

# DBMS/SQL

Lesson 04: Aggregate  
(GROUP) Functions

## Lesson Objectives

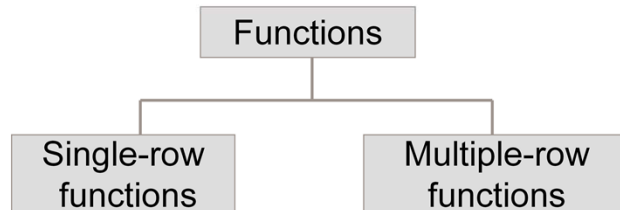
- To understand the following topics:
  - Introduction to Functions
  - Aggregate (Group) functions:
    - GROUP BY clause
    - HAVING clause



## 4.1: Introduction to Functions

## Types of SQL Functions

- Single row functions :
  - Operate on single rows only and return one result per row
- Multiple row functions:
  - Manipulates groups of rows to give one result per group of rows. Also called as group functions



Copyright © Capgemini 2015. All Rights Reserved 3

**Functions:**

- Functions can be used to manipulate data values in a variety of ways.
  - Functions help in making the basic query block more powerful.
  - Functions may be used to perform calculations on data, alter data formats for display, convert data types, etc.
- Functions are similar to operators in that they manipulate data items and return a result.
- Functions differ from operators in the format in which they appear with their arguments. This format allows them to operate on zero, one, two, or more arguments:
  - function(argument, argument, ...)
- There are two types of functions: SQL functions and User-defined functions.
  1. **SQL functions:** SQL functions are built into most DBMS and are available for use in various appropriate SQL statements. SQL functions are further categorized as:
    - a) **Single-Row functions:** Single-row functions return a single result row for every row of a queried Table or View.  
**For example:** ABS, ROUND etc.

- b) **Aggregate functions:** Aggregate functions return a single row based on groups of rows, rather than on single rows. **For example:** MAX, MIN, COUNT, etc.
- 2. **User-defined functions:** You can write user-defined functions in PL/SQL or Java to provide functionality that is not available in SQL or SQL functions. User-defined functions can appear in a SQL statement wherever SQL functions can appear, that is, wherever an expression can occur.  
**For example:** User-defined functions can be used in the following:
  - a) The select list of a SELECT statement
  - b) The condition of a WHERE clause
  - c) ORDER BY and GROUP BY clauses
  - d) The VALUES clause of an INSERT statement
  - e) The SET clause of an UPDATE statement

4.1: Introduction to Functions

## The Group Functions

- The Group functions are built-in SQL functions that operate on “groups of rows”, and return one value for the entire group.
- The results are also based on groups of rows.
- For Example, Group function called “SUM” will help you find the total marks, even if the database stores only individual subject marks.



Copyright © Capgemini 2015. All Rights Reserved. 5

### **Aggregate (Group) Functions:**

- SQL provides a set of “built-in” functions for producing a single value for an entire group. These functions are called as “Set functions” or “Aggregate (Group) functions”.
- These functions can work on a “normal result table” or a “grouped result table”.
  - If the result is not grouped, then the aggregate will be taken for the whole result table.

4.1: Introduction to Functions

## Syntax : GROUP BY & HAVING clause

### ■ Syntax

```
SELECT      [column, ] aggregate function(column), .....  
FROM        table  
[WHERE      condition]  
[GROUP BY column]  
[HAVING condition]  
[ORDER BY column] ;
```

## 4.1: Introduction to Functions

## Listing of Group Functions

- Given below is a list of Group functions supported by SQL:

Function	Value returned
SUM (expr)	Sum value of expr, ignoring NULL values.
AVG (expr)	Average value of expr, ignoring NULL values.
COUNT (expr)	Number of rows where expr evaluates to something other than NULL. COUNT(*) counts all selected rows, including duplicates and rows with NULLs.
MIN (expr)	Minimum value of expr.
MAX (expr)	Maximum value of expr.

**Aggregate (Group) Functions supported by SQL:**

- All the above functions operate on a number of rows (for example, an entire table), and are therefore known as “Group (or Aggregate) functions”.
- A Group function can be used on a subset of the rows in a table by using the WHERE clause.
- The Aggregate functions ignore NULL values in the column.
  - To include NULL values, NVL function can be used with Aggregate functions.

**Note :**

- Count(\*):** Returns the number of rows in the table, including duplicates and those with NULLs.
- Count(<Expression>):** Returns the number of rows where expression is NOT NULL.

**Aggregate (Group) Functions supported by SQL (contd.):****SUM(COL\_NAME | EXPRESSION)**

- SUM returns the total of values present in a particular “column” or a “number of columns” that are linked together in the expression. All the columns, which form the argument to SUM, must be numeric only.
- To find the sum of one subject for all students following query is used:

```
SELECT SUM(subject1)
      FROM student_marks;
```

- To find the yearly compensation of staff from department 20, the following query is used:

```
SELECT SUM(12*staff_sal)
      FROM staff_master
      WHERE dept_code = 20;
```

**AVG(COL\_NAME | EXPRESSION)**

- AVG is similar to SUM. AVG returns the average of a NUMBER of values. The restrictions, which apply on SUM also, apply on AVG.
- To find the average salary the staff, the following query is used:

```
SELECT AVG(sal)
      FROM staff_master;
```

- To find the average yearly compensation of staff from department 20, the following query is used:

```
SELECT AVG(12*staff_sal)
      FROM staff_master
      WHERE dept_code = 20;
```



**Aggregate (Group) Functions supported by SQL (contd.):****COUNT(\*)**

- COUNT returns the number of rows.
- To find the total number of staff members, the following query is used:

```
SELECT COUNT(*)  
FROM staff_master;
```

- It is possible to restrict the rows for the operation of COUNT.
- To find the total number of staff members in department 10, the following query is used:

```
SELECT COUNT(*)  
FROM staff_master  
WHERE dept_code = 10 ;
```

- To find the total number of staff members hired after '01-JAN-02', the following query is used:

```
SELECT COUNT(*)  
FROM staff_master  
WHERE hiredate >'01-JAN-02' ;
```

- When an aggregate function is used in a SELECT statement, column names cannot be used in SELECT unless GROUP BY clause is used.

**MIN(COL\_NAME | EXPRESSION)**

- MIN returns the lowest of the values from the column. MIN accepts columns, which are NON-NUMERIC too.
- To find the minimum salary paid to a staff member, the following query is used:

```
SELECT MIN(staff_sal)  
FROM staff_master;
```

- To list the staff member who alphabetically heads the list, the following query is used:

```
SELECT MIN(staff_name)  
FROM staff_master;
```

```
SELECT MAX(subject2)  
FROM student_marks ;
```

**MAX(COL\_NAME | EXPRESSION)**

- MAX is the reverse of MIN. MAX returns the maximum value from among the list of values.
- To find the maximum marks for a student in subject2, the following query is used:

## 4.1: Introduction to Functions

## Examples of using Group Functions

- Example 1: Display the total number of records from student\_marks.

```
SELECT COUNT( * )  
FROM Student_Marks;
```

- Example 2: Display average marks from each subject.

```
SELECT AVG(Student_sub1), AVG(Student_sub2), AVG(Student_sub3)  
FROM Student_Marks;
```



Copyright © Capgemini 2015. All Rights Reserved. 10

**Note:**

- The first query returns the value, which counts the number of rows fetched from the student\_marks table. All the rows in the table are treated as one group. Group Functions operate on sets of rows to give one result per group. Can be used on the whole table or certain set of rows

4.2 : Using the GROUP BY &amp; HAVING clause

## The GROUP BY clause

- GROUP BY clause is used along with the Group functions to retrieve data that is grouped according to one or more columns.
- For example: Displays the average staff salary based on every department. The values are grouped based on dept\_code

```
SELECT Dept_Code, AVG(Staff_sal)
FROM Staff_Master
GROUP BY Dept_Code;
```



Copyright © Capgemini 2015. All Rights Reserved 11

### Usage of GROUP BY and HAVING clauses:

- All the SELECT statements we have used until now have acted on data as if the data is in a "single group". But the rows of data in some of the tables can be thought of as being part of "different groups".  
**For example:** The staff\_master table contains staff information allocated to various departments identified by dept\_code. If we wish to find the minimum salary of each group of employees in respective department, then none of the clauses that we have seen until now are of any use.
- The GROUP BY clause is used to group the result table derived from earlier FROM and WHERE clauses. Further, a HAVING clause is used to apply search condition on these groups.
  - When a GROUP BY clause is used, each row of the resulting table will represent a group having same values in the column(s) used for grouping.
  - Subsequently, the HAVING clause acts on the resulting grouped table to remove the row that does not satisfy the criteria in the HAVING search condition

**The GROUP BY Clause:**

If you have to fetch columns other than group functions or if you want your results to be grouped on certain criteria use "GROUP BY" clause

- The GROUP BY clause is of the form:

GROUP BY <column list>

- The columns specified must be selected in the query. If a table is grouped, the SELECT list should have columns and expressions, which are single-valued for a group. These can be:
  - columns on which grouping is done
  - constants
  - aggregate functions on the other columns on which no grouping is done
- 1. The GROUP BY clause should contain all the columns in the SELECT list, except those used along with the Group functions. Only the column names which have been used in GROUP BY clause and aggregate columns can be used in SELECT clause
- 2. When an Aggregate (Group) function is used in a SELECT statement, the column names cannot be used in SELECT, unless GROUP BY clause is used.
- 3. Grouping can be done on multiple columns, as well.

4.2 : Using the GROUP BY &amp; HAVING clause

## The HAVING clause

- HAVING clause is used to filter data based on the Group functions.
  - HAVING clause is similar to WHERE condition. However, it is used with Group functions.
- Group functions cannot be used in WHERE clause. However, they can be used in HAVING clause.



Copyright © Capgemini 2015. All Rights Reserved. 13

### The HAVING Clause:

- A HAVING clause is of the form:

HAVING <search condition>

- The HAVING search condition applies to “each group”. It can be:
  - formed using various predicates like between, in, like, null, comparison, etc
  - combined with Boolean operators like AND, OR, NOT
- Since the search condition is for a “grouped table”. The predicates should be:
  - on a column by which grouping is done.
  - on a set function (Aggregate function) on other columns.
- The aggregate functions can be used in HAVING clause. However, they cannot be used in the WHERE clause.
- When WHERE, GROUP BY, and HAVING clauses are used together in a SELECT statement:
  1. The WHERE clause is processed first in order.
  2. Subsequently, the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.
  3. Finally, any conditions, on the Group functions in the HAVING clause, are applied to the grouped rows before the final output is displayed.

4.2 : Using the GROUP BY &amp; HAVING clause

## Examples – GROUP BY and HAVING clause

- For example: Display all department numbers having more than five employees.

```
SELECT Department_Code, Count(*)  
FROM Staff_Master  
GROUP BY Department_Code  
HAVING Count(*) > 5;
```



Copyright © Capgemini 2015. All Rights Reserved 14

### The HAVING clause (contd.):

- To find out Average, Maximum, Minimum salary of departments, where average salary is greater than 2000.

```
SELECT dept_code,AVG(staff_sal),MIN(staff_sal),  
MAX(staff_sal)  
FROM staff_master  
GROUP BY dept_code HAVING AVG(staff_sal) >  
2000;
```

- To find out average salary of all staff members who belong to department 10 or 20 and their average salary is greater than 10000

```
SELECT design_code,dept_code,avg(staff_sal)  
FROM staff_master where dept_code in(10,20)  
GROUP BY design_code,dept_code  
HAVING avg(staff_sal) >10000;
```

4.3: Tips and Tricks on using Group Functions, GROUP BY & HAVING clause

## Quick Guidelines

- All group functions except COUNT(\*) ignores NULL values.
- To substitute a value for NULL values use NVL functions.
- DISTINCT clause makes the function consider only non duplicate values.
- The AVG and SUM are used with numerica data.
- The MIN and MAX functions used with any data type.



NVL function is covered in the next lesson



## Quick Guidelines

- All individual columns included in the SELECT clause other than group functions must be specified in the GROUP BY clause.
- Any column other than selected column can also be placed in GROUP BY clause.
- By default rows are sorted by ascending order of the column included in the GROUP BY list.
- WHERE clause specifies the rows to be considered for grouping.

NVL function is covered in the next lesson



## Quick Guidelines

- Suppose your SELECT statement contains a HAVING clause. Then write your query such that the WHERE clause does most of the work (removing undesired rows) instead of the HAVING clause doing the work of removing undesired rows.  

- Use the GROUP BY clause only with an Aggregate function, and not otherwise.  

- Since in other cases, you can accomplish the same end result by using the DISTINCT option instead, and it is faster.

### Tips and Tricks:

- By appropriately using the WHERE clause, you can eliminate unnecessary rows before they reach the GROUP BY and HAVING clause. Thus saving some unnecessary work, and boosting performance.  
**For example:** In a SELECT statement with WHERE, GROUP BY, and HAVING clauses, the query executes in the following sequence.
  - First, the WHERE clause is used to select the appropriate rows that need to be grouped.
  - Next, the GROUP BY clause divides the rows into sets of grouped rows, and then aggregates their values.
  - And last, the HAVING clause then eliminates undesired aggregated groups.
    - If the WHERE clause is used to eliminate as many of the undesired rows as possible, then the GROUP BY and the HAVING clauses will have to do less work. Thus boosting the overall performance of the query.

contd.

**Tips and Tricks (contd.):**

- The GROUP BY clause can be used with or without an Aggregate function. However, if you want optimum performance, do not use the GROUP BY clause without an Aggregate function. This is because you can accomplish the same end result by using the DISTINCT option instead, and it is faster.

**For example:** You can write your query two different ways:

```
SELECT OrderID
FROM [Order Details]
WHERE UnitPrice > 10
GROUP BY OrderID
```

or

```
SELECT DISTINCT OrderID
FROM [Order Details]
WHERE UnitPrice > 10
```

Both of the above queries produce the same results, but the second one will use less resources and perform faster.

## Summary

- In this lesson, you have learnt about:
  - Aggregate (Group functions)
    - GROUP BY clause
    - HAVING clause



## Review – Questions

- Question 1: Identify the various group functions from the list given below:
  - Option 1: maximum
  - Option 2: sum
  - Option 3: count
  - Option 4: minimum



## Review – Questions

- Question 2: The AVG function ignores NULL values in the column.
  - True / False
  
- Question 3: Count(\*) returns the number of rows in the table, including duplicates and those with NULLs.
  - True / False

