The HAVING Clause enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

Syntax

Following is the syntax of the HAVING clause −

ij> SELECT column1, column2 . . . from table\_name GROUP BY column having

condition;

Example

Suppose, we have a table named Employees in the database with the 13 records as shown below −

ID |NAME |SALARY |LOCATION

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

1 |Amit |30000 |Hyderabad

2 |Rahul |39000 |Lucknow

3 |Kalyan |40000 |Vishakhapatnam

4 |Renuka |50000 |Hyderabad

5 |Archana |15000 |Vishakhapatnam

6 |Krishna |40000 |Hyderabad

7 |Trupthi |45000 |Vishakhapatnam

8 |Raghav |12000 |Lucknow

9 |Radha |50000 |Delhi

10 |Anirudh |15000 |Mumbai

11 |Tara |45000 |Kochin

12 |Sucharita |44000 |Kochin

13 |Rizwan |20000 |Lucknow

The following query displays maximum salaries of the employees in the location which have at least 3 employees −

ij> SELECT Location, MAX(Salary) from Employees GROUP BY Location having

count(Location)>=3;

This generates the following output −

LOCATION |2

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

Hyderabad |50000

Lucknow |39000

Vishakhapatnam |45000

3 rows selected

Sorting Data using JDBC program

This section teaches you how to use Having a clause in the 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 sort the records of 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, and 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 executes 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 HavingClauseExample {

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, MAX(Salary) "

+ "from EmployeesData GROUP BY Location having "

+ "count(Location)>=3";

ResultSet rs = stmt.executeQuery(query);

while(rs.next()) {

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

System.out.println(" ");

}

}

}

Output

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

Hyderabad

50000

Lucknow

39000

Vishakhapatnam

45000