**JOIN IN HIVE:**

**cast(str\_column as int)**

**regexp\_replace("”", "") --> VL**

**Employee.txt**

id,name,age,address,salary

1,Ramesh,32,Delhi,80000.00

2,Kiran,23,Patna,60000.00

3,Shilpa,21,Ranchi,20000.00

4,Chandan,23,Patliputra,26000.00

5,Harsha,24,Chandighar,34000.00

6,Manohar,20,UP,180000.00

7,Mufty,22,Lucknow,40000.00

**Expense.txt**

Oid,date,id,expanse

104,2018-04-08 00:00:00,3,5000

102,2018-05-08 00:00:00,3,2500

103,2018-03-20 00:00:00,2,2560

105,2018-01-20 00:00:00,4,3060

CREATE TABLE emp(ID int,NAME varchar(10),AGE int,ADDRESS varchar(10),SALARY int)row format delimited FIELDS terminated BY ',' stored AS textfile;

Load data local inpath '/home/hadoop/Downloads/employee.txt' into table emp;

select \* from emp;

CREATE TABLE expense(OID int,DATE varchar(20),emp\_ID int,AMOUNT int)row format delimited FIELDS terminated BY ',' stored AS textfile;

Load data local inpath '/home/cloudera/Downloads/expense.txt' into table expense;

select \* from expense;

**JOIN OPERATION:**

The above JOIN is also known as INNER JOIN as the value which is available is both the tables gets joined. Missing values in either table gets discarded. From the figure above we can understand that the values which are common in both the tables get joined and hence the name Inner Join.

SELECT ref.ID, ref.NAME, ref.AGE, ref1.AMOUNT FROM **emp** ref JOIN **expense** ref1 ON (ref.ID = ref1.emp\_ID);

If you have a requirement where in you want to keep all the rows of the table a (present in left side) + the common rows of the table a & b, you can use left outer join.

**LEFT OUTER JOIN:**

SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key);

**RIGHT OUTER JOIN:**

This join will return all the rows from right hand side table along with the common rows present in both left and right table. In case there is no match, join operation will still return the row but with NULL values.

SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM emp c RIGHT OUTER JOIN expense o ON (c.ID = o.emp\_ID);

**FULL OUTER JOIN**

Full outer join combines all the records present in both left as well as right table based on the condition.In brief, you will not lose the data after performing the join operation.

SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM emp c FULL OUTER JOIN expense o ON (c.ID = o.emp\_ID);

**MAP JOIN**

**Dataset description of the first dataset is as follows:**

id,first\_name,last\_name,email,gender,ip\_address

**Dataset description of the second dataset is as follows:**

id,first\_name,last\_name

CREATE TABLE IF NOT EXISTS dataset1 (id int, first\_name String, last\_name String, gender String, marks int, attendance float) row format delimited fields terminated BY ',' tblproperties("skip.header.line.count"="1");

CREATE TABLE IF NOT EXISTS dataset2 (eid int, first\_name String, last\_name String) row format delimited fields terminated BY ',' tblproperties("skip.header.line.count"="1");

**Map Join**

1. By specifying the keyword, /\*+ MAPJOIN(b) \*/ in the join statement.

By setting the following property to true.

**set hive.auto.convert.join=true**

For performing Map-side joins, there should be two files, one is of larger size and the other is of smaller size. You can set the small file size by using the following property:

**set hive.mapjoin.smalltable.filesize=(default it will be 25MB)**

Now, let us perform Map-side joins and join the two datasets based on their IDs.

**SELECT /\*+ MAPJOIN(dataset2) \*/ s.first\_name, s.eid,dataset2.eid FROM student s JOIN dataset2 ON s.first\_name = dataset2.first\_name;**

As it is a Map-side join, the number of reducers will be set to 0 automatically.

**Bucket-Map join**

The constraint for performing Bucket-Map join is:

If tables being joined are bucketed on the join columns, and the number of buckets in one table is a multiple of the number of buckets in the other table, the buckets can be joined with each other.

**To perform bucketing, we need to have bucketed tables.**

**CREATE TABLE IF NOT EXISTS dataset1\_bucketed (eid int,first\_name String, last\_name String, email String, gender String, ip\_address String) clustered by(first\_name) into 4 buckets row format delimited fields terminated BY ',';**

**CREATE TABLE IF NOT EXISTS dataset2\_bucketed (eid int,first\_name String, last\_name String) clustered by(first\_name) into 8 buckets row format delimited fields terminated BY ',' ;**

**insert into dataset1\_bucketed select \* from dataset1;**

**insert into dataset2\_bucketed select \* from dataset2;**

**For performing Bucket-Map join, we need to set this property in the Hive shell.**

set hive.optimize.bucketmapjoin = true

**SELECT /\*+ MAPJOIN(dataset2\_bucketed) \*/ dataset1\_bucketed.first\_name,dataset1\_bucketed.eid, dataset2\_bucketed.eid FROM dataset1\_buc**

**Creating Index in Hive**

CREATE INDEX index\_name

ON TABLE table\_name (columns,....)

AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'

WITH DEFERRED REBUILD;

**create table olympic(athelete STRING,age INT,country STRING,year STRING,closing STRING,sport STRING,gold INT,silver INT,bronze INT,total INT) row format delimited fields terminated by '\t' stored as textfile;**

**#First Run this to create a file**

create database if not exists partitiondb;

use partitiondb;

create table employee(id int,first\_name string,

last\_name string,gender string,comapny string,salary int,bonus int,city string,

country string)

row format delimited fields terminated by ',';

load data local inpath '/home/hadoop/Downloads/Employee.csv'

overwrite into table employee;

**#Partition Table**

create table partition\_employee(id int,first\_name string,

last\_name string,gender string,comapny string,salary int,bonus int,city string)

partitioned by (country string)

row format delimited

fields terminated by ',';

**#Insert the data**

insert overwrite table employee

partition(country)

select \* from partitioned\_employee

where partitioned\_employee.country = 'China';