SQL GROUP BY and HAVING Clause with Examples

What is the SQL Group by Clause?

The GROUP BY clause is a SQL command that is used to **group rows that have the same values**. The GROUP BY clause is used in the SELECT statement. Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.

That's what it does, **summarizing data** from the database.

The queries that contain the GROUP BY clause are called grouped queries and only return a single row for every grouped item.

SQL GROUP BY Syntax

Now that we know what the SQL GROUP BY clause is, let's look at the syntax for a basic group by query.

```
SELECT statements... GROUP BY
column_name1[,column_name2,...] [HAVING condition];
```

HERE

- "SELECT statements..." is the standard SQL SELECT command query.
- "**GROUP BY** *column_name1*" is the clause that performs the grouping based on column_name1.
- "[,column_name2,...]" is optional; it represents other column names when the grouping is done on more than one column.
- "[HAVING condition]" is optional; it is used to restrict the rows affected by the GROUP BY clause. It is similar to the WHERE clause.

Grouping using a Single Column

In order to help understand the effect of SQL Group By clause, let's execute a simple query that returns all the gender entries from the members table.

```
SELECT `gender` FROM `members` ;
```

gender
Female
Female
Male
Female
Male
Male
Male
Male
Male

Suppose we want to get the unique values for genders. We can use a following query -

```
SELECT `gender` FROM `members` GROUP BY `gender`;
```

Executing the above script in MySQL workbench against the Myflixdb gives us the following results.

gender

Female
Male

Note only two results have been returned. This is because we only have two gender types Male and Female. The GROUP BY clause in SQL grouped all the "Male" members together and returned only a single row for it. It did the same with the "Female" members.

Grouping using multiple columns

Suppose that we want to get a list of movie category_id and corresponding years in which they were released.

Let's observe the output of this simple query

```
SELECT `category_id`,`year_released` FROM `movies` ;
```

category_id	year_released
1	2011
2	2008
NULL	2008
NULL	2010
8	2007
6	2007
6	2007
8	2005

NULL	2012
7	1920
8	NULL
8	1920

The above result has many duplicates.

Let's execute the same query using group by in SQL -

```
SELECT `category_id`,`year_released` FROM `movies` GROUP BY `category_id`,`year_released`;
```

Executing the above script in MySQL workbench against the myflixdb gives us the following results shown below.

category_id	**year_released**
NULL	2008
NULL	2010
NULL	2012
1	2011
2	2008
6	2007
7	1920
8	1920

8	2005
8	2007

The GROUP BY clause operates on both the category id and year released to identify **unique** rows in our above example.

If the category id is the same but the year released is different, then a row is treated as a unique one .If the category id and the year released is the same for more than one row, then it's considered a duplicate and only one row is shown.

Grouping and aggregate functions

Suppose we want a total number of males and females in our database. We can use the following script shown below to do that.

```
SELECT `gender`,COUNT(`membership_number`)  FROM
`members` GROUP BY `gender`;
```

Executing the above script in MySQL workbench against the myflixdb gives us the following results.

gender	COUNT('membership_number')
Female	3
Male	5

The results shown below are grouped by every unique gender value posted and the number of grouped rows is counted using the COUNT aggregate function.

Restricting query results using the HAVING clause

It's not always that we will want to perform groupings on all the data in a given table. There will be times when we will want to restrict our results to a certain given criteria. In such cases , we can use the HAVING clause

Suppose we want to know all the release years for movie category id 8. We would use the following script to achieve our results.

```
SELECT * FROM `movies` GROUP BY `category_id`,`year_released` HAVING `category_id` = 8;
```

Executing the above script in MySQL workbench against the Myflixdb gives us the following results shown below.

movie_id	title	director	year_released	category_id
9	Honey mooners	John Schultz	2005	8
5	Daddy's Little Girls	NULL	2007	8

Note only movies with category id 8 have been affected by our GROUP BY clause.

Summary

- The GROUP BY Clause SQL is used to group rows with the same values.
- The GROUP BY Clause is used together with the SQL SELECT statement.

- The SELECT statement used in the GROUP BY clause can only be used to contain column names, aggregate functions, constants and expressions.
- SQL Having Clause is used to restrict the results returned by the GROUP BY clause.
- MYSQL GROUP BY Clause is used to collect data from multiple records and return records set by one or more columns.