1. **EXPLAIN THE DIFFRENCE BETWEEN "PARTITIONING" AND "BUCKETING"...when TO USE THESE CONCEPTS**

* **Partitioning –**Apache Hive organizes tables into partitions for grouping same type of data together based on a column or partition key. Each table in the hive can have one or more partition keys to identify a particular partition. Using partition we can make it faster to do queries on slices of the data.
* Hive Partitioning dividing the large amount of data into number pieces of folders based on table columns value.
* Hive Partition is often used for distributing load horizontally, this has performance benefit, and helps in organizing data in a logical fashion.

If you want to use Partition in hive then you should use **PARTITIONED BY** (COL1,COL2…etc) command while hive table creation.

* We can perform partition on any number of columns in a table by using hive partition concept.
* We can perform Hive Partitioning concept on Managed tables or External tables
* Partitioning is works better when the cardinality of the partitioning field is not too high.
* **Bucketing –**In Hive Tables or partition are subdivided into buckets based on the hash function of a column in the table to give extra structure to the data that may be used for more efficient queries.
* Hive bucketing is responsible for dividing the data into number of equal parts
* If you want to use bucketing in hive then you should use **CLUSTERED BY** (Col) command while creating a table in Hive
* We can perform Hive bucketing optimization only on one column only not more than one.
* We can perform Hive bucketing concept on Hive Managed tables or External tables
* The value of this column will be hashed by a user-defined number into buckets.
* *bucketing works well when the field has high cardinality and data is evenly distributed among buckets*

1. **HOW MANY PARTITIONING WE HAVE ...WHEN TO USE WHAT TYPE OF PARTITIONING**

There are 2 types of partitioning 1) Static partitioning and 2) Dynamic partitioning.

* 1. **Static Partitioning:**
     + Insert input data files individually into a partition table is Static Partition.
     + Usually when loading files (big files) into **Hive tables**static partitions are preferred.
     + Static Partition saves your time in loading data compared to dynamic partition.
     + You “statically” add a partition in the table and move the file into the partition of the table.
     + We can alter the partition in the static partition.
     + You can get the partition column value from the filename, day of date etc without reading the whole big file.
     + If you want to use the Static partition in the hive you should set property **set hive.mapred.mode = strict**This property set by default in hive-site.xml
     + Static partition is in Strict Mode.
     + You should use where clause to use limit in the static partition.
     + You can perform Static partition on Hive Manage table or external table.
  2. **Dynamic Partitioning**
     + Single insert to partition table is known as a dynamic partition.
     + Usually, dynamic partition loads the data from the non-partitioned table.
     + Dynamic Partition takes more time in loading data compared to static partition.
     + When you have large data stored in a table then the Dynamic partition is suitable.
     + If you want to partition a number of columns but you don’t know how many columns then also dynamic partition is suitable.
     + Dynamic partition there is no required where clause to use limit.
     + We can’t perform alter on the Dynamic partition.
     + You can perform dynamic partition on hive external table and managed table.
     + If you want to use the Dynamic partition in the hive then the mode is in non-strict mode.
     + Here are Hive dynamic partition properties you should allow

1. **HOW DO YOU HANDLE NULL VALUES IN HIVE**

If “empname” column date having null values we can filter data like “ empname is not null” or “LENGTH(TRIM(empname)) = 0” in where clause.

If you want to replace any other value while selecting columns

Select eid,CASE WHEN LENGTH(empname) > 0 THEN empname ELSE ‘ABCDE’ END AS empname,esal from employee

1. **WHEN TO CHOOSE "SMB" IN HIVE?**

When all the below conditions are met we can perform a Sort Merge Bucket Map Join and it will be performed at the map side.

* All join tables are bucketized, and each small table’s number of buckets can be divided by big table’s number of buckets.
* Bucket columns and Join columns are same
* Join columns are sorted

**Below properties should be set to enable this feature:**

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

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

1. **EXPLAIN A SCENARIO WHERE WE CAN MAKE USE OF HIVE MAP SIDE JOIN? ANY LIMITATION OVER THERE?**

**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

**LIMITATION:**

Map side joins will be suitable only when one of the tables on which you perform map-side join operation should be small enough to fit into memory. That means it is not suitable solution for the tables which are having huge amount of data.

=> If below property is set to true, the query we executed will be automatically perform map side joins

**Set hive.auto.convert.join = true;**

1. **WHAT ARE PERFORMANCE OPTIMIZATION TECHNIQUES OF HIVE?**

* Enable vectorization
* Store data in ORC format
* Use map side joins
* Use partitioning and bucketing
* Creating indexes
* Increase split size
* Use tez

set hive.execution.engine=tez;

* Use parallel execution

Set hive.exec.parallel = true;

Set hive.exec.parallel.thread.number = true;

1. **WHAT IS VECTORIZATION IN HIVE?**

Vectorized query execution is a Hive feature that greatly reduces the CPU usage for typical query operations like scans, filters, aggregates, and joins. A standard query execution system processes one row at a time. This involves long code paths and significant metadata interpretation in the inner loop of execution. Vectorized query execution streamlines operations by processing a block of 1024 rows at a time. Within the block, each column is stored as a vector (an array of a primitive data type). Simple operations like arithmetic and comparisons are done by quickly iterating through the vectors in a tight loop, with no or very few function calls or conditional branches inside the loop.

To execute the Vectorization is query, data must store in ORC format. By default vectorization execution is disabled, to enable vectorization following configurations need to run.

Set hive.vectorized.execution.enabled=true;

Set hive.vectorized.execution.reduce.enabled=true;

1. **EXPLAIN THE CONCEPT OF "SERDE" WITH AN EXAMPLE?**

SerDe is a short name for Serializer and Deserializer. The SerDe interface allows users to instruct Hive as to how a record should be processed. The Deserializer takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working

with, and turn it into something that Hive can write to HDFS or another supported system. SerDe is commonly used to map data with complex structures to a row in Hive tables.

Below illustration gives the position of Serializer and Deserializer during write and read operation.

Row object --> Serializer --> <key, value> --> OutputFileFormat --> HDFS files

HDFS files --> InputFileFormat --> <key, value> --> Deserializer --> Row object

1. **WHEN TO USE "HIVE" AND "HBASE"? WHAT DESIGN CONSTRAINTS MAKES THE DIFFERENCE?**

**?**

1. **DID YOU FACE ANY SCENARIO OF LONG RUNNING QURIES IN HIVE? HOW DID YOU RESOLVED THE SAME?**

**?**

1. **WHY WE NEED TO DO THE HIVE INTEGRATION WITH HBASE?**

Reasons to use Hive on HBase is that a lot of data sitting in HBase due to its usage in a real-time environment, but never used for analysis as there are less connectivity tools to HBase directly.

We will use storage handler mechanism to create hbase tables via hive. **HBaseStorageHandler** allows **Hive DDL** for managing table definitions in both Hive metastore and HBase’s catalog simultaneously and consistently.