**Querying Data-Sorting**

By using HiveQL ORDER BY and SORT BY clause, we can apply sort on the column. It returns the result set either in ascending or descending order. Here, we are going to execute these clauses on the records of the below table:

**HiveQL - ORDER BY Clause**

In HiveQL, The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

hive> select \* from emp order by salary desc;

**HiveQL - 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.

**hive>** select \* from emp sort by salary desc;

**hive>** SELECT Id, Name, Dept FROM employee ORDER BY DEPT;

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

**HiveQL - Select-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.

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.

**hive>** SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM CUSTOMERS c JOIN ORDERS o ON (c.ID = o.CUSTOMER\_ID);

**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.

**hive>** SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c

LEFT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER\_ID);

## 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.

**hive>** SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER\_ID);

## 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);

**Views**

A view allows a query to be saved and treated like a table. It is a logical construct, as it does not store data like a table.

When a query becomes long or complicated, a view may be used to hide the complexity by dividing the query into smaller, more manageable pieces; similar to writing a function in a programming language or the concept of layered design in software. Encapsulating the complexity makes it easier for end users to construct complex queries from reusable parts.

hive> CREATE VIEW test AS

SELECT \* FROM emp\_list

WHERE salary>1000;

Hive>DROP VIEW view\_name

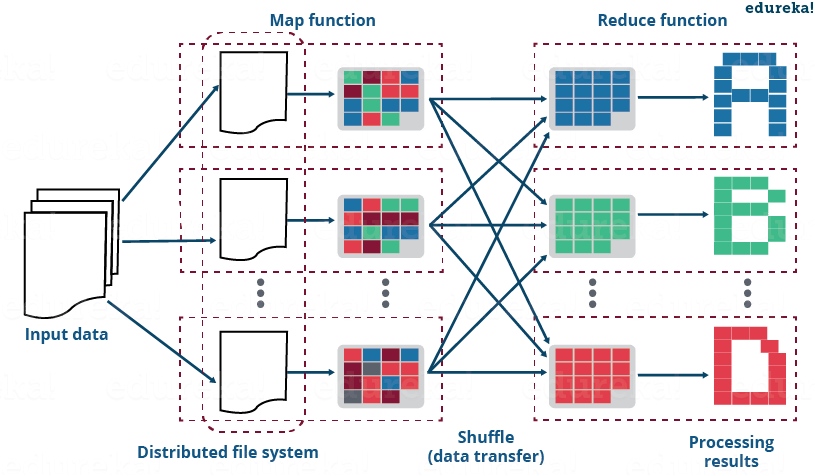
**What is a Join?**

The join operation is used to combine two or more database tables based on foreign keys. In general, companies maintain separate tables for the customer and the transaction records in their database. And, many times these companies need to generate analytic reports using the data present in such separate tables. Therefore, they perform a join operation on these separate tables using a common column (foreign key), like customer id, etc., to generate a combined table. Then, they analyze this combined table to get the desired analytic reports.

## **Joins in MapReduce**

Just like SQL join, we can also perform join operations in MapReduce on different data sets. There are two types of join operations in MapReduce:

* **Map Side Join:** As the name implies, the join operation is performed in the map phase itself. Therefore, in the map side join, the mapper performs the join and it is mandatory that the input to each map is partitioned and sorted according to the keys.
* **Reduce Side Join:**As the name suggests, in the reduce side join, the reducer is responsible for performing the join operation. It is comparatively simple and easier to implement than the map side join as the sorting and shuffling phase sends the values having identical keys to the same reducer and therefore, by default, the data is organized for us.



**Data manipulation with Hive,**

Hive DML (Data Manipulation Language) commands are used to insert, update, retrieve, and delete data from the Hive table once the table and database schema has been defined using Hive DDL commands.

The various Hive DML commands are:

1. LOAD

*LOAD DATA*to the student hive table with the help of the below command.

**hive> LOAD DATA LOCAL INPATH './usr/Desktop/test1.txt' OVERWRITE INTO TABLE Employee;**

1. SELECT
2. INSERT
3. DELETE
4. UPDATE
5. EXPORT
6. IMPORT

**UDFs (User Defined Functions)**

In Hive, the users can define own functions to meet certain client requirements. They can be create by using HIVE API.

**Appending data into existing Hive table**

**Step1:** Load the data of a file into table using load command.

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename.

**Step2:** You can insert new data into table by using select query.

INSERT INTO table tablename1 select columnlist FROM secondtable;