

Introduction to Hive

Hive evolved as a data warehousing solution built on top of Hadoop Map-Reduce framework.

Hive engine compiles these queries into Map-Reduce jobs to be executed on Hadoop. In addition, custom Map-Reduce scripts can also be plugged into queries. Hive operates on data stored in tables which consists of primitive data types and collection data types like arrays and maps.

Hive comes with a command-line shell interface which can be used to create tables and execute queries.

Hive query language is similar to SQL wherein it supports subqueries. With Hive query language, it is possible to take a MapReduce joins across Hive tables. It has a support for simple SQL like functions- CONCAT, SUBSTR, ROUND etc., and aggregation functions- SUM, COUNT, MAX etc. It also supports GROUP BY and SORT BY clauses. It is also possible to write user defined functions in Hive query language.

Hive Vs Map Reduce

While choosing between Hive and Map reduce following factors are taken in consideration;

- Type of Data
- Amount of Data
- Complexity of Code

Hive Vs Map Reduce?

Feature	Hive	Map Reduce
Language	It Supports SQL like query language for interaction and for Data modeling	<ul style="list-style-type: none">• It compiles language with two main tasks present in it. One is map task, and another one is a reducer.• We can define these task using Java or Python
Level of abstraction	Higher level of Abstraction on top of HDFS	Lower level of abstraction

Efficiency in Code	Comparatively lesser than Map reduce	Provides High efficiency
--------------------	--------------------------------------	--------------------------

Extent of code	Less number of lines required for execution	code	More number of lines of codes to be defined
----------------	---	------	---

Type of Development work required	Less Development work required
-----------------------------------	--------------------------------

Architecture & Modes

What is Hive?

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). Hive makes job easy for performing operations like

- Data encapsulation
- Ad-hoc queries
- Analysis of huge datasets

Important characteristics of Hive

- 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.
- While dealing with structured data, Map Reduce doesn't have optimization and usability features like UDFs but Hive framework does. Query optimization refers to an effective way of query execution in terms of performance.
- Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, etc. for ease of learning.
- 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](#) 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.

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.

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

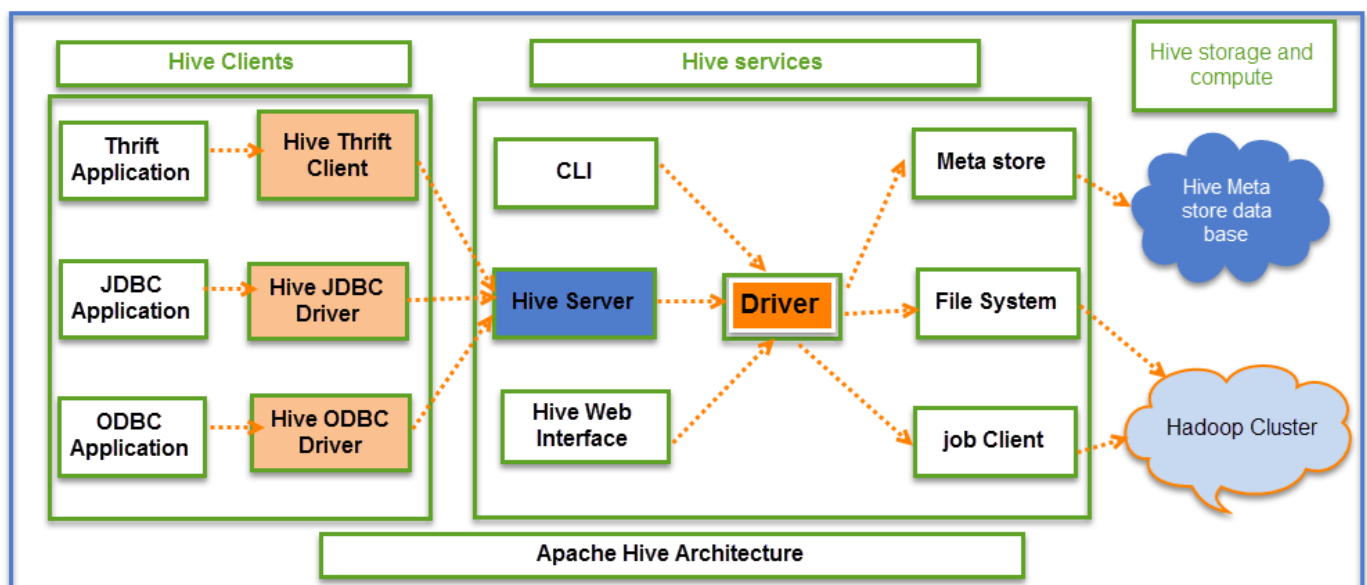
Relational databases are of "Schema on READ and Schema on Write". First creating a table then inserting data into the particular table. On relational database tables, functions like Insertions, Updates, and Modifications can be performed.

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

Hive Architecture



The above screenshot explains the Apache Hive architecture in detail

Hive Consists of Mainly 3 core parts

1. Hive Clients

2. Hive Services
3. Hive Storage and Computing

Hive Clients:

Hive provides different drivers for communication with a different type of applications. For Thrift based applications, it will provide Thrift client for communication.

Hive Services:

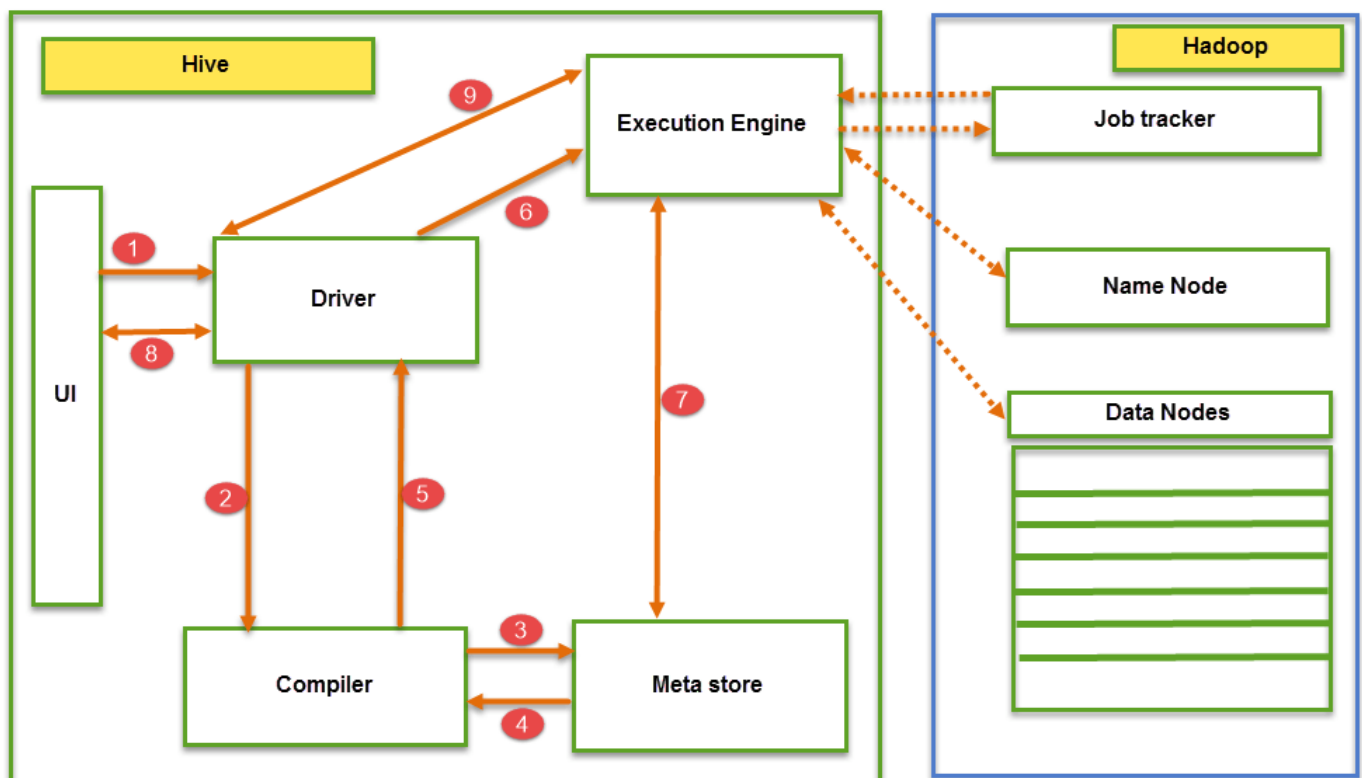
Client interactions with Hive can be performed through Hive Services. If the client wants to perform any query related operations in Hive, it has to communicate through Hive Services.

Hive Storage and Computing:

Hive services such as Meta store, File system, and Job Client in turn communicates with Hive storage and performs the following actions

- Metadata information of tables created in Hive is stored in Hive "Meta storage database".
- Query results and data loaded in the tables are going to be stored in Hadoop cluster on HDFS.

Job execution flow:



From the above screenshot we can understand the Job execution flow in Hive with Hadoop

The data flow in Hive behaves in the following pattern;

1. Executing Query from the UI(User Interface)
2. The driver is interacting with Compiler for getting the plan. (Here plan refers to query execution) process and its related metadata information gathering
3. The compiler creates the plan for a job to be executed. Compiler communicating with Meta store for getting metadata request
4. Meta store sends metadata information back to compiler
5. Compiler communicating with Driver with the proposed plan to execute the query
6. Driver Sending execution plans to Execution engine
7. Execution Engine (EE) acts as a bridge between Hive and Hadoop to process the query. For DFS operations.
 - EE should first contacts Name Node and then to Data nodes to get the values stored in tables.
 - EE is going to fetch desired records from Data Nodes. The actual data of tables resides in data node only. While from Name Node it only fetches the metadata information for the query.
 - It collects actual data from data nodes related to mentioned query
 - Execution Engine (EE) communicates bi-directionally with Meta store present in Hive to perform DDL (Data Definition Language) operations. Here DDL operations like CREATE, DROP and ALTERING tables and databases are done. Meta store will store information about database name, table names and column names only. It will fetch data related to query mentioned.
 - Execution Engine (EE) in turn communicates with Hadoop daemons such as Name node, Data nodes, and job tracker to execute the query on top of Hadoop file system
8. Fetching results from driver
9. Sending results to Execution engine. Once the results fetched from data nodes to the EE, it will send results back to driver and to UI (front end)

Hive Continuously in contact with Hadoop file system and its daemons via Execution engine. The dotted arrow in the Job flow diagram shows the Execution engine communication with Hadoop daemons.

Different modes of Hive

Hive can operate in two modes depending on the size of data nodes in Hadoop.

These modes are,

- Local mode
- Map reduce mode

When to use Local mode:

- If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
- If the data size is smaller in term of limited to single local machine, we can use this mode
- Processing will be very fast on smaller data sets present in the local machine

When to use Map reduce mode:

- If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
- It will perform on large amount of data sets and query going to execute in parallel way
- Processing of large data sets with better performance can be achieved through this mode

Hive to work in local mode set

SET mapred.job.tracker=local;

What is Hive Server2 (HS2)?

HiveServer2 (HS2) is a server interface that performs following functions:

- Enables remote clients to execute queries against Hive
- Retrieve the results of mentioned queries

From the latest version it's having some advanced features Based on Thrift RPC like;

- Multi-client concurrency
- Authentication

Hive Data Types & Create, Drop Database

Data types in Hive

These are

- Numeric Types
- String Types
- Date/Time Types
- Complex Types

Numeric Types:

Type	Memory allocation
TINY INT	Its 1-byte signed integer (-128 to 127)
SMALL INT	2-byte signed integer (-32768 to 32767)
INT	4 –byte signed integer (-2,147,484,648 to 2,147,484,647)
BIG INT	8 byte signed integer
FLOAT	4 – byte single precision floating point number
DOUBLE	8- byte double precision floating point number
DECIMAL	We can define precision and scale in this Type

String Types:

Type	Length
CHAR	255
VARCHAR	1 to 65355
STRING	We can define length here(No Limit)

Date/Time Types:

Type	Usage
Timestamp	Supports traditional Unix timestamp with optional nanosecond precision
Date	<ul style="list-style-type: none">• It's in YYYY-MM-DD format.• The range of values supported for the Date type is be 0000-01-01 to 9999-12-31, dependent on support by the primitive Java Date type

Complex Types:

Type	Usage
Arrays	ARRAY<data_type> Negative values and non-constant expressions not allowed

Maps	MAP<primitive_type, data_type>	Negative values and non-constant expressions not allowed
Structs	STRUCT<col_name :datatype, >	
Union	UNIONTYPE<data_type, datatype,>	

Creation and dropping of Database in Hive:

Create Database:

Create database <DatabaseName>

Example: -Create database "guru"

Hive Create, Alter & Drop Table

Table Operations such as Creation, Altering, and Dropping tables in Hive can be observed in this tutorial.

In the Below screenshot, we are creating a table with columns and altering the table name.

1. Creating table guru_sample with two column names such as "empid" and "empname"
2. Displaying tables present in guru99 database
3. Guru_sample displaying under tables
4. Altering table "guru_sample" as "guru_sampleNew"
5. Again when you execute "show" command, it will display the new name Guru_sampleNew

```
hive> create table guru_sample(empid int, empname string) ; 1
OK
Time taken: 1.635 seconds
hive> show tables; 2
OK
allstates
guru_sample 3
Time taken: 0.105 seconds, Fetched: 2 row(s)
hive> ALTER table guru_sample RENAME to guru_sampleNew; 4
OK
Time taken: 0.544 seconds
hive> show tables;
OK
allstates
guru_sampleNew 5
```

Dropping table guru_sampleNew:

```
hive> drop table guru_sampleNew;
OK
Time taken: 2.732 seconds
```

Dropping table

Table types and its Usage:

Hive deals with two types of table structures like Internal and External tables depending on the loading and design of schema in Hive.

Internal tables

- Internal Table is tightly coupled in nature. In this type of table, first we have to create table and load the data.
- We can call this one as data on schema.
- By dropping this table, both data and schema will be removed.
- The stored location of this table will be at /user/hive/warehouse.

When to Choose Internal Table:

27.4M

237

What is Software Testing Why Testing is Important

- If the processing data available in local file system
- If we want Hive to manage the complete lifecycle of data including the deletion

Sample code Snippet for Internal Table

1. To create the internal table

```
Hive>CREATE TABLE guruhive_internalsample (id INT,Name STRING);
Row format delimited
Fields terminated by '\t';
```

2. Load the data into internal table

Hive>LOAD DATA INPATH '/user/guru99hive/data.txt' INTO table guruhive_internaltable;

3. Display the content of the table

Hive>select * from guruhive_internaltable;

4. To drop the internal table

Hive>DROP TABLE guruhive_internaltable;

```
hive> CREATE TABLE guruhive_internaltable (id INT,Name STRING)
> Row format delimited
> fields terminated by ',';
OK
Time taken: 0.131 seconds
hive> load data local inpath '/home/hduser/data.txt' into table guruhive_internaltable;
Loading data to table default.guruhive_internaltable
Table default.guruhive_internaltable stats: [numFiles=1, totalSize=131]
OK
Time taken: 0.289 seconds
hive> select * from guruhive_internaltable;
OK
101      Ram
102      Santosh
103      Ramesh
104      Rajesh
105      Sreekanth
106      Veerendra
107      Samuel Simon
108      Rahim
109      Sravanthi
110      Lakshmi
Time taken: 0.133 seconds, Fetched: 10 row(s)
hive> drop table guruhive_internaltable;
OK
Time taken: 0.242 seconds
```

Creation of table
"guruhive_internaltable"

Loading Data into
"guruhive_internaltable"

Displaying Contents of table
"guruhive_internaltable"

dropping table
"guruhive_internaltable"

- Create the internal table
- Load the data into internal table
- Display the content of the table
- To drop the internal table

External tables

- External Table is loosely coupled in nature. Data will be available in HDFS. The table is going to create on HDFS data.
- In other way, we can say like its creating schema on data.
- At the time of dropping the table it drops only schema, the data will be still available in HDFS as before.
- External tables provide an option to create multiple schemas for the data stored in HDFS instead of deleting the data every time whenever schema updates

When to Choose External Table:

- If processing data available in HDFS
- Useful when the files are being used outside of Hive

1. Create External table

Hive>CREATE EXTERNAL TABLE guruhive_external(id INT,Name STRING)

Row format delimited

Fields terminated by '\t'

LOCATION '/user/guruhive/guruhive_external';

2. If we are not specifying the location at the time of table creation, we can load the data manually

Hive>LOAD DATA INPATH '/user/guruhive/data.txt' INTO TABLE guruhive_external;

3. Display the content of the table

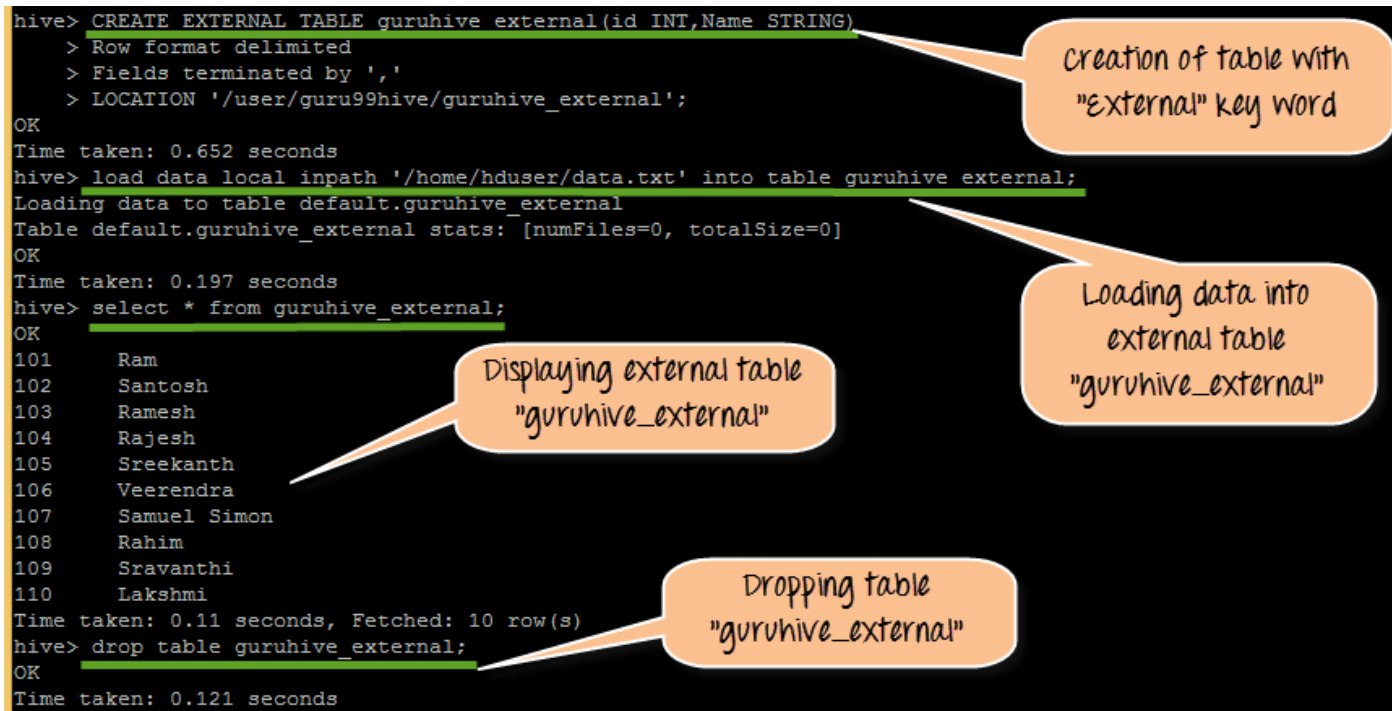
Hive>select * from guruhive_external;

4. To drop the internal table

Hive>DROP TABLE guruhive_external;

From the following screen shot, we can observe the output

```
hive> CREATE EXTERNAL TABLE guruhive_external(id INT,Name STRING)
> Row format delimited
> Fields terminated by ','
> LOCATION '/user/guru99hive/guruhive_external';
OK
Time taken: 0.652 seconds
hive> load data local inpath '/home/hduser/data.txt' into table guruhive_external;
Loading data to table default.guruhive_external
Table default.guruhive_external stats: [numFiles=0, totalSize=0]
OK
Time taken: 0.197 seconds
hive> select * from guruhive_external;
OK
101      Ram
102      Santosh
103      Ramesh
104      Rajesh
105      Sreekanth
106      Veerendra
107      Samuel Simon
108      Rahim
109      Sravanthi
110      Lakshmi
Time taken: 0.11 seconds, Fetched: 10 row(s)
hive> drop table guruhive_external;
OK
Time taken: 0.121 seconds
```



In above code, we do following things

- Create the External table
- Load the data into External table
- Display the content of the table
- Dropping external table

Difference between Internal Vs External tables

Feature	Internal	External
Schema	Data on Schema	Schema on Data
Storage Location	/usr/hive/warehouse	HDFS location
Data availability	Within local file system	Within HDFS

Hive Partitions & Buckets with Example

Tables, Partitions, and Buckets are the parts of Hive data modeling.

Hive Partitions is a way to organizes tables into partitions by dividing tables into different parts based on partition keys.

Partition is helpful when the table has one or more Partition keys. Partition keys are basic elements for determining how the data is stored in the table.

Sample Code Snippet for partitions

1. Creation of Table all states

```
create table all states(state string, District string,Enrolments string)
row format delimited
fields terminated by ',';
```

2. Loading data into created table all states

Load data local inpath '/home/hduser/Desktop/AllStates.csv' into table allstates;

3. Creation of partition table

```
create table state_part(District string,Enrolments string) PARTITIONED BY(state string);
```

4. For partition we have to set this property

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

6. Loading data into partition table

```
INSERT OVERWRITE TABLE state_part PARTITION(state)
```

```
SELECT district,enrolments,state from allstates;
```

6. Actual processing and formation of partition tables based on state as partition key

7. There are going to be 38 partition outputs in HDFS storage with the file name as state name. We will check this in this step

The following screen shots will show u the execution of above mentioned code

The image contains three screenshots of a Hive CLI session, each with a red circle number and an orange callout box:

- 1** *creation of table "allstates"*
The first screenshot shows the command: `hive> create table allstates(state string, District string,Enrolments string)` followed by `> row format delimited` and `> fields terminated by ',';`. The output is `OK`.
- 2** *Loading Data into "allstates"*
The second screenshot shows the command: `hive> load data local inpath '/home/hduser/AllStates.csv' into table allstates;`. The output includes: `Loading data to table default.allstates`, `Table default.allstates stats: [numFiles=1, totalSize=36913]`, `OK`, and `Time taken: 1.459 seconds`.
- 3** *creation of partition table "state_part"*
The third screenshot shows the command: `hive> create table state part(District string,Enrolments string) PARTITIONED BY(state string);`. The output is `OK` and `Time taken: 0.199 seconds`.

```

hive> set hive.exec.dynamic.partition.mode=nonstrict
> ;
hive> insert overwrite table state part PARTITION(state) SELECT district,enrolments,state from allstates;
Query ID = hduser_20151104161604_ce10a013-7e6e-4545-94b9-b26dcaad8879
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201511041430_0001, Tracking URL = http://localhost:50030/jobdetails
Kill Command = /usr/local/hadoop-1.2.1/libexec/./bin/hadoop job -kill job_2015110414
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2015-11-04 16:16:13,677 Stage-1 map = 0%, reduce = 0%
2015-11-04 16:16:18,699 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.48 sec
2015-11-04 16:16:19,702 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.48 sec
MapReduce Total cumulative CPU time: 1 seconds 480 msec
Ended Job = job_201511041430_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:54310/user/hive/warehouse/state_part/.hive-staging_hive_2015-11-04_16-16-04_2
Loading data to table default.state part partition (state=null)
Time taken for load dynamic partitions : 5282
Loading partition {state=Haryana}
Loading partition {state=Uttarakhand}
Loading partition {state=Daman_and_Diu}
Loading partition {state=Puducherry}
Loading partition {state=Uttar_Pradesh}
Loading partition {state=Assam}
Loading partition {state=Others}
Loading partition {state=Arunachal_Pradesh}
Loading partition {state=Lakshadweep}
Loading partition {state=West_Bengal}
Loading partition {state=Sikkim}
Loading partition {state=Himachal_Pradesh}
Loading partition {state=Jharkhand}
Loading partition {state=Tripura}
Loading partition {state=Punjab}
Loading partition {state=Tamil_Nadu}
Loading partition {state=Gujarat}

```

4

5

Making partition based on "state" field

6

Creation Partition tables using "state" as partition key during Map reduce process

```

hduser@datamatics-Ubuntu:~/usr/local/hadoop-1.2.1/bin$ ./hadoop dfs -ls /user/hive/warehouse/state_part
Warning: SHADOOP_HOME is deprecated.
Found 38 items
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Andaman_and_Nicobar_Island
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Andhra_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Arunachal_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Assam
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Bihar
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Chandigarh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Chhattisgarh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Dadra_and_Nagar_Haveli
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Daman_and_Diu
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Delhi
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Goa
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Gujarat
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Haryana
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Himachal_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Jammu_and_Kashmir
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Jharkhand
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Karnataka
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Kerala
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Lakshadweep
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Madhya_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Maharashtra
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Manipur
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Meghalaya
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Mizoram
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Nagaland
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Odisha
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Others
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Puducherry
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Punjab
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Rajasthan
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Sikkim
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Tamil_Nadu
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Tripura
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Uttar_Pradesh
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=Uttarakhand
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=West_Bengal
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=california
drwxr-xr-x - hduser supergroup 0 2015-11-04 16:16 /user/hive/warehouse/state_part/state=newyork

```

Total 38 states present in table

38 partition tables stored in HDFS system

7

From the above code, we do following things

1. Creation of table all states with 3 column names such as state, district, and enrollment
2. Loading data into table all states

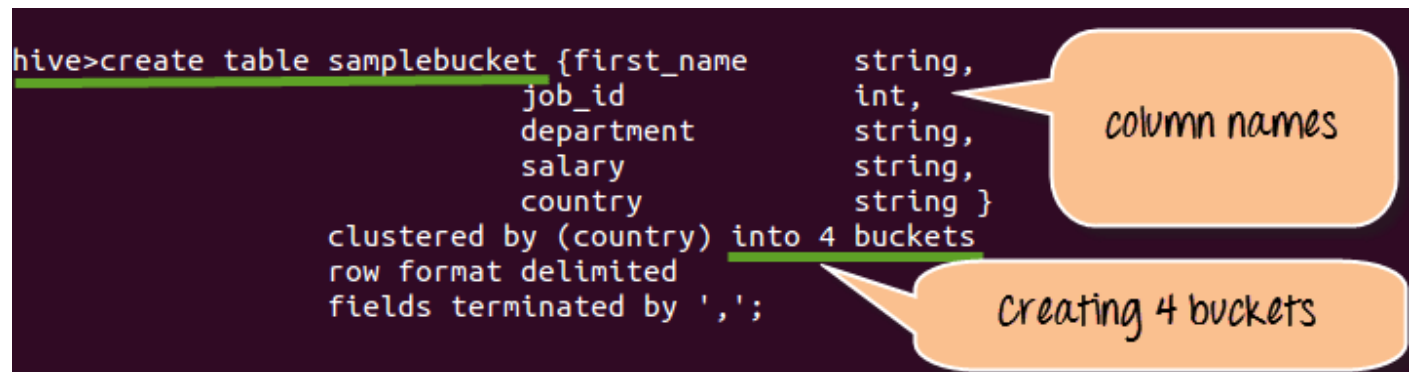
3. Creation of partition table with state as partition key
4. In this step Setting partition mode as non-strict(This mode will activate dynamic partition mode)
5. Loading data into partition tablestate_part
6. Actual processing and formation of partition tables based on state as partition key
7. There is going to 38 partition outputs in HDFS storage with the file name as state name. We will check this in this step. In This step, we seeing the 38 partition outputs in HDFS

What is Buckets?

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

- The data i.e. present in that partitions can be divided further into Buckets
- The division is performed based on Hash of particular columns that we selected in the table.
- Buckets use some form of Hashing algorithm at back end to read each record and place it into buckets
- In Hive, we have to enable buckets by using the set.hive.enforce.bucketing=true;

Step 1) Creating Bucket as shown below.



```
hive>create table samplebucket {first_name    string,
                                job_id       int,
                                department    string,
                                salary        string,
                                country       string }
                                clustered by (country) into 4 buckets
                                row format delimited
                                fields terminated by ',';
```

The screenshot shows a Hive CLI session. The command to create a table named 'samplebucket' is entered. The table has five columns: first_name (string), job_id (int), department (string), salary (string), and country (string). It is clustered by the 'country' column into 4 buckets. The row format is delimited and fields are terminated by commas. Two orange callout boxes are present: one pointing to the column definitions with the text 'column names', and another pointing to the 'into 4 buckets' part of the command with the text 'creating 4 buckets'.

From the above screen shot

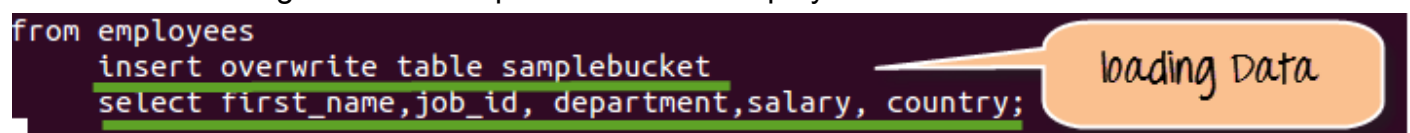
- We are creating sample_bucket with column names such as first_name, job_id, department, salary and country
- We are creating 4 buckets overhere.
- Once the data get loaded it automatically, place the data into 4 buckets

Step 2) Loading Data into table sample bucket

Assuming that"Employees table" already created in Hive system. In this step, we will see the loading of Data from employees table into table sample bucket.

Before we start moving employees data into buckets, make sure that it consist of column names such as first_name, job_id, department, salary and country.

Here we are loading data into sample bucket from employees table.

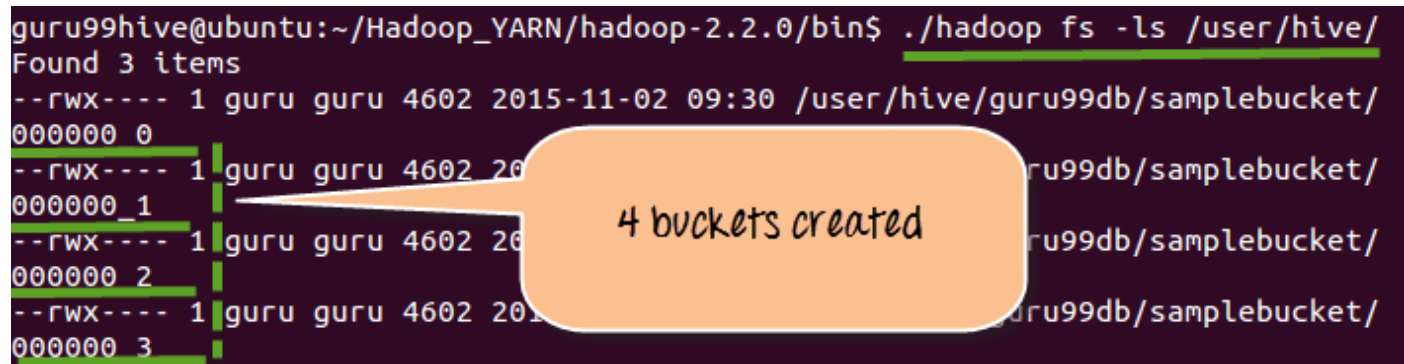


```
from employees
insert overwrite table samplebucket
select first_name,job_id, department,salary, country;
```

The screenshot shows a Hive CLI session. The command to load data from the 'employees' table into the 'samplebucket' table is entered. The command is 'insert overwrite table samplebucket select first_name,job_id, department,salary, country;'. An orange callout box points to the 'insert overwrite' part of the command with the text 'loading Data'.

Step 3)Displaying 4 buckets that created in Step 1

```
guru99hive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -ls /user/hive/
Found 3 items
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 0
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000_1
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 2
--rwx---- 1 guru guru 4602 2015-11-02 09:30 /user/hive/guru99db/samplebucket/
000000 3
```



From the above screenshot, we can see that the data from the employees table is transferred into 4 buckets created in step 1.