The GROUP BY clause is used with SELECT statements. It is used to form subsets in case of identical data. Usually, this clause is followed by ORDER BY clause and placed after the WHERE clause.

Syntax

Following is the syntax of GROUP BY clause −

ij>SELECT column1, column2, . . . table\_name GROUP BY column1, column2, . . .;

Example

Suppose we have a table named Employees in the database with the following records −

ID |NAME |SALARY |LOCATION

------------------------------------------------------------------

1 |Amit |30000 |Hyderabad

2 |Rahul |39000 |Lucknow

3 |Renuka |50000 |Hyderabad

4 |Archana |15000 |Vishakhapatnam

5 |Kalyan |40000 |Hyderabad

6 |Trupthi |45000 |Vishakhapatnam

7 |Raghav |12000 |Lucknow

8 |Suchatra |33000 |Vishakhapatnam

9 |Rizwan |20000 |Lucknow

The following SELECT statement with GROUP BY clause groups the table based on location. It displays the total amount of salary given to employees at a location.

ij> SELECT Location, SUM(Salary) from Employees GROUP BY Location;

This will generate the following output −

LOCATION |2

-------------------------------------------------------

Hyderabad |120000

Lucknow |71000

Vishakhapatnam |93000

3 rows selected

In the same way, following query finds the average amount spent on the employees as salary in a location.

ij> SELECT Location, AVG(Salary) from Employees GROUP BY Location;

This will generate the following output −

LOCATION |2

-----------------------------------------------------

Hyderabad |40000

Lucknow |23666

Vishakhapatnam |31000

3 rows selected

Group By clause JDBC example

This section teaches you how to use Group By clause and perform CURD operations on a table in Apache Derby database using JDBC application.

If you want to request the Derby network server using network client, make sure that the server is up and running. The class name for the Network client driver is org.apache.derby.jdbc.ClientDriver and the URL is jdbc:derby://localhost:1527/**DATABASE\_NAME;**create=true;user=**USER\_NAME;**passw ord=**PASSWORD**"

Follow the steps given below to use Group By clause and perform CURD operations on a table in Apache Derby

Step 1: Register the driver

To communicate with the database, first of all, you need to register the driver. The **forName()** method of the class **Class** accepts a String value representing a class name loads it in to the memory, which automatically registers it. Register the driver using this method.

Step 2: Get the connection

In general, the first step we do to communicate to the database is to connect with it. The **Connection** class represents the physical connection with a database server. You can create a connection object by invoking the **getConnection()** method of the **DriverManager** class. Create a connection using this method.

Step 3: Create a statement object

You need to create a **Statement** or **PreparedStatement** or, **CallableStatement** objects to send SQL statements to the database. You can create these using the methods **createStatement(), prepareStatement() and, prepareCall()** respectively. Create either of these objects using the appropriate method.

Step 4: Execute the query

After creating a statement, you need to execute it. The **Statement** class provides various methods to execute a query like the **execute()** method to execute a statement that returns more than one result set. The **executeUpdate()** method is used to execute queries like INSERT, UPDATE, DELETE. The **executeQuery()** method returns data. Use either of these methods and execute the statement created previously.

Example

Following JDBC example demonstrates how to use **Group By** clause and perform CURD operations on a table in Apache Derby using JDBC program. Here, we are connecting to a database named sampleDB (will create if it does not exist) using the embedded driver.

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

import java.sql.ResultSet;

public class GroupByClauseExample {

public static void main(String args[]) throws Exception {

//Registering the driver

Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

//Getting the Connection object

String URL = "jdbc:derby:sampleDB;create=true";

Connection conn = DriverManager.getConnection(URL);

//Creating the Statement object

Statement stmt = conn.createStatement();

//Creating a table and populating it

stmt.execute("CREATE TABLE EmployeesData( "

+ "Id INT NOT NULL GENERATED ALWAYS AS IDENTITY, "

+ "Name VARCHAR(255), "

+ "Salary INT NOT NULL, "

+ "Location VARCHAR(255), "

+ "PRIMARY KEY (Id))");

stmt.execute("INSERT INTO EmployeesData(Name, Salary, Location) "

+ "VALUES ('Amit', 30000, 'Hyderabad'), "

+ "('Rahul', 39000, 'Lucknow'), "

+ "('Renuka', 50000, 'Hyderabad'), "

+ "('Archana', 15000, 'Vishakhapatnam'), "

+ "('Kalyan', 40000, 'Hyderabad'), "

+ "('Trupthi', 45000, 'Vishakhapatnam'), "

+ "('Raghav', 12000, 'Lucknow'), "

+ "('Suchatra', 33000, 'Vishakhapatnam'), "

+ "('Rizwan', 20000, 'Lucknow')");

//Executing the query

String query = "SELECT Location, SUM(Salary) from EmployeesData GROUP BY Location";

ResultSet rs = stmt.executeQuery(query);

while(rs.next()) {

System.out.println("Location: "+rs.getString(1));

System.out.println("Sum of salary: "+rs.getString(2));

System.out.println(" ");

}

}

}

Output

On executing the above program, you will get the following output −

Location: Hyderabad

Sum of salary: 120000

Location: Lucknow

Sum of salary: 71000

Location: Vishakhapatnam

Sum of salary: 93000