**Problem Statement:**

Explain in brief with an example

● Bucketing

● Bucketing V/S Partitioning

● Sampling

**Bucketing**

Hive partition divides table into number of partitions and these partitions can be further subdivided into more manageable parts known as Buckets or Clusters.

The Bucketing concept is based on Hash function, which depends on the type of the bucketing column.

Records which are bucketed by the same column will always be saved in the same bucket.

Here, CLUSTERED BY clause is used to divide the table into buckets.

In Hive Partition, each partition will be created as directory. But in Hive Buckets, each bucket will be created as file.

Bucketing can also be done even without partitioning on Hive tables.

***Advantages of Bucketing:***

Bucketed tables allows much more efficient sampling than the non-bucketed tables. With sampling, we can try out queries on a section of data for testing and debugging purpose when the original data sets are very huge. Here, the user can fix the size of buckets according to the need.

Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. Since the data files are equal sized parts, map-side joins will be faster on the bucketed tables.

***For example,***

CREATE TABLE mytable (

name string,

city string,

employee\_id int )

PARTITIONED BY (year STRING, month STRING, day STRING)

CLUSTERED BY (employee\_id) INTO 256 BUCKETS

**Bucketing V/S Partitioning:**

**Partitioning** data is often used for distributing load horizontally, this has performance benefit, and helps in organizing data in a logical fashion. *Example*: if we are dealing with a large employee table and often run queries with WHERE clauses that restrict the results to a particular country or department. For a faster query response Hive table can be PARTITIONED BY (country STRING, DEPT STRING). Partitioning tables changes how Hive structures the data storage and Hive will now create subdirectories reflecting the partitioning structure like

.../employees/*country=ABC/DEPT=XYZ*.

If query limits for employee from country=ABC, it will only scan the contents of one directory country=ABC. This can dramatically improve query performance, but only if the partitioning scheme reflects common filtering. Partitioning feature is very useful in Hive, however, a design that creates too many partitions may optimize some queries, but be detrimental for other important queries. Other drawback is having too many partitions is the large number of Hadoop files and directories that are created unnecessarily and overhead to NameNode since it must keep all metadata for the file system in memory.

**Bucketing** is another technique for decomposing data sets into more manageable parts. For example, suppose a table using date as the top-level partition and employee\_id as the second-level partition leads to too many small partitions. Instead, if we bucket the employee table and use employee\_id as the bucketing column, the value of this column will be hashed by a user-defined number into buckets. Records with the same employee\_id will always be stored in the same bucket. Assuming the number of employee\_id is much greater than the number of buckets, each bucket will have many employee\_id. While creating table you can specify like CLUSTERED BY (employee\_id) INTO XX BUCKETS; where XX is the number of buckets. Bucketing has several advantages. The number of buckets is fixed so it does not fluctuate with data. If two tables are bucketed by employee\_id, Hive can create a logically correct sampling. Bucketing also aids in doing efficient map-side joins etc.

**Sampling**

Sampling is concerned with the selection of a subset of data from a large dataset to run queries and verify results. The dataset may be too large to run queries on the whole data. Therefore in development and testing phases it is a good idea to run queries on a sample of dataset.

**There are two types of sampling. They are**

* **Sampling by bucketing**
* **Block sampling**

**Sampling by Bucketing**

We can use TABLESAMPLE clause to bucket the table on the given column and get data from only some of the buckets.

    TABLESAMPLE (BUCKET x OUT OF y [ON colname])

Column name indicates the column to be used to bucket the data into y buckets[1-y]. All the rows which are in the bucket x are returned.

If the table is not bucketed on the column(s) used in sampling, TABLESAMPLE will scan the entire table and fetch the sample.

If the hive table is bucketed on some column(s), then we can directly use that column(s) to get a sample. In this case Hive need not read all the data to generate sample as the data is already organized into different buckets using the column(s) used in the sampling query. Hive will read data only from some buckets as per the size specified in the sampling query.

**Block Sampling**

Block sampling allows Hive to select at least n% data from the whole dataset. Sampling granularity is at the HDFS block size level. If HDFS block size is 64MB and n% of input size is only 10MB, then 64MB of data is fetched.

TABLESAMPLE (n PERCENT)