Explain the following in brief with an example.

● Map side Join

Map side join is a process where joins between two tables are performed in the Map phase without the involvement of Reduce phase. Map-side Joins allows a table to get loaded into memory ensuring a very fast join operation, performed entirely within a mapper and that too without having to use both map and reduce phases. In case your queries frequently run with small table joins , you might see a very substantial decrease in the time taken to compute the queries after usage of map-side joins.

Map Join

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

2. By setting the following property to true.

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:

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) \*/ dataset1.first\_name, dataset1.eid,dataset2.eid FROM dataset1 JOIN dataset2 ON dataset1.first\_name = dataset2.first\_name;

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

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

Now, let us create two tables to store these two datasets.

CREATE TABLE IF NOT EXISTS dataset1 ( eid int, first\_name String, last\_name String, email String, gender String, ip\_address String) 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");

● Reduce side Join

Reduce-Side joins are more simple than Map-Side joins since the input datasets need not to be structured. But it is less efficient as both datasets have to go through the MapReduce shuffle phase. the records with the same key are brought together in the reducer.

● 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\_bucketed JOIN dataset2\_bucketed ON dataset1\_bucketed.first\_name = dataset2\_bucketed.first\_name ;

● SMBM Join

If the tables being joined are sorted and bucketized on the join columns and have the same number of buckets, a sort-merge join can be performed. The corresponding buckets are joined with each other at the mapper.

For performing the SMB-Map join, we need to set the following properties:

Set hive.input.format=org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

To perform this join, we need to have the data in the bucketed tables sorted by the join column.

insert overwrite table dataset1\_bucketed select \* from dataset1 sort by first\_name;

The above command will overwrite the data in the old table and insert the data as per the query.

To perform SMB-Map join, we need to have the same number of buckets in both the tables with the bucketed column sorted.

Now, we will create another table for dataset2 having 4 buckets and will insert the data that is sorted by first\_name.

CREATE TABLE IF NOT EXISTS dataset2\_bucketed1 (eid int,first\_name String, last\_name String) clustered by(first\_name) into 4 buckets row format delimited fields terminated BY ',' ;

insert overwrite table dataset2\_bucketed1 select \* from dataset2 sort by first\_name;

tables with same number of buckets and the joined column sorted. The join query is as follows:

SELECT /\*+ MAPJOIN(dataset2\_sbucketed1) \*/dataset1\_bucketed.first\_name, dataset1\_bucketed.eid, dataset2\_bucketed1.eid FROM dataset1\_bucketed JOIN dataset2\_bucketed1 ON dataset1\_bucketed.first\_name = dataset2\_bucketed1.first\_name ;