







# **SQL Group BY**

### **Description :**

* The GROUP BY clause is used to arrange identical data into groups.
* This clause is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to perform an operation on each group of data.
* The GROUP BY statement comes after any WHERE clause in the SQL query and precedes the ORDER BY and HAVING clauses.

### **Syntax :**

SELECT column\_name(s), AGGREGATE\_FUNCTION(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

### **Example :**

#### **Show all the data from the table.**

SELECT \* FROM bigbasketproducts;

#### **Show all the unique brands from the table**

SELECT DISTINCT category FROM bigbasketproducts;

SELECT category

FROM bigbasketproducts

GROUP BY category;

#### **Calculate count of products in each category**

SELECT category , COUNT(\*)

FROM bigbasketproducts

GROUP BY category;

#### **Calculate count of procucts and Maximum price in each category**

SELECT category , COUNT(\*) , MAX(sale\_price)

FROM bigbasketproducts

GROUP BY category;

#### **Calculate count of procucts and Maximum price in each category and sub category**

SELECT category , sub\_category , COUNT(\*) , MAX(sale\_price)

FROM bigbasketproducts

GROUP BY category , sub\_category;

#### **Calculate the count of products in each category where rating is greater than - 4.**

SELECT category , COUNT(\*)

FROM bigbasketproducts

WHERE rating > 4

GROUP BY category;

# **SQL Group by with Having**

### **Description :**

* The HAVING clause is used in combination with the GROUP BY clause to filter groups based on a specified condition.
* It is different from the WHERE clause in that WHERE filters rows before the grouping takes place, and HAVING filters groups after the GROUP BY has been applied.

### **Syntax :**

SELECT column\_name(s), AGGREGATE\_FUNCTION(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

HAVING condition

ORDER BY column\_name(s);

### **Example :**

#### **Give all the brand where count of product is = 2**

SELECT brand , COUNT(\*)

FROM bigbasketproducts

GROUP BY brand

HAVING COUNT(\*) = 2;

#### **For each category and subcategory generate the count of products and max price , where rating is greater than 3.8 and Having count of products is greater than 3. Finally sort the data based on descending order of count.**

SELECT category , sub\_category , COUNT(\*) , MAX(sale\_price)

FROM bigbasketproducts

WHERE rating > 3.8

GROUP BY category , sub\_category

HAVING COUNT(\*) > 3

ORDER BY COUNT(\*) DESC;

# **Order of Execution in SQL Query**

### **Description :**

* Understanding the order of execution in an SQL query is crucial for both optimizing query performance and ensuring the accuracy of the results.
* SQL does not execute statements in the order they are written in the query.
* Instead, it follows a specific sequence to execute the various components of a query.

### **Order of Execution :**

#### **FROM**

The query execution starts with the FROM clause, where it identifies the base table(s) or view(s) from which to retrieve data.

#### **WHERE**

* After identifying and combining the necessary tables, the WHERE clause is applied to filter rows based on the specified condition.
* Only the rows that meet the condition are included in the result set moving forward.
* The WHERE clause operates on individual rows.

#### **GROUP BY**

* If the query includes a GROUP BY clause, the rows returned from the previous step are grouped into summary rows by the specified columns.
* This step is essential for aggregate calculations that follow.

#### **HAVING**

* The HAVING clause filters groups created by the GROUP BY clause.
* It is similar to the WHERE clause but operates on groups rather than on individual rows.
* This allows you to apply conditions on aggregate functions, like filtering groups based on the sum or average of a column.

#### **SELECT**

* The SELECT statement specifies the columns that the query will return.
* Although it appears at the beginning of the query, it is executed after the data has been filtered and grouped.
* If there are any expressions or functions in the SELECT clause, they are computed at this stage.

#### **DISTINCT**

* If the query includes the DISTINCT keyword, duplicate rows are removed from the results after the SELECT clause has been processed.
* DISTINCT applies to the entire result set, ensuring each row is unique.

#### **ORDER BY**

* The ORDER BY clause sorts the result set based on specified column(s) or expression(s).
* This is one of the last steps executed in a query. Note that sorting is computationally expensive, so this operation can affect query performance, especially on large datasets.

#### **LIMIT / OFFSET**

Finally, if there’s a LIMIT or OFFSET clause it is applied to the sorted result set to limit the number of rows returned, often used for pagination.

## **Resources - Official Documentation and Other Resources**

* SQL GROUP BY - <https://www.w3schools.com/sql/sql_groupby.asp>
* SQL Having - <https://www.w3schools.com/sql/sql_having.asp>
* SQL Order of Execution - <https://sqlbolt.com/lesson/select_queries_order_of_execution>

SHOW DATABASES;

USE db101;

SHOW TABLES;

-- ###################### Part - 1 ############################

-- See all the data from emp table

SELECT \* FROM emp;

-- I want to know all the unique department present

SELECT DISTINCT department FROM emp;

-- What if I want to calculate total count of emp in each department ?

-- 'Business Development' - 10

-- 'Sales' - 20

-- 'Marketing' - 30

-- 'Services' - 50

-- 'Training' - 100

-- 'Human Resources' - 40

-- 'Accounting' - 33

-- 'Engineering' - 55

-- 'Product Management' - 28

-- 'Legal' - 5

-- 'Support' - 25

-- 'Research and Development' - 5

SELECT department , COUNT(\*) AS dept\_count FROM emp

GROUP BY department;

-- What if I want to calculate sum of salary for each dept

SELECT department , SUM(salary) AS dept\_salary FROM emp

GROUP BY department;

-- What if I want to calculate avg of salary for each dept

-- Give me top 3 dept with lowest average salary.

SELECT department , AVG(salary) AS dept\_avg\_salary FROM emp

GROUP BY department ORDER BY dept\_avg\_salary LIMIT 3;

-- ###########################################################################

-- ACTIVITY :

SHOW TABLES;

SELECT \* FROM bigbasket;

-- For each brand how many count of product is present ?

SELECT brand , COUNT(\*) as product\_count FROM bigbasket

GROUP BY brand;

-- max sale price for each brand?

SELECT brand , COUNT(\*) as product\_count , MAX(sale\_price) FROM bigbasket

GROUP BY brand;

-- ###########################################################################

-- ###################### Part - 2 ############################

-- List all the unique department and gender

SELECT department , gender

FROM emp

GROUP BY department , gender

ORDER BY department;

SELECT DISTINCT department , gender FROM emp ORDER BY department;

-- For all unique department and gender calculate the count of emp

SELECT department , gender , COUNT(\*)

FROM emp

GROUP BY department , gender

ORDER BY department;

-- Also add maximum and minimum salary for each

SELECT department , gender , COUNT(\*) , MAX(salary) , MIN(salary)

FROM emp

GROUP BY department , gender

ORDER BY department;

-- ###########################################################################

-- ACTIVITY :

SELECT \* FROM bigbasket;

-- Calculate the count of products for each category and sub category ?

SELECT category , sub\_category , COUNT(\*) as product\_count FROM bigbasket

GROUP BY category , sub\_category;

-- ###########################################################################

-- ###################### Part - 3 ############################

-- I want all the department where average salary for that department is 62000 or less.

SELECT department , AVG(salary) AS dept\_avg\_salary FROM emp

GROUP BY department

HAVING dept\_avg\_salary <= 62000;

-- Lets use our mobile data

SELECT \* FROM flipkart\_mobiles;

-- For all out mobile brand calculate counts of mobile for each brand.

SELECT Brand , COUNT(\*)

FROM flipkart\_mobiles

GROUP BY Brand;

-- Give only those groupped data where mobile count is greater than 100.

SELECT Brand , COUNT(\*) AS mobile\_count

FROM flipkart\_mobiles

GROUP BY Brand

HAVING mobile\_count > 100;

-- ###########################################################################

-- ACTIVITY :

SELECT \* FROM bigbasket;

-- Calculate the count of products for each category and sub category where count of product is >= 10 ?

SELECT category , sub\_category , COUNT(\*) as product\_count FROM bigbasket

GROUP BY category , sub\_category HAVING product\_count >= 10;

-- ###########################################################################

-- ###################### Part - 4 ############################

SELECT \* FROM emp;

-- Give me count of only Female emp for each department.

SELECT department , COUNT(\*) FROM emp

WHERE gender = "Female"

GROUP BY department;

-- Give me count of only Female emp for each department Where count is > 30.

-- Finally sort the data in desc order and give 5 data from the top.

SELECT department , COUNT(\*) FROM emp

WHERE gender = "Female"

GROUP BY department

HAVING COUNT(\*) > 30

ORDER BY COUNT(\*) DESC

LIMIT 5;

SELECT \* FROM flipkart\_mobiles;

-- Calculate the count of mobile for each memory type Where Color contains black in it.

SELECT Memory , COUNT(\*) AS mobile\_count

FROM flipkart\_mobiles

WHERE LOWER(Color) LIKE "%black%"

GROUP BY Memory;

-- Difference between Having and Where.