

ASSIGNMENT 11.3

Explain in brief different types of groups in hive with an example

JOINS

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database. It is more or less similar to SQL JOIN.

Syntax

join_table:

```
table_reference JOIN table_factor [join_condition]
| table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference
join_condition
| table_reference LEFT SEMI JOIN table_reference join_condition
| table_reference CROSS JOIN table_reference [join_condition]
```

Example

We will use the following two tables in this chapter. Consider the following table named CUSTOMERS..

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Consider another table ORDERS as follows:

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

Types of joins

There are different types of joins given as follows:

JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

JOIN

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
FROM CUSTOMERS c JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AGE	AMOUNT
----	------	-----	--------

3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

+---+-----+-----+-----+

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
LEFT OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL

6 Komal NULL NULL
7 Muffy NULL NULL
+---+-----+-----+-----+

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
notranslate"> hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID =
o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

+---+-----+-----+-----+
ID NAME AMOUNT DATE
+---+-----+-----+-----+
3 kaushik 3000 2009-10-08 00:00:00
3 kaushik 1500 2009-10-08 00:00:00
2 Khilan 1560 2009-11-20 00:00:00
4 Chaitali 2060 2008-05-20 00:00:00
+---+-----+-----+-----+

FULL OUTER JOIN

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
FULL OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

GROUP BY CLAUSE

The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

Syntax

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[LIMIT number];
```

Example

Let us take an example of SELECT...GROUP BY clause. Assume employee table as given below, with Id, Name, Salary, Designation, and Dept fields. Generate a query to retrieve the number of employees in each department.

ID	Name	Salary	Designation	Dept
1201	Gopal	45000	Technical manager	TP
1202	Manisha	45000	Proofreader	PR
1203	Masthanvali	40000	Technical writer	TP
1204	Krian	45000	Proofreader	PR
1205	Kranthi	30000	Op Admin	Admin

The following query retrieves the employee details using the above scenario.

```
hive> SELECT Dept,count(*) FROM employee GROUP BY DEPT;
```

On successful execution of the query, you get to see the following response:

```
+-----+
```

Dept	Count(*)
Admin	1
PR	2
TP	3

JDBC Program

Given below is the JDBC program to apply the Group By clause for the given example.

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;
```

```
public class HiveQLGroupBy {
    private static String driverName =
"org.apache.hadoop.hive.jdbc.HiveDriver";
```

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

```
        // Register driver and create driver instance
        Class.forName(driverName);
```

```
        // get connection
        Connection con = DriverManager.
getConnection("jdbc:hive://localhost:10000/userdb", "", "");
```

```
        // create statement
        Statement stmt = con.createStatement();
```

```
        // execute statement
```

```

    Resultset res = stmt.executeQuery("SELECT Dept,count(*) " +
    "FROM employee GROUP BY DEPT; ");
    System.out.println(" Dept \t count(*)");

    while (res.next()) {
        System.out.println(res.getString(1) + " " + res.getInt(2));
    }
    con.close();
}
}

```

Save the program in a file named HiveQLGroupBy.java. Use the following commands to compile and execute this program.

```
$ javac HiveQLGroupBy.java
```

```
$ java HiveQLGroupBy
```

Output:

Dept	Count(*)
Admin	1
PR	2
TP	3