**Apache Hive**

Apache Hive is a Data warehouse system which is built to work on Hadoop. It is used to querying and managing large datasets residing in distributed storage,  It provides a mechanism to project structure onto the data in Hadoop and to query that data using a SQL-like language called HiveQL (HQL), that are similar to SQL statements. Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler.

Hive is initially developed at Facebook but now, it is an open source Apache project used by many organizations as a general-purpose, scalable data processing platform.

***Uses of Hive:***

* The Apache Hive distributed storage.
* Hive provides tools to enable easy data extract/transform/load (ETL)
* It provides the structure on a variety of data formats.
* By using Hive, we can access files stored in Hadoop Distributed File System (HDFS is used to querying and managing large datasets residing in) or in other data storage systems such as Apache HBase.

**Limitations of Hive:**

* Hive is not designed for Online transaction processing (OLTP ), it is only used for the Online Analytical Processing.
* Hive supports overwriting or apprehending data, but not updates and deletes.
* In Hive, sub queries are not supported.

**Why Hive is used inspite of Pig?**

The following are the reasons why Hive is used in spite of Pig’s availability:

* Hive-QL is a declarative language line SQL, PigLatin is a data flow language.
* Pig: a data-flow language and environment for exploring very large datasets.
* Hive a distributed data warehouse.

**Important characteristics of Hive:**

* Tools to enable easy data extract/transform/load (ETL)
* A mechanism to project structure on a variety of data formats
* Access to files stored either directly in HDFS or other data storage systems as HBase
* Query execution through MapReduce jobs.
* SQL like language called HiveQL that facilitates querying and managing large data sets residing in hadoop.
* In Hive, tables and databases are created first and then data is loaded into these tables.
* Hive as data warehouse designed for managing and querying only structured data that is stored in tables.
* Hadoop's programming works on flat files. So, Hive can use directory structures to "partition" data to improve performance on certain queries.
* A new and important component of Hive i.e. Metastore used for storing schema information. This Metastore typically resides in a relational database. We can interact with Hive using methods like
  + Web GUI
  + Java Database Connectivity (JDBC) interface
* Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language(HQL)
* Generally, HQL syntax is similar to the[SQL](https://www.guru99.com/sql.html)syntax that most data analysts are familiar with. The Sample query below display all the records present in mentioned table name.
  + Sample query : Select \* from <TableName>
* Hive supports four file formats those are TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).
* For single user metadata storage, Hive uses derby database and for multiple user Metadata or shared Metadata case Hive uses MYSQL.

**Some of the key points about Hive:**

* The major difference between HQL and SQL is that Hive query executes on Hadoop's infrastructure rather than the traditional database.
* The Hive query execution is going to be like series of automatically generated map reduce Jobs.
* Hive supports partition and buckets concepts for easy retrieval of data when the client executes the query.
* Hive supports custom specific UDF (User Defined Functions) for data cleansing, filtering, etc. According to the requirements of the programmers one can define Hive UDFs

##### **Limitations:**

* Hive is best suited for data warehouse applications, where a large data set is maintained and mined for insights, reports, etc.
* Hive does not provide record-level update, insert, nor delete.
* Hive queries have higher latency than SQL queries, because of start-up overhead for MapReduce jobs submitted for each hive query.
* As Hadoop is a batch-oriented system, Hive doesn’t support OLTP (Online Transaction Processing).
* Hive is close to OLAP (Online Analytic Processing) but not ideal since there is significant latency between issuing a query and receiving a reply, both due to the overhead of Mapreduce jobs and due to the size of the data sets Hadoop was designed to serve.
* If we need OLAP, we need to use NoSQL databases like HBase that can be integrated with Hadoop.

##### **Differences Between Hive and HBase:**

The key differences between Apache Hive and HBase are as follows:

* The Hive is a data warehousing infrastructure whereas HBase is a NoSQL database on top of Hadoop.
* Apache Hive queries are executed as MapReduce jobs internally whereas HBase operations run in a real-time on its database rather than MapReduce.

Hive is not a database but a data warehousing frame work. Hive doesn’t provide record level operations on tables.

HBase is a NoSQL Database and it provides record level updates, inserts and deletes to the table data.

HBase doesn’t provide a query language like SQL, but Hive is now integrated with  
HBase.

## Hive Vs Relational Databases:

By using Hive, we can perform some peculiar functionality that is not achieved in Relational Databases. For a huge amount of data that is in peta-bytes, querying it and getting results in seconds is important. And Hive does this quite efficiently, it processes the queries fast and produce results in second's time.

Let see now what makes Hive so fast.

**Some key differences between Hive and relational databases are the following;**

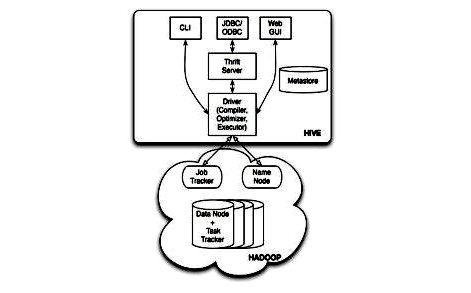
Relational databases are of "**Schema on READ and Schema on Write**". First creating a table then inserting data into the particular table. On relational database tables, functions like Insertions, Updates, and Modifications can be performed.

Hive is "**Schema on READ only**". So, functions like the update, modifications, etc. don't work with this. Because the Hive query in a typical cluster runs on multiple Data Nodes. So it is not possible to update and modify data across multiple nodes.( Hive versions below 0.13)

Also, Hive supports "**READ Many WRITE Once**" pattern. Which means that after inserting table we can update the table in the latest Hive versions.

**NOTE**: However, the new version of Hive comes with updated features. Hive versions (Hive 0.14) comes up with Update and Delete options as new features

**Below is the hive level architecture of Hive:**

[](http://hadooptutorial.info/wp-content/uploads/2014/09/hive-arch.jpg)

 In Hive distribution, we can find the below components majorly.

* **CLI :**  Command Line Interface. It is the most common way of interacting with Hive. (Hive shell) This is the default service.
* **HWI :** Hive Web Interface. It is an alternative to the shell for interacting with hive through web browser.
* **JDBC/ODBC/Thrift Server :** These are provide programmatic access to Hive server. Applications using Thrift, JDBC, and ODBC connectors need to run a Hive server to communicate with Hive. HIVE\_PORT environment variable need to be specified with the available port(defaults to 10,000) number to let the server listen on.
* **Driver:** Driver compiles the input commands and queries, optimizes the  
  computation required, and executes the required steps with MapReduce jobs.
* **Apache Hive Driver**: It is responsible for receiving the queries submitted through the CLI, the web UI, Thrift, ODBC or JDBC interfaces by a client. Then, the driver passes the query to the compiler where parsing, type checking, syntax of query and fetch the meta data of the table and validate the metadata weather that is matching with select query written, In the next step, converting the map-reduce tasks and HDFS tasks. Finally, the execution engine executes these tasks in the order of their dependencies Then the MapReduce jobs are executed  and gives the required result
* **Metastore:**You can think metastore as a central repository for storing all the Hive metadata information. Hive metadata includes various types of information like structure of tables and the partitions along with the column, column type, serializer and deserializer which is required for Read/Write operation on the data present in HDFS. The metastore comprises of two fundamental units**:**
* A service that provides metastore access to other Hive services.
* Disk storage for the metadata which is separate from HDFS storage.

**Metastore:**

Hive stores the schema of the Hive tables in a Hive Metastore. Metastore is used to hold all the information about the tables and partitions that are in the warehouse. By default, the metastore is run in the same process as the Hive service and the default Metastore is DerBy Database.

**Example show metastore:**

mysql –u root –p cloudera

show databases;

use metastore

show tables

select \* from tbls

select \* from columns\_v2;

**SerDe:**

Serializer, Deserializer gives instructions to hive on how to process a record.

### **Hive Table Types:**

Hive supports two types of tables.

###### **Managed Tables**– Default table type in Hive

* + Tables data is manged by Hive by moving data into its warehouse directory configured by hive.metastore.warehouse.dir (by default /user/hive/warehouse).
  + If this table is dropped both data and metadata (schema) are deleted. I.e. these tables are owned by Hive.
  + Less convenient to share with other tools like Pig, HBase etc, as these are maintained by Hive and data can be deleted without informing these tools.

###### **External Tables**

* + These tables are not managed or owned by Hive. And tables data will not be copied into hive warehouse directory but maintained at external location
  + If these tables are dropped only the schema from metastore will be deleted but not the data files from external location.
  + Provides convenience to share the tables data with other tools like Pig, HBase, etc…
  + *“Location”* Clause is mandatory to create an external table otherwise table will be managed by Hive only even if we create it with “External” keyword.

###### **Temporary Tables**

* + By the name itself, these are temporary and available till end of current session only.
  + Useful in case of creating intermediate tables to copy data records from one table to another but can be deleted after our copy operation.
  + Table’s Data will be stored in the user’s scratch directory configured by hive.exec.scratchdir, and deleted at the end of the session.
  + Temporary tables doesn’t support Partitioning & Indexing.

**Hive Data Types:**

#### Numeric Types

* TINYINT (1-byte signed integer, from -128 to 127)
* SMALLINT (2-byte signed integer, from -32,768 to 32,767)
* INT (4-byte signed integer, from -2,147,483,648 to 2,147,483,647)
* BIGINT (8-byte signed integer, from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
* FLOAT (4-byte single precision floating point number)
* DOUBLE (8-byte double precision floating point number)
* DECIMAL (Hive 0.13.0 introduced user definable precision and scale)

#### 2. Date/Time Types

* TIMESTAMP
* DATE

#### 3. String Types

* STRING
* VARCHAR
* CHAR

#### 4. Misc Types

* BOOLEAN
* BINARY

Apart from these primitive data types Hive offers some complex data types which are listed below:   
  
*5. Complex Types*

* arrays: ARRAY<data\_type>
* maps: MAP<primitive\_type, data\_type>
* structs: STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>
* union: UNIONTYPE<data\_type, data\_type, ...>

**Hive Commands:**

**Create table:**

CREATE DATABASE IF NOT EXISTS test\_db

COMMENT "Test Database created for tutorial"

Location

WITH DBPROPERTIES(

'Date' = '2014-12-03',

'Creator' = 'Bala G',

'Email' = 'bala@somewhere.com'

);

**Example:**

CREATE TABLE IF NOT EXISTS user (

first\_name VARCHAR(64),

last\_name VARCHAR(64),

company\_name VARCHAR(64),

address STRUCT<zip:INT, street:STRING>,

country VARCHAR(64),

city VARCHAR(32),

state VARCHAR(32),

post INT,

phone\_nos ARRAY<STRING>,

mail MAP<STRING, STRING>,

web\_address VARCHAR(64)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

COLLECTION ITEMS TERMINATED BY '\t'

MAP KEYS TERMINATED BY ':'

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/user/User\_Records.txt' OVERWRITE INTO TABLE user;

**External Table:**

CREATE EXTERNAL TABLE User\_ORC(

first\_name VARCHAR(64),

last\_name VARCHAR(64),

company\_name VARCHAR(64),

address STRUCT<zip:INT, street:STRING>,

country VARCHAR(64),

city VARCHAR(32),

state VARCHAR(32),

post INT,

phone\_nos ARRAY<STRING>,

mail MAP<STRING, STRING>,

web\_address VARCHAR(64)

)

COMMENT 'Temporary ORC table for testing purpose'

STORED AS ORC

LOCATION '/user/hive/orc/user'

   TBLPROPERTIES ("orc.compress"="SNAPPY");

INSERT OVERWRITE TABLE user\_ORC SELECT \* FROM user;

**TEMPORARY Table:**

CREATE TEMPORARY TABLE temp (col1 STRING, col2 INT);

**Drop Databases:**

When we no longer need a database, we can drop such databases with Drop command in hive.

By default, Hive does not allow us to drop databases that contain at least one or more tables. In this case, we need to either drop the tables first and then drop database or we need to provide

CASCADE argument to DROP command.

Ex: Drop database test\_db2

Observe the hive error message” *InvalidOperationException(message:Database test\_db2 is not empty. One or more tables exist.)”*when trying to delete test\_db2 which has onetest\_table with no CASCADE argument.

Ex: Drop database test\_db2 CASCADE;

When a database is dropped, its directory is also deleted

**Partition:**

Table partitioning means dividing table data into some parts based on the values of particular columns like date or country, segregate (separates) the input records into different files/directories based on date or country, this make it faster to do queries on slices(pieces) of data.

Partition helps in increasing the efficiency when performing a query on a table. Instead of scanning the whole table, it will only scan for the partitioned set and does not scan or operate on the un partitioned sets, which helps us to provide results in lesser(reduced) time and the details will be displayed very quickly because of Hive Partition.

Partitions are defined at the time of table creation using the PARTITIONED BY clause, with a list of column definitions for partitioning.

**Example Scenarios:**

* In a large user table where the table is partitioned by country, then selecting users of country ‘IN’ will just scan one directory ‘country=IN’ instead of all the directories
* Partitioning is used in real-time log files analysis to segregate the records based on time stamp or date value to see the results day wise quickly.
* Another real-time use is that, Customer/user details are partitioned by country/state or department for fast retrieval of subset data pertaining to some category.
* Sales records by-product type, country, year and month is another commonly used scenario.

Two types of partitions:

* Static partitioning
* Dynamic Partitioning

**Static partitioning:**

Knowing into which partition we should load data is called static partitioning.

Suppose we have receiving different sources us country sendiong separate file Canada country sending separate file, so already records are grouped by country, ex, us country file contains only us records.

In this mode, input data should contain the columns listed only in table definition (for example, firstname, lastname, address, city, post, phone1, phone2, email and web) but not the columns defined in partitioned by clause (country and state).

CREATE TABLE partitioned\_user(

firstname VARCHAR(64),

lastname  VARCHAR(64),

address   STRING,

city   VARCHAR(64),

post      STRING,

phone1    VARCHAR(64),

phone2    STRING,

email     STRING,

web       STRING

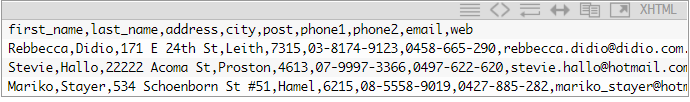
)

PARTITIONED BY (country VARCHAR(64), state VARCHAR(64))

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS SEQUENCEFILE;



LOAD DATA LOCAL INPATH '${env:HOME}/staticinput.txt'

      INTO TABLE partitioned\_user

      PARTITION (country = 'US', state = 'CA');

Output directory:

/user/hive/warehouse/partitioned\_user/country=US/state=CA/

Attribute mapping:

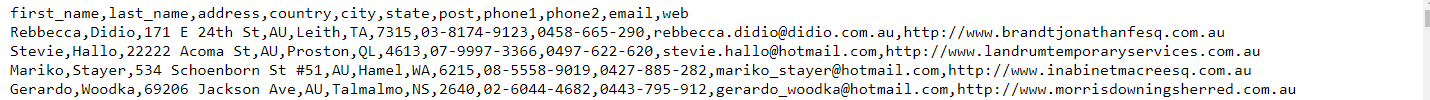
Set hive.cli.print,header=true

Select \* from partition\_user where country = 'US' and state = 'CA'

**Dynamic Partitioning:**

Examples:

**Input File:**



set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.exec.max.dynamic.partitions=1000;

set hive.exec.max.dynamic.partitions.pernode=1000;

DROP TABLE IF EXISTS partitioned\_user;

CREATE TEMPORARY TABLE temp\_user(

firstname VARCHAR(64),

lastname VARCHAR(64),

address STRING,

country VARCHAR(64),

city VARCHAR(64),

state VARCHAR(64),

post STRING,

phone1 VARCHAR(64),

phone2 STRING,

email STRING,

web STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/siva/UserRecords.txt' INTO TABLE temp\_user;

SELECT firstname, phone1, city

FROM temp\_user

WHERE country='US' AND state='CA'

ORDER BY city

LIMIT 5;

CREATE TABLE partitioned\_user(

firstname VARCHAR(64),

lastname VARCHAR(64),

address STRING,

city VARCHAR(64),

post STRING,

phone1 VARCHAR(64),

phone2 STRING,

email STRING,

web STRING

)

PARTITIONED BY (country VARCHAR(64), state VARCHAR(64))

STORED AS SEQUENCEFILE;

INSERT INTO TABLE partitioned\_user

PARTITION (country, state)

SELECT firstname ,

lastname ,

address ,

city ,

post ,

phone1 ,

phone2 ,

email ,

web ,

country ,

state

FROM temp\_user;

SELECT firstname, phone1, city

FROM partitioned\_user

WHERE country='US' AND state='CA'

ORDER BY city

LIMIT 5;

**Show partition:**

SHOW PARTITIONS partitioned\_user;

**PARTITION clause:**

SHOW PARTITIONS partitioned\_user PARTITION(country='US');

**Describe partitions:**

DESCRIBE FORMATTED partitioned\_user;

DESCRIBE FORMATTED partitioned\_user PARTITION(country='US', state='CA');

**Alter Partitions**

We can alter/change partitions (add/change/drop) with the help of below commands.

Adding Partitions

We can add partitions to an existing table with ADD PARTITION clause as shown below.

ALTER TABLE partitioned\_user ADD IF NOT EXISTS

PARTITION (country = 'US', state = 'XY') LOCATION '/hdfs/external/file/path1'

PARTITION (country = 'CA', state = 'YZ') LOCATION '/hdfs/external/file/path2'

PARTITION (country = 'UK', state = 'ZX') LOCATION '/hdfs/external/file/path2'

...;

**Changing Partitions:**

We can change a partition location with commands like below. This command does not move the data from the old location and does not delete the old data but the reference to old data file will be lost.

ALTER TABLE partitioned\_user PARTITION (country='US', state='CA')

SET LOCATION '/hdfs/partition/newpath';

**Drop Partitions:**

We can drop partitions of a table with DROP IF EXISTS PARTITION clause as shown below.

ALTER TABLE partitioned\_user DROP IF EXISTS PARTITION(country='US', state='CA');

**Note:** that we didn’t include country and state columns in table definition but included in partition definition. If we include them, then we will encounter error scenario 1. We can verify the partition columns of the table with the help of below command

But by default, Dynamic Partitioning is disabled in Hive.

**set hive.exec.dynamic.partition=true;**

**set hive.exec.dynamic.partition.mode=nonstrict;**

**set hive.exec.max.dynamic.partitions=1000;**

**set hive.exec.max.dynamic.partitions.pernode=1000;**

**Bucketing:**

Buckets in hive is used in segregating of hive table-data into multiple files or directories. it is used for efficient querying.

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

let’s assume a condition that there is a huge [dataset](https://acadgild.com/big-data/big-data-development-training-certification). At times, even after partitioning on a particular field or fields, the partitioned file size doesn’t match with the actual expectation and remains huge and we want to manage the partition results into different parts. To overcome this problem of partitioning, Hive provides Bucketing concept, which allows user to divide table data sets into more manageable parts

**Note:**

If created bucket table in hive, if you want use bucket table in impala/spark sql we cannot use them

Does not support bucket tables in impala and spark.

Example:

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.exec.max.dynamic.partitions.pernode=1000;

set hive.enforce.bucketing = true;

DROP TABLE IF EXISTS bucketed\_user;

CREATE TEMPORARY TABLE temp\_user(

firstname VARCHAR(64),

lastname VARCHAR(64),

address STRING,

country VARCHAR(64),

city VARCHAR(64),

state VARCHAR(64),

post STRING,

phone1 VARCHAR(64),

phone2 STRING,

email STRING,

web STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/user/user\_table.txt' INTO TABLE temp\_user;

CREATE TABLE bucketed\_user(

firstname VARCHAR(64),

lastname VARCHAR(64),

address STRING,

city VARCHAR(64),

state VARCHAR(64),

post STRING,

phone1 VARCHAR(64),

phone2 STRING,

email STRING,

web STRING

)

COMMENT 'A bucketed sorted user table'

PARTITIONED BY (country VARCHAR(64))

CLUSTERED BY (state) SORTED BY (city) INTO 32 BUCKETS

STORED AS SEQUENCEFILE;

set hive.enforce.bucketing = true;

INSERT OVERWRITE TABLE bucketed\_user PARTITION (country)

SELECT firstname ,

lastname ,

address ,

city ,

state ,

post ,

phone1 ,

phone2 ,

email ,

web , country FROM temp\_user;

**Hive Query:**

* Only equality joins, outer joins and left semi joins are supported in hive.
* Hive does’t support join conditions that are not equality condition as it is very difficult to express such conditions as map reduce jobs
* Also more than two tables can be joined in hive.

Select a.\* from a join b on (a.id=b.id)

Select a.val,b.val c.val from a join b on (a.key =b.key) join c on (c.key = b.key)

Note: > < >= <= not supoerted

**Partitioning:**

**Pros:**

* It distributes execution load horizontally.
* In partition, faster execution of queries with the low volume of data takes place. For example, search population from Vatican City returns very fast instead of searching entire world population.

**Cons:**

* There is the possibility of too many small partition creations- too many directories.
* Partition is effective for low volume data. But there some queries like group by on high volume of data take a long time to execute. For example, grouping population of China will take a long time as compared to a grouping of the population in Vatican City.
* There is no need for searching entire table column for a single record.

**Bucketing:**

**Pros:**

* It provides faster query response like portioning.
* In bucketing due to equal volumes of data in each partition, joins at Map side will be quicker.

**Cons:**

* We can define a number of buckets during table creation. But loading of an equal volume of data has to be done manually by programmers.

**Hive Interview Questions and Answers:**

**1.What is Hive?**

Hive is a data warehousing tool. It is an abstraction and it gives SQL queries to perform an analysis. It gives you logical abstraction over the databases and the tables but it is not a database.

**2.What is Hive a metastore?**

Hive contains two things: data and the metadata. The metadata contains the (column names, partitions information, bucketing information, SerDe etc.) i.e., the data about the actual table this is by default stored in the Derby database, we can also configure it to Oracle or MySQL database.

**3.What is the limitation of Derby database for Hive metastore?**

With derby database, you cannot have multiple connections or multiple sessions instantiated at the same time. Derby database runs in the local mode and it creates a log file so that multiple users cannot access Hive simultaneously.

**4.What are managed and external tables?**

We have got two things, one of which is data present in the HDFS and the other is the metadata, present in some database.

There are two categories of Hive tables i.e., Managed and External Tables.

In the Managed tables, both the data and the metadata are managed by Hive and if you drop the managed table, both data and metadata are deleted.

There are some situations where your data will be controlled by some other application and you want to read that data but you must allow Hive to delete that data. In such case, you can create an external table in Hive. In the external table, metadata is controlled by Hive but the actual data will be controlled by some other application. So, when you delete a table accidentally, only the metadata will be lost and the actual data will reside wherever it is.

**5.What are the complex data types in Hive?**

MAP

The Map contains a key-value pair where you can search for a value using the key.

STRUCT

A Struct is a collection of elements of different data types. For example, if you take the address, it can have different data types. For example, pin code will be in Integer format.

ARRAY

An Array will have a collection of homogeneous elements. For example, if you take your skillset, you can have N number of skills

UNIONTYPE

It represents a column which can have a value that can belong to any of the data types of your choice.

**6.How does partitioning help in the faster execution of queries?**

With the help of partitioning, a subdirectory will be created with the name of the partitioned column and when you perform a query using the WHERE clause, only the particular sub-directory will be scanned instead of scanning the whole table. This gives you faster execution of queries.

**7.How to enable dynamic partitioning in Hive?**

Related to partitioning there are two types of partitioning Static and Dynamic. In the static partitioning, you will specify the partition column while loading the data.

Whereas in dynamic partitioning, you push the data into Hive and then Hive decides which value should go into which partition. To enable dynamic partitioning, you have set the below property

set hive.exec.dynamic.parition.mode = nonstrict;

Example

insert overwrite table emp\_details\_partitioned

partition(location)

select \* from emp\_details;

**8.How does bucketing help in the faster execution of queries?**

If you have to join two large tables, you can go for reduce side join. But if both the tables have the same number of buckets or same multiples of buckets and also sorted on the same column there is a possibility of SMBMJ in which all the joins take place in the map phase itself by matching the corresponding buckets. Buckets are basically files that are created inside the HDFS directory.

There are different properties which you need to set for bucket map joins and they are as follows:

set hive.enforce.sortmergebucketmapjoin = false;

set hive.auto.convert.sortmerge.join =  false;

set hive.optimize.bucketmapjoin =  ture;

set hive.optimize.bucketmapjoin.sortedmerge = true;

**9.How to enable bucketing in Hive?**

By default bucketing is disabled in Hive, you can enforce to enable it by setting the below property

set hive.enforce.bucketing  = true;

**10.Which method has to be overridden when we use custom UDF in Hive?**

Whenever you write a custom UDF in Hive, you have to extend the UDF class and you have to override the evaluate() function.

**11.What are the different file formats in Hive?**

There are different file formats supported by Hive

Text File format

Sequence File format

RC file format

Parquet

Avro

ORC

Every file format has its own characteristics and Hive allows you to choose easily the file format which you wanted to use.

**12.How is SerDe different from File format in Hive?**

SerDe stands for Serializer and Deserializer. It determines how to encode and decode the field values or the column values from a record that is: how you serialize and deserialize the values of a column

But file format determines how records are stored in key value format or how do you retrieve the records from the table.

**13.What is RegexSerDe?**

Regex stands for a regular expression. Whenever you want to have a kind of pattern matching, based on the pattern matching, you have to store the fields. RegexSerDe is present in org.apache.hadoop.hive.contrib.serde2.RegexSerDe.

In the SerDeproperties, you have to define your input pattern and output fields. For example, you have to get the column values from line [xyz/pq@def](about:blank) if you want to take xyz, pq and def separately.

To extract the pattern, you can use:

‘input.regex’ = ‘(.\*)/(.\*)@(.\*)’

To specify how to store them, you can use

‘output.format.string’ = ‘%1$s%2$s%3$s’;

**14.How is ORC file format optimised for data storage and analysis?**

ORC stores collections of rows in one file and within the collection the row data will be stored in a columnar format. With columnar format, it is very easy to compress, thus reducing a lot of storage cost.

While querying also, it queries the particular column instead of querying the whole row as the records are stored in columnar format.

ORC has got indexing on every block based on the statistics min, max, sum, count on columns so when you query, it will skip the blocks based on the indexing.

**15.How to access HBase tables from Hive?**

Using Hive-HBase storage handler, you can access the HBase tables from Hive and once you are connected, you can query HBase using the SQL queries from Hive. You can also join multiple tables in HBase from Hive and retrieve the result.

**Petrol use case:**

create table petrol (distributer\_id STRING,distributer\_name STRING,amt\_IN STRING,amy\_OUT STRING,vol\_IN INT,vol\_OUT INT,year INT) row format delimited fields terminated by ',' lines terminated by '\n';

load data local inpath '/home/cloudera/Desktop/petrol.txt' into table petrol;

**In real life what is the total amount of petrol in volume sold by every distributor?**

select distributer\_name,sum(vol\_OUT) from petrol group by distributer\_name;

select distributer\_name,sum(vol\_OUT) from petrol where distributer\_name='reliance';

**Which are the top 10 distributors ID’s for selling petrol and also display the amount of petrol sold in volume by them individually?**

select distributer\_name,vol\_OUT from petrol order by vol\_OUT desc limit 10;

**Find real life 10 distributor name who sold petrol in the least amount.**

select distributer\_name,vol\_OUT from petrol order by vol\_OUT limit 10;

**List all distributors who have this difference, along with the year and the difference which they have in that year.**

select distributer\_name, year from petrol where (vol\_IN-vol\_OUT) > 400;

### Olympic:

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;

load data local inpath ‘/home/acadgild/Downloads/olympic\_data.csv’ into table olympic;

**Using the dataset list the total number of medals won by each country in swimming.**

select country,SUM(total) from olympic where sport = “Swimming” GROUP BY country;

2)Display real life number of medals India won year wise.

**select year,SUM(total) from olympic where country = “India” GROUP BY year;**

3)Find the total number of medals each country won display the name along with total medals.

**select country,SUM(total) from olympic GROUP BY country;**

4)Find the real life number of gold medals each country won.

**select country,SUM(gold) from olympic GROUP BY country;**

**Which country got medals for Shooting, year wise classification?**

**Census use Case:**

Census is an official count or survey, especially of a population. In India, the census data has been surveyed 15 times as of 2011. Census is conducted every 10 years. It has been conducted by the Registrar General and Census Commissioner of India, under the Ministry of Home Affairs, Government of India.

1. **State-Wise Population**

Here we will find the population of each state and we have arranged them in a descending order. The state of Uttar Pradesh (UP) stays in the first place and the same can be seen in the following bar and pie charts:

select state,sum(persons) as total\_population from census group by state order by total\_population desc;

1. **Growth Rate of Each State Between 1991-2001**

Census are calculated for every 10 years. In between, 1991-2001, we can see the growth rate of each state in the following bar and pie charts. Nagaland stands at the top with the highest growth rate.

select state,avg(Growth\_1991\_2001) as total\_growth from census group by state

1. **Literacy Rate of Each State**

select state,avg(Persons\_literacy\_rate) from census group by state

1. **States with More Female Population**

In every state, women are given some special privileges. Here, from the following query, you can find states which have a greater female population

select state, sum(Males)-sum(Females) from census group by state

1. **Percentage of Population in Every State**

Here, we can find the percentage of a population each state has using the following query. UP itself has 16% of India’s total population.

select state, (sum(persons) \* 100.0) / SUM(sum(persons)) over() as percent\_pop\_by\_state

FROM census group by state

1. **Percentage of People Working in Each State**

Here, we will find the percentage of people working in each state and visualize the same using bar and pie charts. Percentage of working people can be calculated using the following query.

select state,sum(Total\_workers)\*100/sum(persons) from census group by state

<http://www.geoinsyssoft.com/hive-partition-bucketing/>

<https://acadgild.com/blog/partitioning-in-hive/>

<https://acadgild.com/blog/category/big-data-and-hadoop-advanced/>

<http://www.hadooptechs.com/category/hive/page/4>

<http://www.hadooptechs.com/category/hive/page/4>

<http://hadooped.blogspot.in/2013/06/apache-sqoop-part-2-for-data.html>

<https://acadgild.com/blog/map-side-joins-in-hive/>