# Experiment:- 9 RollNo:-

Aim: Install hive and use hive to create database and tables

A) Create and Drop databases

B) Create, Alter and Drop tables

C)Insert, Update and Delete

recordsWhat is Hive

Hive is a data warehouse infrastructure tool process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

A relational database

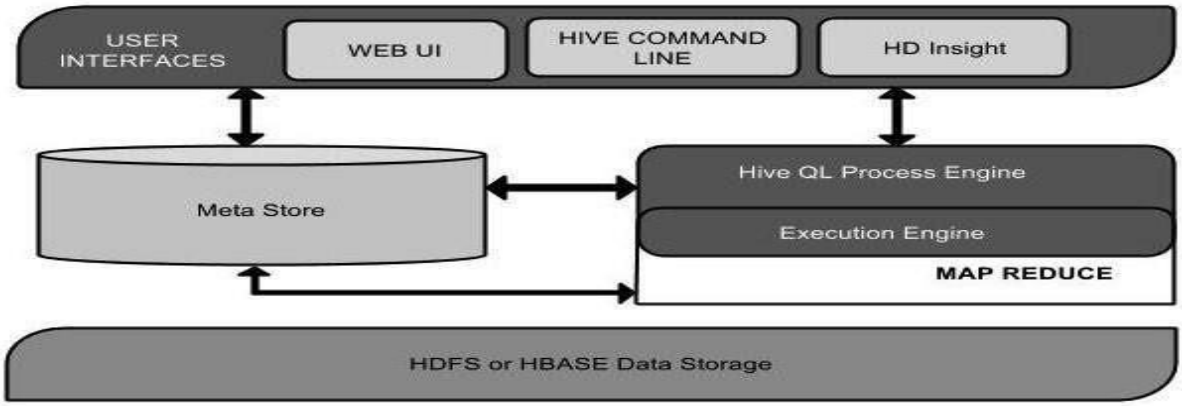
A design for On Line Transaction Processing (OLTP)

A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Architecture of Hive




DATA TYPES

All the data types in Hive are classified into four types, given as follows:

- Column Types
- Literals
- Null Values
- Complex Types

## Column Types:-

Column type are used as column data types of Hive. They are as follows:

Integral Types

Integer type data can be specified using integral data types, INT. When the data range exceeds the range of INT, you need to use BIGINT and if the data range is smaller than the INT, you use SMALLINT. TINYINT is smaller than SMALLINT.

| Type | Postfix | Example |
|---|---|---|
| TINYINT | Y | 10Y |
| SMALLINT | S | 10S |
| INT | - | 10 |
| BIGINT | L | 10L |

### String Types:-

String type data types can be specified using single quotes (' ') or double quotes (" "). It containstwo data types: VARCHAR and CHAR. Hive follows C-types escape characters.

| Data Type | Length |
|---|---|
| VARCHAR | 1 to 65355 |
| CHAR | 255 |

### Timestamp:-

It supports traditional UNIX timestamp with optional nanosecond precision. It supports java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" and format "yyyy-mm-dd hh:mm:ss.fffffffff".

Dates

DATE values are described in year/month/day format in the form {{YYYY-MM-DD}}.

Literals

The following literals are used in Hive:

### Floating Point Types:-

Floating point types are nothing but numbers with decimal points. Generally, this type of data is composed of DOUBLE data type.

### Decimal Type:-

Decimal type data is nothing but floating point value with higher range than DOUBLE data type. The range of decimal type is approximately $-10^{-308}$ to $10^{308}$

## Null Value

Missing values are represented by the special value NULL.

## Complex Types

The Hive complex data types are as follows:

Arrays
Maps
Structs

## Create Databases:-

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables. The **syntax** for this statement is as follows:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

```
hive> CREATE DATABASE [IF NOT EXISTS] student;
hive> CREATE SCHEMA stident;
```

O/P:-

```
hive> /*
OK
Time taken: 0.337 seconds
```

The following query is used to verify a databases list:

```
hive> show databases;
o/p:
OK
default
student
Time taken: 0.384 seconds
```

## Drop Database

## Use database:

Syntax: use <database-name>

```
Hive> use student;
OK
Time taken: 0.02 seconds
```

## Create Table:-

Create Table is a statement used to create a table in Hive. The syntax and example are asfollows:

Example:

```
CREATE TABLE IF NOT EXISTS employee ( eid int, name String,
salary String, destination String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

```
hive> create table stu(sno int, name string, branch string);
OK
Time taken: 0.507 seconds
```

## Show Schema

Syntax: describe table-name

```
hive> describe stu1;
OK
sno     int
name    string
branch string
Time taken: 0.413 seconds
```

## Load Data Statement [INSERT DATA]

Generally, after creating a table in SQL, we can insert data using the Insert statement. But inHive, we can insert data using the LOAD DATA statement.

While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There aretwo ways to load data: one is from local file system and second is from Hadoop file system.

Syntax:-

The syntax for load data is as follows:

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

- LOCAL is identifier to specify the local path. It is optional.
- OVERWRITE is optional to overwrite the data in the table.
- PARTITION is optional.

Example:-

We will insert the following data into the table. It is a text file named **sample.txt** in **/home/user** director.

```
[cloudera@localhost ~]$ vi sample.txt
[cloudera@localhost ~]$ cat sample.txt
```

```
501 Ravi CSE
502 Rani CSE
1201 Raja IT
1202 Roja IT
4201 vinay AI&ML

hive> load data local inpath '/home/cloudera/sample.txt' overwrite into table stu1;
Copying data from file:/home/cloudera/sample.txt
Copying file: file:/home/cloudera/sample.txt
Loading data to table default.stu1
rmr: DEPRECATED: Please use 'rm -r' instead.
Moved: 'hdfs://localhost.localdomain:8020/user/hive/warehouse/stu1' to trash at:
hdfs://localhost.localdomain:8020/user/cloudera/.Trash/Current
chgrp: changing ownership of '/user/hive/warehouse/stu1': User does not belong to hive
Table default.stu1 stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size: 26,
raw_data_size: 0]
OK
Time taken: 0.343 seconds
hive> select * from stu1;
OK
501 Ravi CSE
502 Rani CSE
1201 Raja IT
1202 Roja IT
4201 vinay AI&ML
Time taken: 0.1 seconds
hive> [cloudera@localhost ~]$
```

## Alter Table Statement:-

It is used to alter a table in Hive.

Syntax-

The statement takes any of the following syntaxes based on what attributes we wish to modify ina table.

```
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
ALTER TABLE name DROP [COLUMN] column_name
ALTER TABLE name CHANGE column_name new_name new_type
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
```

### Rename To… Statement

The following query renames the table from **employee** to **emp**.

EXAMPLE-

```
hive> ALTER TABLE employee RENAME TO emp;
hive> alter table stu1 rename to student;
OK
Time taken: 0.558 seconds
hive> select * from student;
OK
501 Ravi CSE
502 Rani CSE
1201 Raja IT
1202 Roja IT
4201 vinay AI&ML
Time taken: 0.698 seconds
```

## Add Columns:-

The following query adds a column named dept to the employee table.

Example

```
hive> ALTER TABLE employee ADD COLUMNS (
dept STRING COMMENT 'Department name');
```

```
hive> alter table student add columns(total string);
OK
Time taken: 0.451 seconds
hive> describe student;
OK
sno     int
name    string
branch string
total    string
```

Time taken: 0.309 seconds.

## Replace Statement:-

The following query deletes all the columns from the **employee** table and replaces it with **emp** and **name** columns:

```
hive> ALTER TABLE employee REPLACE COLUMNS (
eid INT empid Int,
ename STRING name String);
```

## Drop Table Statement:-

The syntax is as follows:

```
DROP TABLE [IF EXISTS] table_name;
```

The following query drops a table named **employee**:

```
hive> DROP TABLE IF EXISTS employee;
```

# ChangeStatement:

# Example

```
hive> ALTER TABLE employee CHANGE name ename String;
```

```
hive> alter table student change sno rno int;
OK
Time taken: 0.179 seconds
hive> describe student;
OK
rno     int
name    string
branch string
total    string
Time taken: 0.096 seconds
```

## UPDATE

Use the UPDATE statement to modify data already written to Apache Hive. Depending on the condition specified in the optional WHERE clause, an UPDATE statement may affect every row in a table. You must have both the SELECT and UPDATE privileges to use this statement.

```
UPDATE tablename SET column = value [, column = value ...] [WHERE expression];
```

The UPDATE statement has the following limitations:

- The expression in the WHERE clause must be an expression supported by a Hive SELECT clause.
- Partition and bucket columns cannot be updated.
- Query vectorization is automatically disabled for UPDATE statements. However, updated tables can still be queried using vectorization.
- Subqueries are not allowed on the right side of the SET statement.

The following example demonstrates the correct usage of this statement:

```
UPDATE students SET rno = 4201 WHERE rno==501;
```

## DELETE

```
DELETE FROM tablename [WHERE expression];
```

```
DELETE FROM students WHERE rno=4201;
```

Use the DELETE statement to delete data already written to Apache Hive.

**Experiment 10:**

**Aim: Perform data processing operations using Hive**

a) **Sort and Aggregation of data**
b) **Joins**

## Program:

## Sorting of Data:

1. First enter into the hive environment in the VM and create a database named 'csebda'.

```
cloudera@localhost:~
File  Edit  View  Search  Terminal  Help
[cloudera@localhost ~]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo
n-0.10.0-cdh4.7.0.jar!/hive-log4j.properties
Hive history file=/tmp/cloudera/hive_job_log_d7ca8b59-e1b8-45c9-a6dd-b545fcff913
1 1736743160.txt
```

```
hive> create database csebda;
OK
Time taken: 0.025 seconds
hive>
```

2)Create a table in the database named 'student'.

```
hive> create table if not exists student(sno int,sname string,m1 int,m2 int,m3 i
nt)
    > comment 'student'
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.287 seconds
hive>
```

3)To insert the data into the table crate a text file in local environment and overwrite it into thetable.

```
cloudera@loca
File  Edit  View  Search  Terminal  Help
[cloudera@localhost ~]$ vi marks.txt
[cloudera@localhost ~]$ cat marks.txt
501     leena   10      20      30
502     vidhya  20      30      25
503     arjun   30      20      30
504     rahul   40      20      30
505     radha   20      40      25
[cloudera@localhost ~]$
```

```
hive> load data local inpath '/home/cloudera/marks.txt' overwrite into table stu
dent;
Copying data from file:/home/cloudera/marks.txt
Copying file: file:/home/cloudera/marks.txt
Loading data to table default.student
```

### Order by:

```
hive> select * from student;
OK
501     leena   10      20      30
502     vidhya  20      30      25
503     arjun   30      20      30
504     rahul   40      20      30
505     radha   20      40      25
Time taken: 0.222 seconds
hive>
```

```
hive> desc student;
OK
sno     int
sname   string
m1      int
m2      int
m3      int
Time taken: 0.16 seconds
hive>
```

**To arrange data in ascending order:**

```
hive> select * from student order by sno;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
```

**Output:**

```
96 Success
Total MapReduce CPU Time Spent: 880 msec
OK
501     leena   10      20      30
502     vidhya  20      30      25
503     arjun   30      20      30
504     rahul   40      20      30
505     radha   20      40      25
Time taken: 9.791 seconds
hive>
```

**To arrange data in descending order:**

```
hive> select * from student order by sno desc;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

**Output:**

```
Total MapReduce CPU Time Spent: 790 msec
OK
505     radha   20      40      25
504     rahul   40      20      30
503     arjun   30      20      30
502     vidhya  20      30      25
501     leena   10      20      30
Time taken: 11.44 seconds
hive>
```

**Sort by:**

**To arrange data in ascending order:-**

```
hive> select * from student sort by sname asc;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
```

Output:-

```
Total MapReduce CPU Time Spent: 810 msec
OK
503     arjun   30      20      30
501     leena   10      20      30
505     radha   20      40      25
504     rahul   40      20      30
502     vidhya  20      30      25
Time taken: 10.413 seconds
```

**To arrange data in descending order**:

```
hive> select * from student sort by sname desc;
Total MapReduce jobs = 1
Launching Job 1 out of 1
```

**Output:**

```
Total MapReduce CPU Time Spent: 740 msec
OK
502     vidhya  20      30      25
504     rahul   40      20      30
505     radha   20      40      25
501     leena   10      20      30
503     arjun   30      20      30
Time taken: 9.467 seconds
```

## Aggregate Functions:

**min():**

```
hive> select min(m1) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
```

**Output:**

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 0.77 sec
 3 SUCCESS
Total MapReduce CPU Time Spent: 770 msec
OK
10
Time taken: 10.344 seconds
hive>
```

**max():**

```
hive> select max(m1) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at
```

# Output:-

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative
 3 SUCCESS
Total MapReduce CPU Time Spent: 780 m
OK
40
Time taken: 8.348 seconds
```

**count(*):**

```
hive> select count(*) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
```

```
hive> select count(*) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
```

**Output:**

```
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 0.78 sec
 2 SUCCESS
Total MapReduce CPU Time Spent: 780 msec
OK
5
Time taken: 10.3 seconds
hive>
```

**sum():**

```
hive> select sum(m1) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
```

**Output:**

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulativ
 4 SUCCESS
Total MapReduce CPU Time Spent: 780
OK
120
Time taken: 10.33 seconds
hive>
```

**avg():**

```
hive> select avg(m1) from student;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

**Output:**

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU:
 5 SUCCESS
Total MapReduce CPU Time Spent: 870 msec
OK
24.0
Time taken: 10.295 seconds
hive>
```

**Joins:**

Create two tables named sales,products and insert the data into them.

```
[cloudera@localhost ~]$ vi sales.txt
[cloudera@localhost ~]$ cat sales.txt
ramesh  10
suresh  20
haresh  0
naresh  30
rakesh  30
[cloudera@localhost ~]$ cat products.txt
10      laptop
20      printer
30      cpu
40      desktop
50      mouse
```

## Sales Table:

```
[cloudera@localhost ~]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo
n-0.10.0-cdh4.7.0.jar!/hive-log4j.properties
Hive history file=/tmp/cloudera/hive_job_log_26b7b8ac-d398-4e54-b75c-b112699a40a
e_1211167408.txt
hive> create table sale(cname string,pid int)
    > comment 'sales'
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.285 seconds
```

```
hive> load data local inpath '/home/cloudera/sales.txt' overwrite into table sal
e;
Copying data from file:/home/cloudera/sales.txt
Copying file: file:/home/cloudera/sales.txt
Loading data to table default.sale
rmr: DEPRECATED: Please use 'rm -r' instead.
Moved: 'hdfs://localhost.localdomain:8020/user/hive/warehouse/sale' to trash at:
 hdfs://localhost.localdomain:8020/user/cloudera/.Trash/Current
chgrp: changing ownership of '/user/hive/warehouse/sale': User does not belong t
o hive
Table default.sale stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_s
ize: 49, raw_data_size: 0]
OK
Time taken: 0.844 seconds
```

```
hive> select *from sale;
OK
ramesh  10
suresh  20
haresh  0
naresh  30
rakesh  30
Time taken: 0.207 seconds
```

```
hive> select sales.*,products.* from sales join products on(sales.pid=products.prodid);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202311022151_0004, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_202311022151_0004
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_202311022151_0004
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2023-11-02 22:28:11,978 Stage-1 map = 0%,  reduce = 0%
2023-11-02 22:28:17,005 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.75 sec
2023-11-02 22:28:18,013 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.75 sec
2023-11-02 22:28:19,019 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.75 sec
2023-11-02 22:28:20,023 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.28 sec
2023-11-02 22:28:21,054 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.28 sec
2023-11-02 22:28:22,061 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.28 sec
MapReduce Total cumulative CPU time: 1 seconds 280 msec
Ended Job = job_202311022151_0004
MapReduce Jobs Launched:
Job 0: Map: 2  Reduce: 1   Cumulative CPU: 1.28 sec   HDFS Read: 549 HDFS Write: 58 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 280 msec
OK
ramesh  10      10      laptop
suresh  20      20      printer
naresh  30      30      cpu
Time taken: 12.737 seconds
hive>
```

```
hive> create table product(prodid int,pname string)
    > comment 'products'
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.053 seconds
```

```
hive> load data local inpath '/home/cloudera/products.txt' overwrite into table
product;
Copying data from file:/home/cloudera/products.txt
Copying file: file:/home/cloudera/products.txt
Loading data to table default.product
rmr: DEPRECATED: Please use 'rm -r' instead.
Moved: 'hdfs://localhost.localdomain:8020/user/hive/warehouse/product' to trash
at: hdfs://localhost.localdomain:8020/user/cloudera/.Trash/Current
chgrp: changing ownership of '/user/hive/warehouse/product': User does not belon
g to hive
Table default.product stats: [num_partitions: 0, num_files: 1, num_rows: 0, tota
l_size: 48, raw_data_size: 0]
OK
Time taken: 0.193 seconds
hive> select * from product;
OK
10      laptop
20      printer
30      cpu
40      desktop
50      mouse
Time taken: 0.077 seconds
```

# Outer Join:

## Left Outer Join:

```
hive> select sales.*,products.* from sales left outer join products on(sales.pid=products.prodid);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202311022151_0006, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_202311022151_0006
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_202311022151_0006
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2023-11-02 22:30:52,317 Stage-1 map = 0%,  reduce = 0%
2023-11-02 22:30:57,364 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.64 sec
2023-11-02 22:30:58,370 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.64 sec
2023-11-02 22:30:59,375 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.64 sec
2023-11-02 22:31:00,378 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.64 sec
2023-11-02 22:31:01,381 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.22 sec
2023-11-02 22:31:02,392 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.22 sec
MapReduce Total cumulative CPU time: 1 seconds 220 msec
Ended Job = job_202311022151_0006
MapReduce Jobs Launched:
Job 0: Map: 2  Reduce: 1   Cumulative CPU: 1.22 sec   HDFS Read: 549 HDFS Write: 92 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 220 msec
OK
rakesh 30       NULL    NULL    NULL
haresh  0       NULL    NULL
ramesh  10      10      laptop
suresh  20      20      printer
naresh  30      30      cpu
Time taken: 12.415 seconds
```

## Right Outer Join:-

```
hive> select sales.*,products.* from sales right outer join products on(sales.pid=products.prodid);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202311022151_0007, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_202311022151_0007
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_202311022151_0007
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2023-11-02 22:31:47,215 Stage-1 map = 0%,  reduce = 0%
2023-11-02 22:31:52,244 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.66 sec
2023-11-02 22:31:53,253 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.66 sec
2023-11-02 22:31:54,257 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.66 sec
2023-11-02 22:31:55,266 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.66 sec
2023-11-02 22:31:56,270 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.2 sec
2023-11-02 22:31:57,282 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.2 sec
MapReduce Total cumulative CPU time: 1 seconds 200 msec
Ended Job = job_202311022151_0007
MapReduce Jobs Launched:
Job 0: Map: 2  Reduce: 1   Cumulative CPU: 1.2 sec   HDFS Read: 549 HDFS Write: 90 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 200 msec
OK
ramesh  10      10      laptop
suresh  20      20      printer
naresh  30      30      cpu
NULL    NULL    40      desktop
NULL    NULL    50      mouse
Time taken: 12.432 seconds
```

## Full Outer Join:

```
hive> select sales.*,products.* from sales full outer join products on(sales.pid=products.prodid);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202311022151_0008, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_202311022151_0008
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_202311022151_0008
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2023-11-02 22:33:29,895 Stage-1 map = 0%,  reduce = 0%
2023-11-02 22:33:34,926 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.69 sec
2023-11-02 22:33:35,933 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.69 sec
2023-11-02 22:33:36,936 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.69 sec
2023-11-02 22:33:37,940 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.21 sec
2023-11-02 22:33:38,944 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.21 sec
2023-11-02 22:33:39,952 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.21 sec
MapReduce Total cumulative CPU time: 1 seconds 210 msec
Ended Job = job_202311022151_0008
MapReduce Jobs Launched:
Job 0: Map: 2  Reduce: 1   Cumulative CPU: 1.21 sec   HDFS Read: 549 HDFS Write: 124 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 210 msec
OK
rakesh 30       NULL    NULL    NULL
haresh  0       NULL    NULL
ramesh  10      10      laptop
suresh  20      20      printer
naresh  30      30      cpu
NULL    NULL    40      desktop
NULL    NULL    50      mouse
Time taken: 12.307 seconds
hive>
```