with both the same SUBJECT and YEAR are placed in the same group. And those whose only SUBJECT is the same but not YEAR belong to different groups. So here we have grouped the table according to two columns or more than one column. The Grouped subject and years are (English,2) , (Mathematics,1) and (Science,3). The above mentioned all groups and years are repeated twice.

What is the SQL HAVING Clause?

The HAVING clause is used to filter the result of the GROUP BY statement based on the specified conditions.

A-It was introduced because the WHERE cannot be used with aggregate functions.

B-Similar to WHERE , need to filter aggregated results, the HAVING clause is the appropriate choice.

Syntax:

```
SELECT
AGGREGATE_FUNCTION(column_name)
FROM table_name
GROUP BY column_name
HAVING condition;
```

SQL HAVING Examples

EmployeeId	Name	Gender	Salary	Department	Experience
5	Priya Sharma	Female	45000	IT	2 years
6	Rahul Patel	Male	65000	Sales	5 years
7	Nisha Gupta	Female	55000	Marketing	4 years
8	Vikram Singh	Male	75000	Finance	7 years
9	Aarti Desai	Female	50000	IT	3 years

Example 1 : Using HAVING to Filter Aggregated Results

This employee table will help us understand the HAVING Clause. It contains employee IDs, Name, Gender, department, and salary. To Know the sum of salaries, we will write the query:

```
SELECT Department, sum(Salary) as Salary
FROM Employee
GROUP BY department;
```

Output:

Department	Salary
Finance	75000
IT	95000
Marketing	55000
Sales	65000

Now if we need to display the departments where the sum of salaries is 50,000 or more. In this condition, we will use the HAVING Clause.

```
SELECT Department, sum(Salary) as Salary
FROM Employee
GROUP BY department
HAVING SUM(Salary) >= 50000;
```

Output:

Department	Salary
Finance	75000
IT	95000
Marketing	55000
Sales	65000

Conclusion

SQL Aggregate Functions are fundamental tools for summarizing large datasets. Whether you're calculating totals, averages, or finding maximum and minimum values, aggregate functions enable you to quickly perform complex calculations on your data. By mastering these functions, you can gain valuable insights, spot trends, and perform deep analysis on any dataset. With practice, you can leverage these functions with GROUP BY, HAVING, and other clauses to create powerful and efficient SQL queries.