

EXPERIMENT NO: 04**Title:** Introduction of Hadoop HIVE**Aim:** Understanding of HIVE working environment.**Theory:**

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.

Let see now what makes Hive so fast.

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

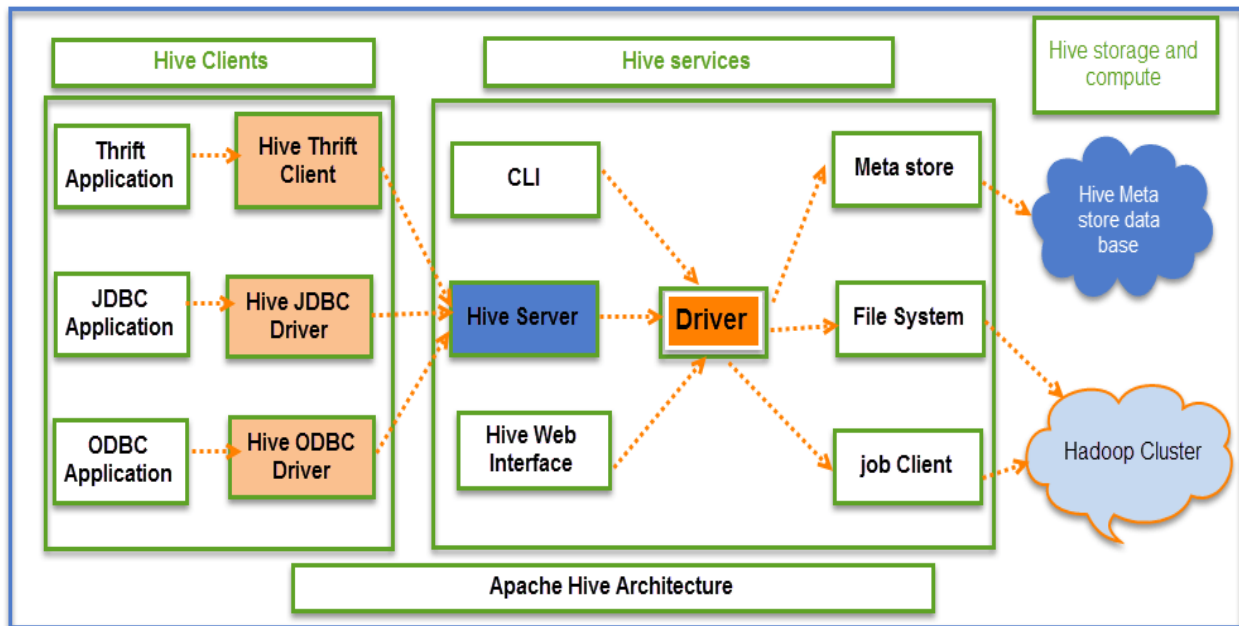
Relational databases are of "**Schema on READ and Schema on Write**". First creating a table then inserting data into the particular table. On relational database tables, functions like Insertions, Updates, and Modifications can be performed.

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

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.

For Java related applications, it provides JDBC Drivers. Other than any type of applications provided ODBC drivers. These Clients and drivers in turn again communicate with Hive server in the Hive services.

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.

CLI is the command line interface acts as Hive service for DDL (Data definition Language) operations. All drivers communicate with Hive server and to the main driver in Hive services as shown in above architecture diagram.

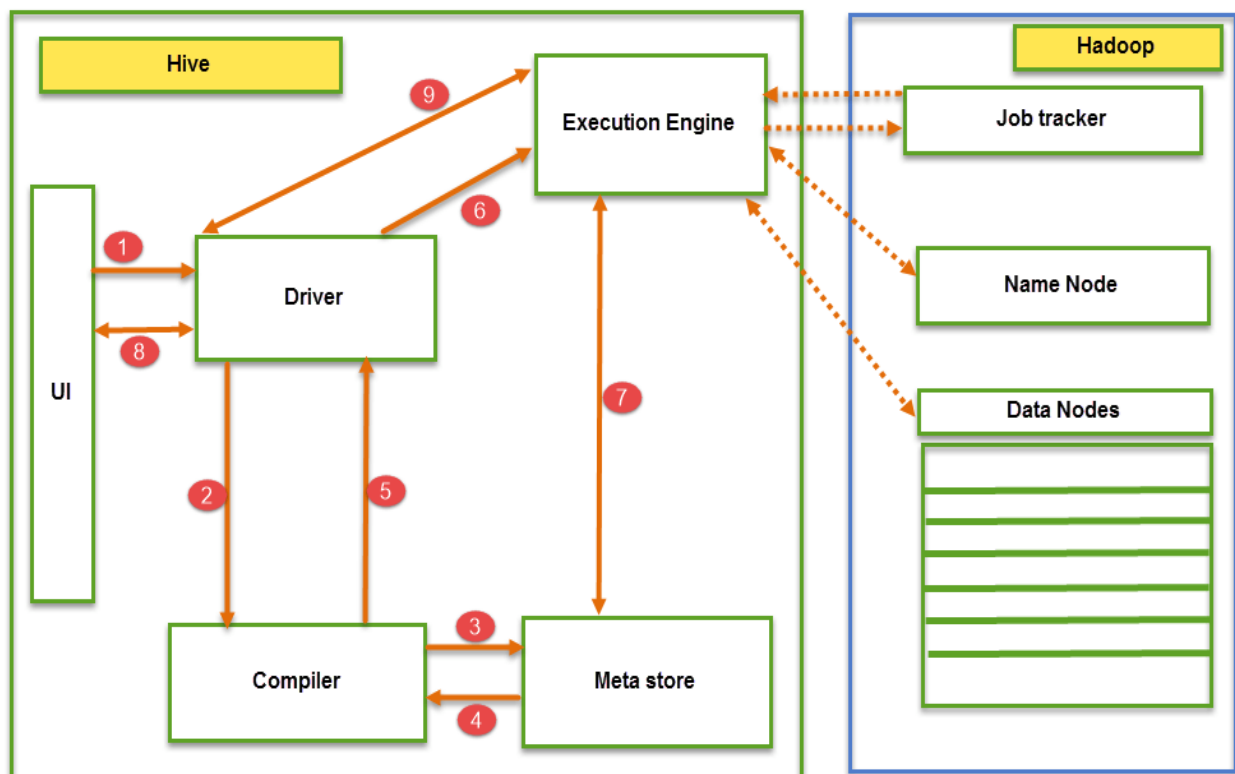
Driver present in the Hive services represents the main driver, and it communicates all type of JDBC, ODBC, and other client specific applications. Driver will process those requests from different applications to meta store and field systems for further processing.

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

In Hive, we can set this property to mention which mode Hive can work? By default, it works on Map Reduce mode and for local mode you can have the following setting.

Hive to work in local mode set

SET mapred.job.tracker=local;

From the Hive version 0.7 it supports a mode to run map reduce jobs in local mode automatically.

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

How to Install Hive**Step 1) Downloading and Installing Hive**


For downloading Hive stable setup refer Apache URL as mentioned below

<http://www.apache.org/dyn/closer.cgi/hive/>. Go to the URL and select the apache mirror download link.

[News](#)
[About ▾](#)
[Make a Donation ▾](#)
[The Apache Way ▾](#)
[Join Us ▾](#)
[Downloads ▾](#)



CELEBRATING 20 YEARS OF COMMUNITY-LED DEVELOPMENT
"THE APACHE WAY"



[Projects ▾](#)
[People ▾](#)
[Community ▾](#)
[License ▾](#)
[Sponsors ▾](#)

We suggest the following mirror site for your download:

<http://mirrors.estointernet.in/apache/hive/>

Other mirror sites are suggested below:

It is essential that you verify the integrity of the downloaded file using the PGP signature (.asc file) or a hash (.md5 or .sha* file).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

HTTP

<http://apachemirror.wuchna.com/hive/>

<http://mirrors.estointernet.in/apache/hive/>

BACKUP SITES









Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

<https://www-eu.apache.org/dist/hive/>

<https://www-us.apache.org/dist/hive/>

Select the Latest version of Hive. (In my current case it is hive – 3.1.2)

Index of /hive

Name	Last modified	Size	Description
 Parent Directory		-	
 hive-1.2.2/	2018-05-04 20:51	-	
 hive-2.3.6/	2019-08-22 18:53	-	
 hive-3.1.2/	2019-08-26 20:21	-	
 hive-standalone-metastore-3.0.0/	2018-06-07 18:12	-	
 hive-storage-2.4.0/	2018-05-04 20:48	-	
 hive-storage-2.6.1/	2018-05-11 22:26	-	
 hive-storage-2.7.0/	2018-07-19 18:37	-	
 stable-2/	2019-08-22 18:53	-	

Apache/2.4.29 (Ubuntu) Server at apachemirror.wuchna.com Port 80

Click on the bin file and downloading will start.

Index of /hive/hive-3.1.2

Name	Last modified	Size	Description
 Parent Directory		-	
 apache-hive-3.1.2-bin.tar.gz	2019-08-26 20:20	266M	
 apache-hive-3.1.2-src.tar.gz	2019-08-26 20:20	24M	

Apache/2.4.29 (Ubuntu) Server at apachemirror.wuchna.com Port 80

Step 2) Extracting the tar file.

Go to the downloaded Tar file location ->extract the tar file by using the following command

```
tar -xvf apache-hive-3.1.2-bin.tar.gz
```

```
guru99hive@ubuntu:~/Desktop$ ls
apache-hive-1.2.0-bin.tar.gz
guru99hive@ubuntu:~/Desktop$ tar -xvf apache-hive-1.2.0-bin.tar.gz
```

Step 3) Different Configuration properties to be placed in Apache Hive.

In this step, we are going to do two things

1. Placing Hive Home path in bashrc file
2. Placing Hadoop Home path location in hive-config.sh

1. Mention **Hive Path**in **~/.bashrc**

```
guru99hive@ubuntu:~$ nano ~/.bashrc
```

- Open the bashrc file as shown in above screenshot
- Mention Hive home path i.e., HIVE_HOME path in bashrc file and export it as shown in below

```
export HIVE_HOME="/home/guru99hive/apache-hive-1.2.0-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

Code to be placed in bashrc


```
export HIVE_HOME="/home/guru99hive/apache-hive-1.2.0-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

2. Exporting **Hadoop path in Hive-config.sh** (To communicate with the Hadoop ecosystem we are defining Hadoop Home path in hive config field)

Open the hive-config.sh as shown in below

```
guru99hive@ubuntu:~/apache-hive-1.2.0-bin/bin$ nano hive-config.sh
```

Mention the HADOOP_HOME Path in hive-config.sh file as shown in below (HADOOP_HOME Path)

```
export HADOOP_HOME=/home/guru99hive/Hadoop_YARN/hadoop-2.2.0
```

Step 4) Creating Hive directories in Hadoop:

To communicate with Hadoop, we need to create directories in Hadoop as shown below.

```
guru99hive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -mkdir /usr/hive/warehouse
```

Giving root permissions to create Hive folders in Hadoop.If it doesn't throw any error message, then it means that Hadoop has successfully given permissions to Hive folders.

```
guru99hive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -chmod g+w /usr/hive/warehouse
```

Step 5) Getting into Hive shell by entering './hive' command as shown in below.

```
guru99hive@ubuntu:~/apache-hive-1.2.0-bin/bin$ ./hive
Logging initialized using configuration in jar:file:/home/datamatics/apache-hive-1.2.0-bin/lib/hive-common-1.2.0.jar!/hive-log4j.properties
Java HotSpot(TM) 64-Bit Server VM warning: You have loaded library /home/datamatics/Hadoop_YARN/hadoop-2.2.0/lib/native/libhadoop.so.1.0.0 which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
hive>
```

Theory:

Hive shell commands

Here we are going to create sample table using Hive shell command "create" with column names.

Sample Code for creating data base in Hive

The screenshot shows a Hive shell session. The first command is 'create table product(product int, pname string, price float);' followed by 'row format delimited;' and 'fields terminated by ',';'. This is annotated with a red circle '1' and a callout bubble stating 'Creation of table 'product' with 3 column values'. The second command is 'describe product;', annotated with a red circle '2' and a callout bubble stating 'description about table 'product''. The output shows the table schema: product (int), pname (string), price (float). It also shows the time taken (0.403 seconds) and the number of rows fetched (3 row(s)).

```
hive> create table product(product int, pname string, price float)
> row format delimited
> fields terminated by ',';
OK
Time taken: 0.434 seconds
hive> describe product;
OK
product          int
pname            string
price            float
Time taken: 0.403 seconds, Fetched: 3 row(s)
hive>
```

From the above screen shot we can observe the following:

1. Creation of Sample Table with column names in Hive

- Here the table name is "product" with three column names **product**, **pname**, and **price**
- The three column names denoted by their respective data type
- All fields are terminated by coma ','

2. Displaying Hive Table information

- Using "describe" command we can able to see the table information present in Hive
- Here it is displaying column names with their respective data types present in table schema
- At the end, it will display time to perform this command and number of rows it fetched

Sample Code for creating data base in Hive (For self check)

1) Create table product(product int, pname string, price float)

Row format delimited

Fields terminated by ',';

2) describe product:

Data types in Hive

Data types are very important elements in Hive query language and data modeling. For defining the table column types, we must have to know about the data types and its usage.

The following gives brief overview of some data types present 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

Creation and dropping of Database in Hive:

Create Database:

For creating database in Hive shell, we have to use the command as shown in syntax below:-

Syntax:**Create database <DatabaseName>**

Example: -Create database "guru99"

The screenshot shows a Hive shell session. The first command is `hive> show databases;`, which returns `OK` and `default`. The second command is `hive> create database guru99`, which returns `OK`. The third command is `hive> show databases;`, which returns `OK` and lists `default` and `guru99`. Annotations in orange speech bubbles point to specific parts of the output: 'Creating Database name 'guru99'' points to the `create database guru99` command; 'Displaying Existing databases' points to the second `show databases;` command; and 'database "guru99"' points to the `guru99` entry in the output of the second `show databases;` command.

```
hive> show databases;
OK
default
Time taken: 0.994 seconds, Fetched: 1 row(s)
hive> create database guru99
> ;
OK
Time taken: 0.292 seconds
hive> show databases;
OK
default
guru99
Time taken: 0.042 seconds, Fetched: 2 row(s)
```

From the above screen shot, we are doing two things

- Creating database "guru99" in Hive
- Displaying existing databases using "show" command
- In the same screen, Database "guru99" name is displayed at the end when we execute the show command. Which means Database "guru99" is successfully created.

Drop Database:

For Dropping database in Hive shell, we have to use the "**drop**" command as shown in syntax below:-

Syntax:**Drop database <DatabaseName>****Example:-****Drop database guru99**

```
hive> drop database guru99;
OK
Time taken: 0.819 seconds
hive> show databases;;
OK
default
Time taken: 0.014 seconds, Fetched: 1 row(s)
```

Dropping Database
"guru99"

In the above screenshot, we are doing two things

- We are dropping database 'guru99' from Hive
- Cross checking the same with "show" command
- In the same screen, after checking databases with show command, database "guru99" does not appear inside Hive.
- So we can confirm now that database "guru99" is dropped

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:

Coming to **Tables** it's just like the way that we create in traditional Relational Databases. The functionalities such as filtering, joins can be performed on the tables.

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:

- 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_internaltable (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;
```

If you dropped the guruhive_internaltable, including its metadata and its data will be deleted from Hive.

From the following screenshot, we can observe the output

```

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"

In above code and from screen shot we do following things,

- 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

Sample code Snippet for External Table

1. Create External table

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

Row format delimited

Fields terminated by '\t'

```
LOCATION '/user/guru99hive/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/guru99hive/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;
```

```
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
```

creation of table with "external" key word

Loading data into external table "guruhive_external"

Displaying external table "guruhive_external"

Dropping table "guruhive_external"

Conclusion:

Sample Questions:

- 1) Explain Hadoop Hive.
- 2) Explain Hadoop job execution flow.
- 3) Explain Hadoop HIVE architecture.
- 4) Explain Characteristics of Hadoop HIVE.
- 5) Explain Hadoop HIVE DML commands.
- 6) Explain Hadoop HIVE DDL commands.

EXPERIMENT NO: 10**Title:** Study of Pie chart in R**Aim:** Understanding of data visualization in R.**Theory:****R – Pie Charts:**

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportions. It depicts a special chart that uses “pie slices”, where each sector shows the relative sizes of data. A circular chart cuts in a form of radii into segments describing relative frequencies or magnitude also known as a circle graph.

R – Pie Charts:

R Programming Language uses the function `pie()` to create pie charts. It takes positive numbers as a vector input.

Syntax: `pie(x, labels, radius, main, col, clockwise)`

Parameters:

- `x`: This parameter is a vector that contains the numeric values which are used in the pie chart.
- `labels`: This parameter gives the description to the slices in pie chart.
- `radius`: This parameter is used to indicate the radius of the circle of the pie chart.(value between -1 and +1).
- `main`: This parameter is represents title of the pie chart.
- `clockwise`: This parameter contains the logical value which indicates whether the slices are drawn clockwise or in anti clockwise direction.
- `col`: This parameter give colors to the pie in the graph.

Creating a simple pie chart:

To create a simple pie chart:

By using the above parameters, we can draw a pie chart.

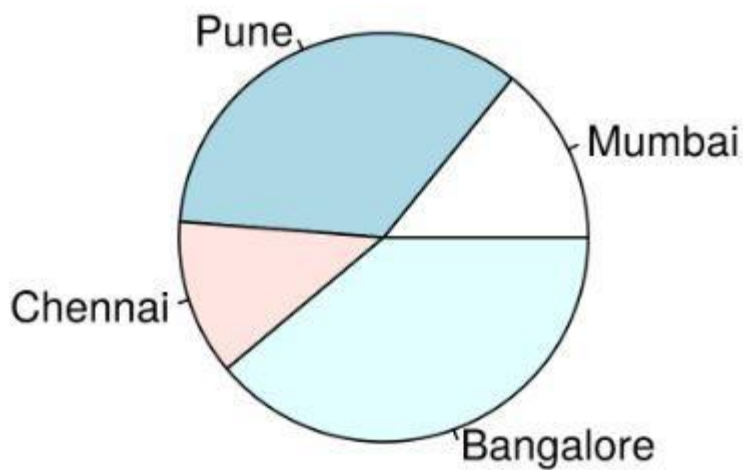
It can be described by giving simple labels.

Example:

```
# Create data for the graph.  
geeks<- c(23, 56, 20, 63)  
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")
```

```
# Plot the chart.  
pie(geeks, labels)
```

Output:



Pie chart including the title and colors:

To create color and title pie chart.

Take all parameters which are required to make pie chart by giving a title to the chart and add labels.

We can add more features by adding more parameters with more colors to the points.

Example:

```
# Create data for the graph.  
geeks<- c(23, 56, 20, 63)
```

```
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")
```

```
# Plot the chart with title and rainbow
```

```
# color pallet.
```

```
pie(geeks, labels, main = "City pie chart",
```

```
col = rainbow(length(geeks)))
```

Output:



Slice Percentage & Chart Legend:

To create chart legend and slice percentage, we can plot by doing the below methods.

There are two more properties of the pie chart:

slice percentage

chart legend.

We can show the chart in the form of percentages as well as add legends.

Example:

```
# Create data for the graph.
```

```
geeks <- c(23, 56, 20, 63)
```

```
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")
```

```
piepercent<- round(100 * geeks / sum(geeks), 1)
```

```
# Plot the chart.
```

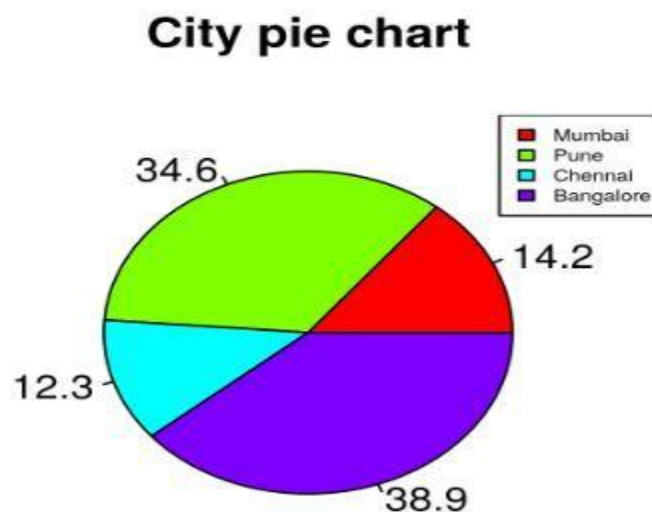
```
pie(geeks, labels = piepercent,
```

```
    main = "City pie chart", col = rainbow(length(geeks)))
```

```
legend("topright", c("Mumbai", "Pune", "Chennai", "Bangalore"),
```

```
      cex = 0.5, fill = rainbow(length(geeks)))
```

Output:



3D Pie Chart:

Here we are going to create a 3D Pie chart using plotrix package and then we will use pie3D() function to plot 3D plot.

Get the library.

```
library(plotrix)
```

Create data for the graph.

```
geeks <- c(23, 56, 20, 63)
```

```
labels <- c("Mumbai", "Pune", "Chennai", "Bangalore")
```

```
piepercent<- round(100 * geeks / sum(geeks), 1)
```

Plot the chart.

```
pie3D(geeks, labels = piepercent,
```

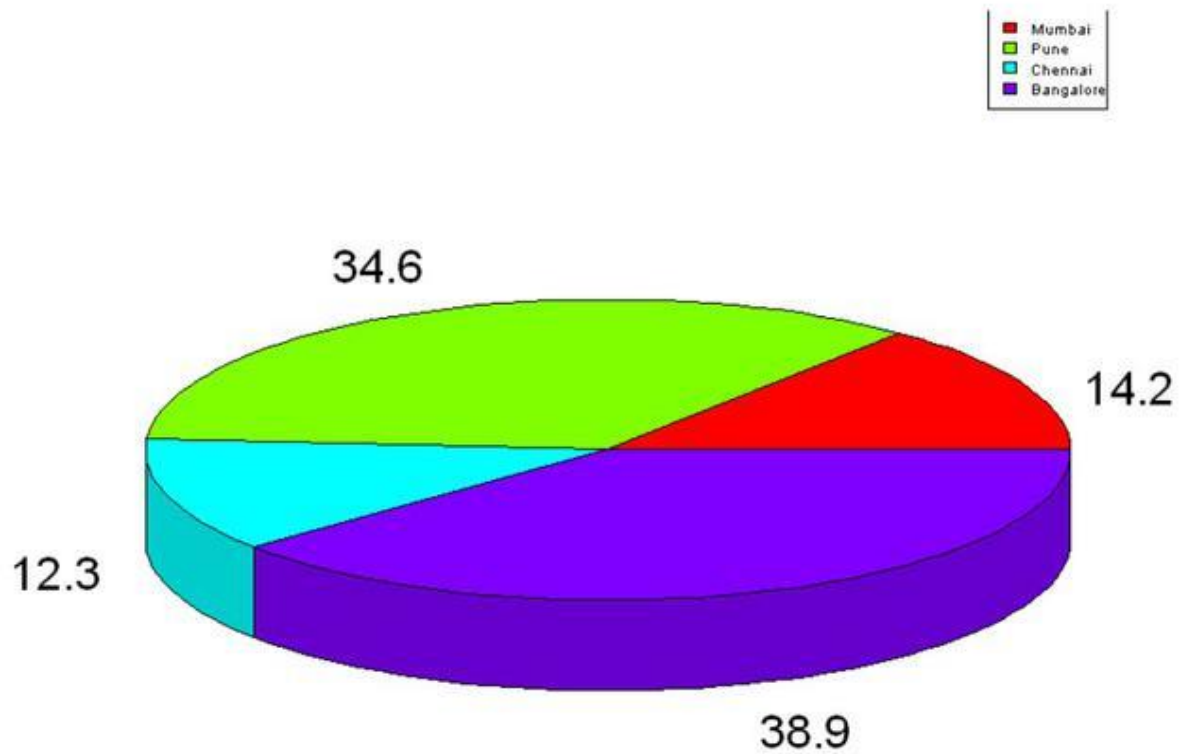
```
      main = "City pie chart", col = rainbow(length(geeks)))
```

```
legend("topright", c("Mumbai", "Pune", "Chennai", "Bangalore"),
```

```
      cex = 0.5, fill = rainbow(length(geeks)))
```

Output:

City pie chart



Conclusions:

Sample Questions:

- 1) Describe data visualization in R.
- 2) Explain Pie charts in R.