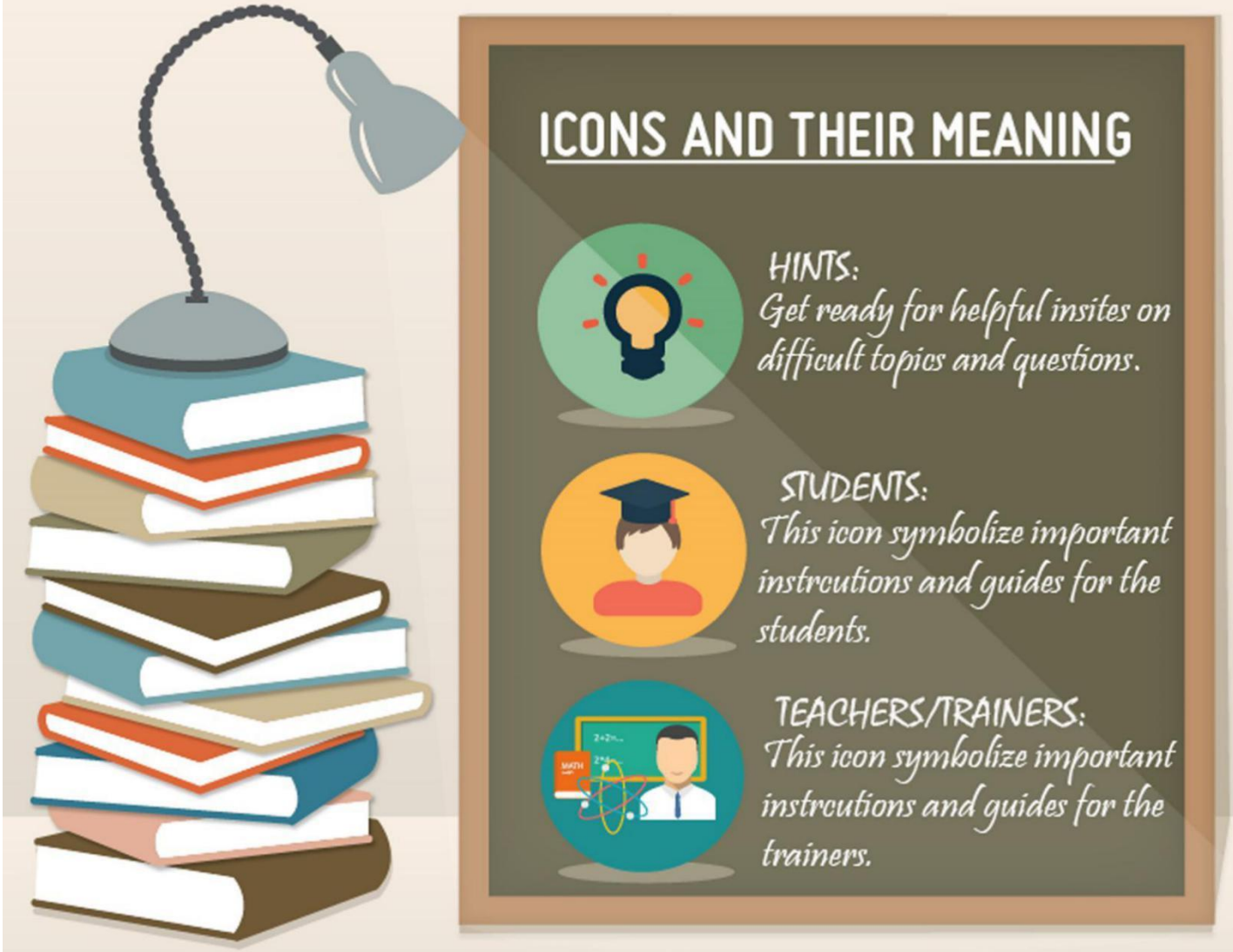
MySQL

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

GROUP BY, HAVING and ROLLUP in MySQL

Objective: After completing this lesson you will be able to : * Gain an understanding of GROUP BY, HAVING and ROLLUP in MySQL	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 120 minutes	**Practical Duration:** 0 minute
Total Duration: 120 minutes	

Chapter 20

MySQL GROUP BY clause

The MySQL GROUP BY CLAUSE is used for grouping a set of rows into a summary rows set by expression and column values. The GROUP BY clause returns one row for every group. The row count in the result set is reduced by this clause.

The GROUP BY clause is utilized with several aggregate functions like COUNT, SUM, AVG, MAX, and MIN. The aggregate function found within the SELECT clause offers information about each group. The clause is an optional clause belonging to the SELECT statement. GROUP BY clause syntax -

```
SELECT
    c1, c2,..., cn, aggregate_function(ci)
FROM
    table
WHERE
    where_conditions
GROUP BY c1 , c2,...,cn;
```

GROUP BY clause should be used after the FROM and WHERE clauses. A list of comma-separated columns or expressions follows GROUP BY keywords. These are used as the requirements for grouping rows.

The GROUP BY clause is evaluated by MySQL after the FROM, WHERE and SELECT clauses. MySQL evaluates the GROUP BY clause before the HAVING, ORDER BY and LIMIT clauses -

The order is -

FROM, WHERE, SELECT, GROUP BY, HAVING, ORDER BY, LIMIT

MySQL GROUP BY examples

GROUP BY clause example -

Simple MySQL GROUP BY example

Take a look at this sample database -

Use the GROUP BY clause with the status column if you intend to group order status values into subgroups -

```
SELECT
    status
```

```
FROM
    orders
GROUP BY status;
```

The GROUP BY clause returns unique instances of status values. See the example to understand that the GROUP BY clause functions similar to the DISTINCT operator -

```
SELECT DISTINCT
    status
FROM
    orders;
```

MySQL GROUP BY with aggregate functions

Aggregate functions let users calculate a set of rows and output a single value. The GROUP BY clause can be used along with an aggregate function to calculate each subgroup and return a single value.

Example - If a user intends to know the orders count in each status, the GROUP BY clause can be used with the COUNT function -

```
SELECT
    status, COUNT(*)
FROM
    orders
GROUP BY status;
```

Take a look the the orders and orderdetails tables given below.

The orders table can be joined with the orderdetails table to find the amount of orders by status. The SUM function can be used for calculating the total amount.

Example query -

```
SELECT
    status,
    SUM(quantityOrdered * priceEach) AS amount
FROM
```

```
  orders
INNER JOIN orderdetails
    USING (orderNumber)
GROUP BY
    status;
```

The below query returns the number of orders and the amount of each order -

```
SELECT
    orderNumber,
    SUM(quantityOrdered * priceEach) AS total
FROM
    orderdetails
GROUP BY
    orderNumber
```

MySQL GROUP BY with expression example

MySQL users can group by expressions as well as columns. Below is a query for fetching each year's total sales -

```
SELECT
    YEAR(orderDate) AS year,
    SUM(quantityOrdered * priceEach) AS total
FROM
    orders
INNER JOIN orderdetails
    USING (orderNumber)
WHERE
    status = 'Shipped'
GROUP BY
```

```
  YEAR(orderDate);
```

The YEAR function was utilized to extract the year data from orderDate. The total sales shows only orders with shipped status. The SELECT clause expression should be the same as the GROUP BY clause.

Example of using MySQL GROUP BY with HAVING clause

A HAVING clause can be used for filtering groups returned by the GROUP BY clause. The query below uses the HAVING clause for selecting the total sales after a particular year.

```
SELECT
    YEAR(orderDate) AS year,
    SUM(quantityOrdered * priceEach) AS total
FROM
    orders
INNER JOIN orderdetails
    USING (orderNumber)
WHERE
    status = 'Shipped'
GROUP BY
    Year
HAVING
    year > 2003;
```

Output -

The query below extracts year from order data.

```
SELECT
    YEAR(orderDate) AS year,
    COUNT(orderNumber)
FROM
    orders
GROUP BY
```

year;

Output -

MySQL enables users to sort groups in either ascending or descending order. By default, the order is ascending. If a user wants to fetch the number of orders by status, and sort it in a descending order, the GROUP BY clause can be used with DESC. Example -

```
SELECT
    status,
    COUNT(*)
FROM
    orders
GROUP BY
    status DESC;
```

Using ASC in the GROUP BY clause is necessary for sorting in ascending order.

HAVING

MySQL HAVING clause

The MySQL HAVING clause is utilized within the SELECT statement for specifying filter conditions for aggregates or row groups.

Groups can be filtered based on a specific condition by combining the HAVING clause with the GROUP BY clause. Removing the GROUP BY clause makes HAVING function similar to the WHERE clause.

HAVING clause syntax

```
SELECT
    select_list
FROM
    table_name
WHERE
    search_condition
GROUP BY
```

```
    group_by_expression
HAVING
    group_condition;
```

The HAVING clause specifies a condition. If the group_condition evaluates to true by a GROUP BY clause row, the query has the result set.

The HAVING clause applies a filter condition to each row group. The WHERE clause is used for applying the filter condition to each row.

The HAVING clause is evaluated before the ORDER BY, and LIMIT clauses, and after the FROM, WHERE, SELECT and GROUP BY clauses.

Examples of MySQL HAVING clause

For this example, a table named orderdetails within the sample database is used.

In the below example, the GROUP BY clause is used for fetching the order numbers, count of items sold per order, and the total number of sales for each, from the table named 'orderdetails'.

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY ordernumber;
```

The next part of the example is finding which order has sales more than 1000 with the HAVING clause -

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
```

```
GROUP BY
    ordernumber
HAVING
    total > 1000;
```

HAVING clause lets users create a complex condition with logical operators including AND and OR.

The example below shows how the HAVING clause can be utilized for finding orders with total amounts more than 1000, with upwards of 600 items -

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemsCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY ordernumber
HAVING
    total > 1000 AND
    itemsCount > 600;
```

Output -

The INNER JOIN clause can be used to join the orders and orderdetails tables. It also helps to apply a status column condition and total aggregate. In the example below, the user attempts to find orders with shipped status and with total amount more than 1500 -

```
SELECT
    a.ordernumber,
    status,
    SUM(priceeach*quantityOrdered) total
FROM
    orderdetails a
```

```
INNER JOIN orders b
    ON b.ordernumber = a.ordernumber
GROUP BY
    ordernumber,
    status
HAVING
    status = 'Shipped' AND
    total > 1500;
```

Output -

The HAVING clause can only be utilized if it's used with the GROUP BY clause, for generating report outputs. The HAVING clause can be utilized for answering questions like fetching orders by a given time period i.e. this week, this month, this year with sales more than 8k.

Rollup

MySQL ROLLUP is an SQL keyword type used along with the GROUP BY clause for the purpose of creating subtotals and grand total for column result sets, as a summary row. The ROLLUP operator is used with the GROUP BY clause as an advanced feature for sum total filtering of single or multiple columns, by the addition of more rows.

The GROUP BY clause is applied with aggregate functions such as MAX, MIN, AVG, SUM and COUNT which groups rows by single or multiple columns.

Multiple grouping of set rows can be created by utilizing a single query that has both the GROUP BY clause and its ROLLUP extension.

Syntax:

```
SELECT r1, r2 AggregateFunction(r3) FROM TableName GROUP BY r1, r2 WITH ROLLUP;
```

ROLLUP is based on a hierarchy of input columns.

Hence, by providing the (r1, r2), MySQL assumes that r1>r2. ROLLUP generates a column group that is based on the r1>r2 hierarchy. The ROLLUP clause is used for generating MySQL reports for creating subtotals and grand totals.

Examples of ROLLUP in MySQL

In this example, the Database PersonDbb has the Customer_Data table with several columns.

Customer_Data table

Example -

ROLLUP with one column

A query for retrieving the total salary by the Customer name parameter with the GROUP BY clause and MySQL's SQL SUM () aggregate function.

SELECT Name, SUM(Salary) FROM `customer_data` GROUP BY Name;

Output:

In the next step, a query is made for fetching the total amount in salary column by utilizing the ROLLUP operator.

SELECT Name, SUM(Salary) FROM `customer_data` GROUP BY Name WITH ROLLUP;

Output:

The NULL value column specifies the grand total line. ROLLUP adds an extra row for displaying the grand total aggregate of all the salary amounts.

SQL COALESCE() function can be used for making the result row more informative.

SELECT COALESCE(Name, 'All Customers ') AS Name, SUM(Salary) FROM `customer_data` GROUP BY Name WITH ROLLUP;

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. The MySQL GROUP BY clause is used for grouping rows by expression and _____ values.

a) row

b) array

c) column

d) none of the mentioned

2. Which of these functions do not use the GROUP BY clause?

a) MAX

b) COUNT

c) both a and b

d) none of the mentioned

3. The GROUP BY clause is an option found within a _____ statement.

a) CREATE

b) SELECT

c) ALTER

d) none of the mentioned

4. Which of these clauses is the GROUP BY statement not used after?

a) FROM

b) WHERE

c) both a and b

d) none of the mentioned

5. The GROUP BY clause is evaluated before the _____ clause.

a) HAVING

b) ORDER BY

c) both a and b

d) none of the mentioned

6. The GROUP BY clause can be used with an aggregate function to calculate _____.

a) each group

b) each subgroup

c) two groups simultaneously

d) none of the mentioned

7. HAVING clause is used for _____ returned by the GROUP BY clause.

a) filtering

b) returning

c) omitting

d) none of the mentioned

8. Without a GROUP BY clause the HAVING clause function similar to the _____.

a) SELECT clause

b) WHERE clause

c) FROM clause

d) none of the mentioned

9. The HAVING clause is evaluated before the _____ and GROUP BY clauses.

a) WHERE

b) LIMIT

c) SELECT

d) none of the mentioned

10. MySQL ROLLUP is used with the GROUP BY clause for creating _____

a) averages

b) subtotals

c) percentages

d) none of the mentioned