* Hive :->

Hive is a data warehouse System which is used to analyze structured data. It is built on the top of Hadoop. It was developed by facebook.
Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive Query Language). which gets internally converted to MapReduce Jobs.
Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs.
Hive ~~supp~~ Supports Data Definition Language (DDL), Data Manipulation Language (DML) and User Defined functions (UDF).

* features of Hive :->

There are the following features of Hive :-

1. Hive is fast and scalable.
2. It provides SQL - like queries (i.e. HQL) that are implicitly transformed to MapReduce or Spark jobs.
3. It is capable of analyzing large datasets stored in HDFS.
4. It allows different storage types such as plain text, RCFile and HBase.
   ↳ (Record Columnar file)
5. It uses indexing to accelerate queries.
6. It can operate on compressed data stored in the Hadoop ecosystem.

7. It supports user-defined functions (UDFs) where user can provide its functionality.

* Limitations of Hive :->

1. Hive is not Capable of handling real-time data.
2. It is not designed for online transaction processing.
3. Hive queries contain high latency.

* Difference between Hive and Pig :->

	Hive	Pig
1.	Hive is Commonly used by Data Analysts.	Pig is Commonly used by Programmers.
2.	It follows Sql-like queries	It follows the data-flow language.
3.	It Can handle Structured data.	It Can handle Semi-structured data.
4.	It works on server-side of HDFS Cluster.	It works on client-side of HDFS Cluster.
5.	Hive is slower than Pig.	Pig is Comparatively faster then Hive.

* Modes of Hive :->

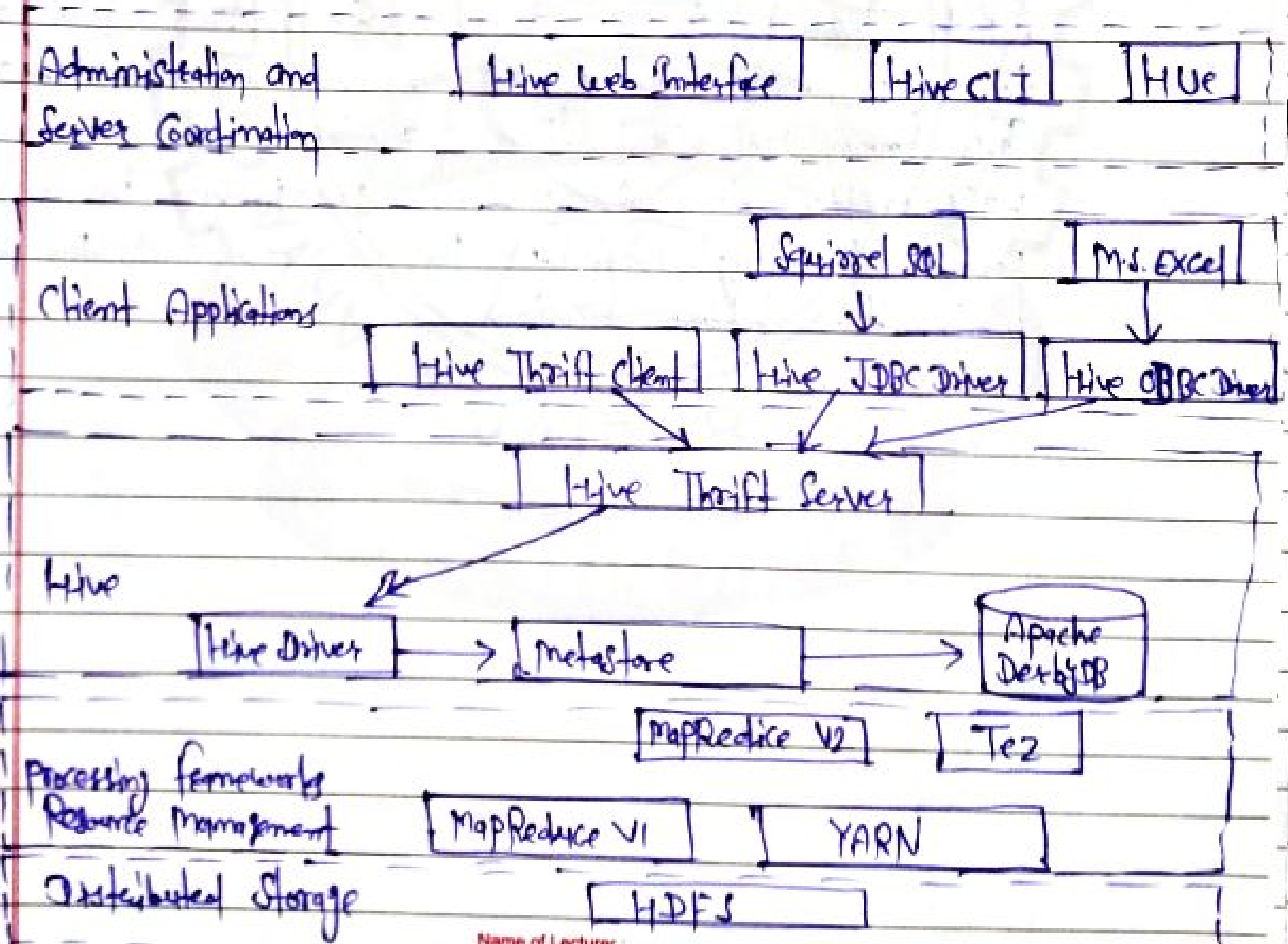
Hive Can operate in two modes depending on the size of data Nodes in Hadoop.

These modes are:

1. Local mode
2. Map reduce mode

* ~~How the~~ Hive is put Together :->

In this we illustrate the architecture of Apache Hive and explain its Various Components, as shown in the illustration in ~~figure~~ following figure:



In the above figure we can see at the bottom the Hive sits on top of the Hadoop Distributed File system (HDFS) and MapReduce systems. In the case of MapReduce, figure-1 shows both the Hadoop1 and Hadoop2 Components.

With Hadoop 1, Hive queries are converted to MapReduce code and executed using the MapReduce V1 (MRV1) infrastructure, like the JobTracker and TaskTracker.

With Hadoop 2, YARN has decoupled resource management and scheduling from the MapReduce framework.

Hive queries can still be converted to MapReduce code and executed, now with MapReduce ~~framework.~~ V2 (MRV2) and the YARN infrastructure.

There is a new framework under development called Apache Tez, which is designed to improve Hive performance for batch-style queries and support smaller interactive (also known as real-time) queries.

HDFS provides the storage, and MapReduce provides the parallel processing capability for higher-level functions within the Hadoop Ecosystem.

* Getting Started with Apache Hive :→

Installation :→

The setup steps seem Something like this:

1. Download the latest Hive →
   we downloaded Hive Version 11.0. You also need the Hadoop and mapReduce Subsystems, So be Sure to Complete Step-2

2. Download Hadoop Version 1.2.1

3. Using the Commands in the following listing, Place the releases in Separate directories, and then uncompress and untar them.

(untar is one of those pesky unix terms which Simply means to expand an achived Software package.)

```
$ mkdir hadoop ; cp hadoop -1.2.1 .tar.gz hadoop ; cd hadoop
$ gunzip hadoop -1.2.1 . tar.gz
$ tar xvf *. tar
$ mkdir hive ; cp hive -0.11.0 .tar.gz hive ; cd hive
$ gunzip hive -0.11.0 . tar.gz
$ tar xvf * . tar
```

4. Using the Commands in the following listing, set up your Apache Hive Environment Variable, including ~~hado~~ HADOOP-HOME, JAVA-HOME, HIVE-HOME and PATH, in your Shell profile script:

```
export HADOOP-HOME =/Home/user/Hive/hadoop/hadoop-1.2.1
export JAVA-HOME =/opt/jdk
export HIVE-HOME =/home/user/hive/hive-0.11.0
export PATH = $HADOOP-HOME/bin: $HIVE-HOME/bin: $JAVA_HOME
        /bin: $PATH
```

5. Create the Hive Configuration file that you'll use to define Specific Hive Configuration Settings

```
$ Cd $HIVE-HOME/Conf
$ Cp hive.default.xml.template hive-site.xml
 <?xml version ="1.0"?>
 <?xml-stylesheet type = "text/xsl" href = "Configuration.xsl"?>
 <Configuration>
 <!-- hive Execution Parameters ->
 <property>
   <name> hive.metastore.warehouse.dir </name>
   <value>/home/biadmin/hive/warehouse </value>
   <description> location of default database for the warehouse
 </description>
 </property>
</configuration>
```

Because you're running Hive in stand-alone mode on a virtual machine rather than in a real-life Apache Hadoop Cluster, Configure the System to use local storage rather then the HDFS : Simply set the hive.metastore.warehouse.dir parameter.

When you start a Hive client, the $HIVE-HOME Environment variable tells the client that it should look for your Configuration file <hive-site.xml> in the conf directory.

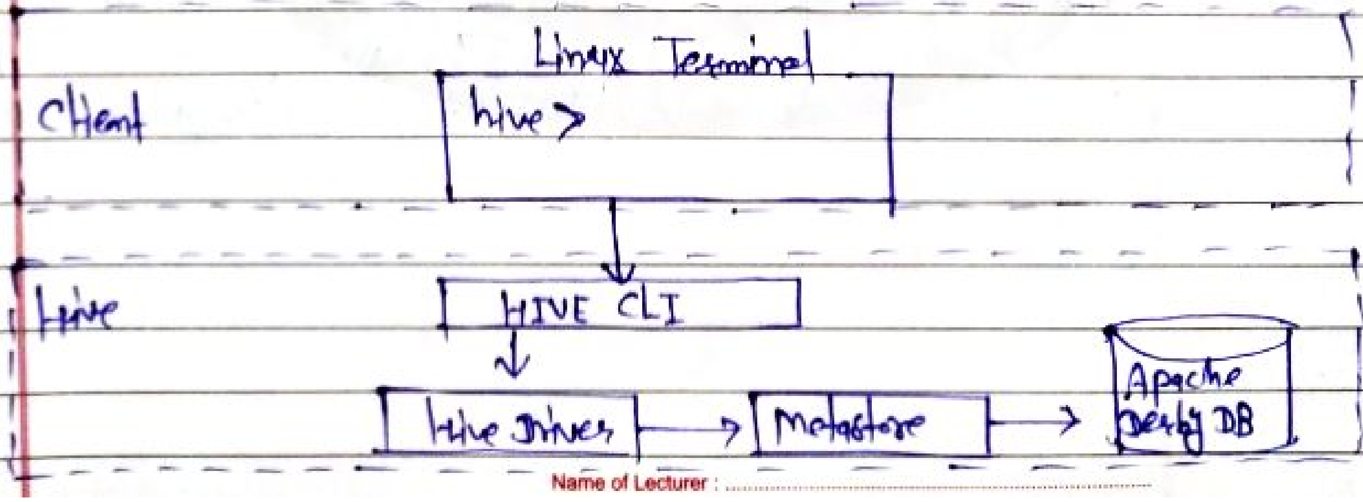
* Examining the Hive Clients :→

In general, we have the following hive clients :

1. Hive Command-line interface (CLI)
2. Hive web Interface (HWI) Server
3. SquirreL

1. The Hive CLI Client :→

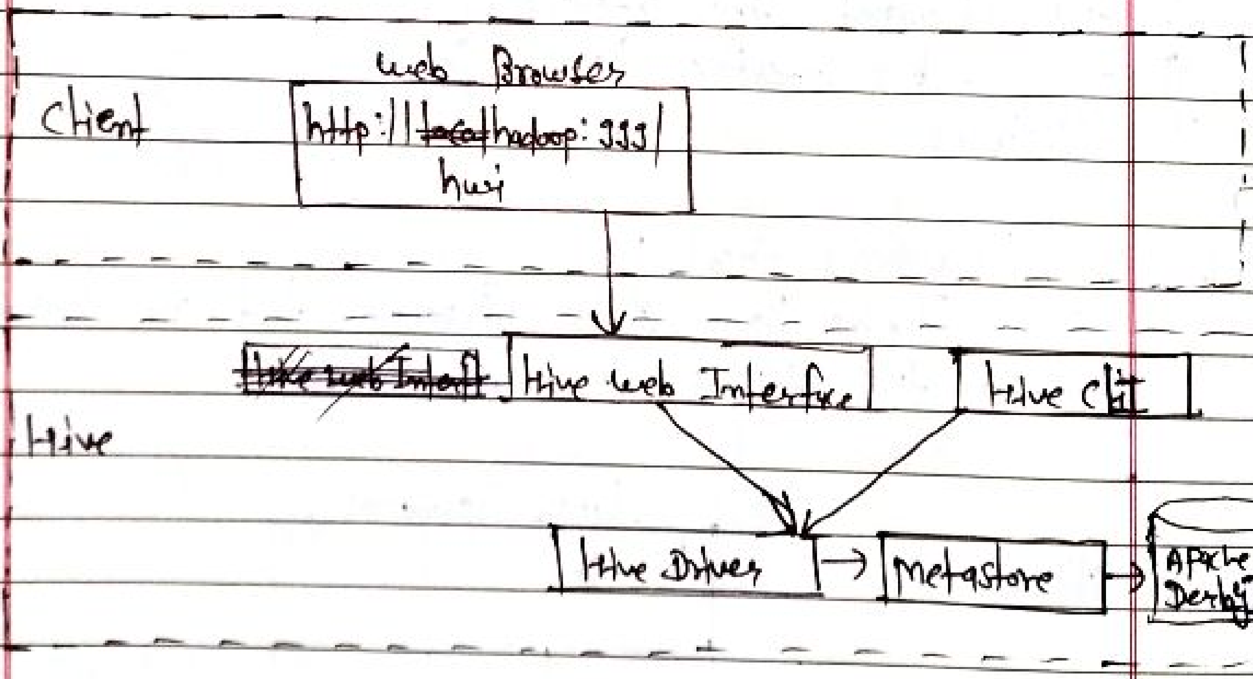
The following figure shows Components that are required when running the CLI on a Hadoop Cluster.



The examples in this chapter, we run Hive in local mode which uses local storage, rather then the HDFS, for your data. To run the HIVE ~~CL~~ CLI, you execute the hive command and specify the CLI as the service you want to run.

9. Hive web Interface (HWI) Server:→

When we want to access Hive using a web browser, you first need to start Hive web Interface (HWI) Server and then point your browser to the port on which the Server is listening. following figure shows the HWI client configuration.



The following steps shows you what you need to do before you can start the HWI Server:

1. Configure the $HIVE-HOME/conf/hive-site.xml file as below to ensure that Hive can find and load the HWI's Java Server pages.

```
<property>
<name> hive.hwi.war.file </name>
<value> ${HIVE-HOME}/lib/hive-hwi.war </value>
<description>
 This is the WAR file with the jsp content for Hive web
  Interface
</description>
</property>
```

2. The HWI Server requires Apache Ant libraries to run, so download Ant.

3. Install Ant using the following commands:

```
mkdir ant
cp apache-ant-1.9.2-bin.tar.gz ant ; cd ant
gunzip apache-ant-1.9.2-bin.tar.gz
tar xvf apache-ant-1.9.2-bin.tar
```

4. Set the $ANT-LIB environment variable and start the HWI Server by using following commands:

```
$ export ANT_LIB = /home/user/ant/apache-ant-1.9.2/lib
$ bin/hive --service hwi
```

3. Squirrel as Hive client with the JDBC Driver:→

The last HIVE client is the open source tool Squirrel SQL. It provides a user interface to Hive and simplifies the tasks of querying large tables and analyzing data with Apache Hive.

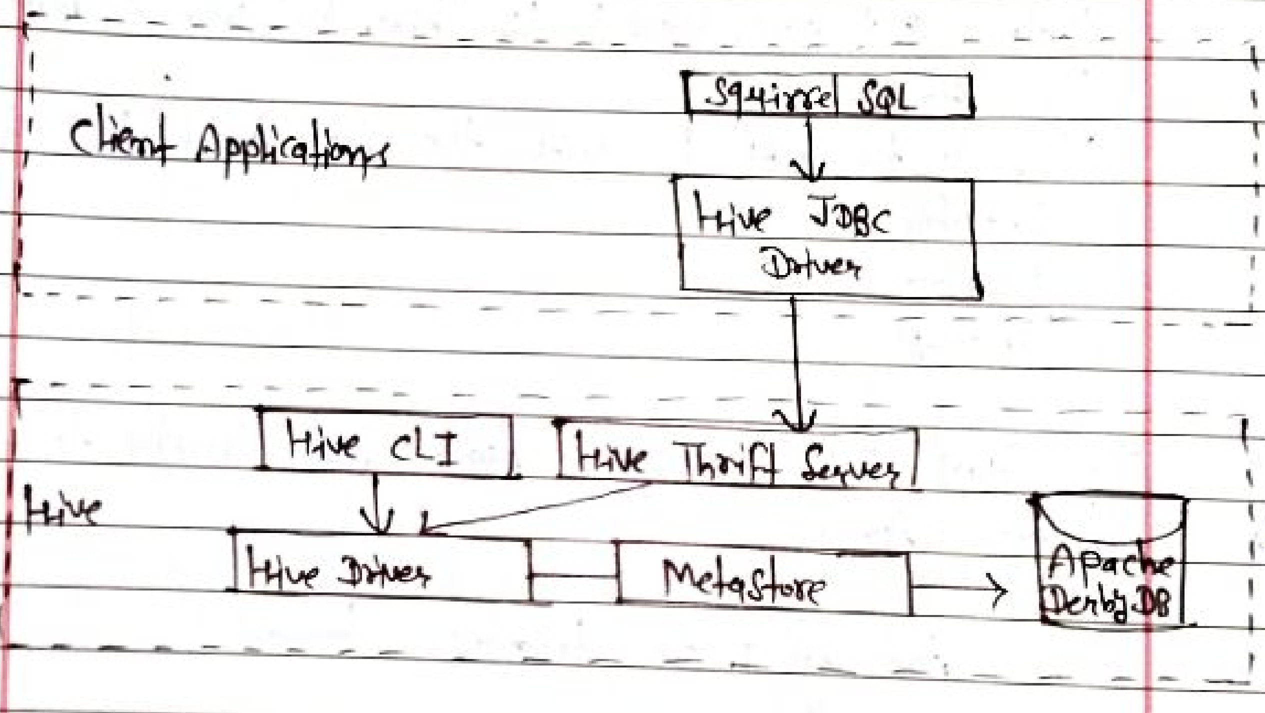


figure:- Using the Squirrel client with Apache Hive.

In the above figure, we can see that the Squirrel client uses the JDBC APIs to pass commands to the Hive Driver by way of the Hive Thrift Server.

* Hive Data types :→

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 in Hive ~~then~~ these are:-

1. Numeric Types
2. String Types
3. Date/Time Types.
4. Complex Types

1. Numeric Types :→

	Type	Memory Allocation
(1.)	~~Tin~~ TINY INT	Its 1-byte Signed Integer (-128 to 127)
(2)	SMALL INT	~~2-byt~~ byte Signed integer (-32768 to 32767)
(3)	INT	4-byte Signed Integer
(4)	BIG INT	8-byte Signed integer
(5)	FLOAT	4-byte Signed precision floating point number
(6)	DOUBLE	8-byte double precision floating point number
(7)	DECIMAL	we Can define precision and Scale in the type.

2. String Types :->

	Type	Length
(1)	CHAR	255
(2)	VARCHAR	1 to 65355
(3)	STRING	we Can define length here (no limit)

3. Date / Time types :->

	Type	Usage
(1)	Timestamp	Supports traditional unix timestamp with optional nanosecond
(2)	Date	• 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

4. Complex Types :->

	Type	Usage
(1)	Arrays	Array<data_type> Negative values and non-Constant expressions not allowed
(2)	Maps	MAP<primitive_type, data_type> Negative values and non-Constant expressions not allowed.
(3)	Structs	STRUCT <col_name : datat-type, ...>
(4)	Union	UNIONTYPE <data-type, datat_type, ...>

* Creating and Managing Databases and Tables :->

→ Create Database :->

Create Database is a statement used to create a database in Hive. A database in Hive is a namespace or a collection of tables.

Syntax of the Create Database :->

```
CREATE DATABASE | SCHEMA [IF NOT EXISTS] <database name>
```

⇒ Drop Database Statement :->

Drop Database is a statement that drops all the tables and deletes the database. Its Syntax is as follows:

```
DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS]
  database_name [RESTRICT|CASCADE];
```

The following queries are used to drop a database. Let us assume that the database name is userdb.

```
hive> DROP DATABASE IF EXISTS userdb;
```

The following query drops the database using CASCADE. It means dropping respective tables before dropping the database.

[<hive> DROP DATABASE IF EXISTS userdb CASCADE;]

The following query drops the database using SCHEMA.

[hive> DROP SCHEMA userdb;]

⇒ Create Table Statement :→

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:-

Syntax:→

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS]
  [db-name.] table-name
[~~col~~(col-name data-type [~~Co~~COMMENT col-comment],...)]
[COMMENT table-comment]
[ROW FORMAT row-format]
[STORED AS file-format]
```

~~In Lets we assume~~

Example:-

~~Lets we assume here give the data~~

Example

Let us assume you need to Create a table named Employee using CREATE TABLE statement. The following table lists the fields and their data types in Employee table:

Sr. No.	Field Name	Data Type
1	Eid	int
2	Name	String
3	Salary	float
4	~~Dest~~ Designation	String.

The following data is a Comment, Row formatted fields such as field terminator, lines terminator ~~or~~ and Stored file type.

```
COMMENT 'Employee details'
 FIELDS TERMINATED BY '\t'
 LINES TERMINATED BY '\n'
 STORED IN TEXT FILE
```

The following query Creates a table named Employee using the above data.

```
CREATE TABLE IF NOT EXISTS employee (eid int, name String
   Salary String, destination String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
```

```
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

⇒ Load Data Statement →

Generally, after Creating a table in SQL, we Can Insert data using the Insert Statement. But in Hive, we Can insert data using the Load DATA Statement.

while inserting data into Hive, it better to use LOAD DATA to store bulk records.

~~There are two ways to load data:~~

~~One 1. from local file System~~

~~2. from Hadoop file System.~~

Syntax:-

The syntax for load data is as follows:

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO
 TABLE tablename [PARTITION (partcol1 = Val1, partcol2=
              Val2 ...)]
```

- LOCAL is identifier to specify the local path. It is optional.
- OVERWRITE is optional to overwrite the data in the table.
- PARTITION is optional.

Example :->

we will insert the following data into the table. It is a text file named sample.txt in the ~~directory~~ /home/user directory.

```
1201    Gopal     50000    OP Admin
1202    Ram       45000    Hr admin.
1203    Shyam     55000    Proof reader
1204    Mohan     35000    Technical Manager
1205    Manisha   60000    Branch Manager
```

The following query loads the given text into the table.

```
hive> LOAD DATA LOCAL ~~P~~INPATH '/home/user/sample.txt'
  OVERWRITE INTO TABLE employee;
```

On successful download, you get to see the following respons

```
ok
Time taken: 15.905 seconds
hive>
```

* Alter Table Statement :→

It is used to alter a table in Hive.

Syntax :-

The Statement takes any of the following Syntaxes based on what attributes we wish to modify in a table.

```
ALTER TABLE name RENAME To new_name
ALTER TABLE name ADD COLUMNS (Col_spec [, Col_spec ...])
ALTER TABLE name DROP [COLUMN] Column_name
ALTER TABLE name CHANGE Column_name new_name new_type
ALTER TABLE name REPLACE Columns (Col_spec [, Col_Spec ...])
```

Rename To... statement :→

The following query renames the table from Employee to emp.

```
ALTER TABLE employee RENAME To emp;
```

* Change Statement :→

The following table Contains the fields of employee table and it shows the fields to be changed (in Black)

Field Name	Convert from Data type	change Field Name	Convert to Data type
eid	int	eid	int
name	String	Ename	String
Salary	float	Salary	Double
designation	String	designation	String

The following queries rename the Column name and Column data type using the above data.

```
ALTER TABLE Employee CHANGE name Ename String ;
ALTER TABLE Employee CHANGE Salary Salary Double ;
```

* Add Columns Statement :→

The following query adds a Column named dept to the Employee table.

```
hive> ALTER TABLE employee ADD ~~Colu~~ COLUMNS (
      dept STRING COMMENT 'Department name');
```

* Replace Statement :->

The following query deletes all the Columns from the Employee table and replaces it with Emp and name Columns:

```
hive> ALTER TABLE Employee REPLACE COLUMNS (
          Eid INT Empid Int,
         Ename STRING name String);
```

* Drop TABLE Statement :->

The Syntax is as follows :-

```
[ DROP TABLE [IF EXISTS] table-name;  ]
```

The following query drops a table named Employee

```
hive> DROP TABLE IF EXISTS Employee;
```

* Hive Data Manipulation Language (DML) Commands :->

Hive DML (Data Manipulation Language) Commands are used to Insert, update, retrieve, and delete data from the Hive table once the table and database Schema has been defined using Hive DDL Commands.

The Various Hive DML Commands are :

1. ~~Load~~ LOAD
2. SELECT
3. INSERT
4. DELETE
5. UPDATE
6. EXPORT
7. IMPORT

1. LOAD Command :->

The LOAD Statement in Hive is used to move data files into the locations Corresponding to Hive tables.

→ If a LOCAL keyword is Specified, then the LOAD Command will look for the file Path in the local file system.

→ If the LOCAL keyword is not Specified, then the Hive will need the absolute URI of the files.

→ In Case the keyword OVERWRITE is Specified, then the Contents of the target table / Partition will be deleted and replaced by the files referred by file path.

→ If the OVERWRITE keyword is not Specified, then the files referred by file path will be appended to the table.

Syntax :->

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE
tablename [PARTITION] (partcol1 = val1 , partcol2 = val2 ...) ];
```

Example ->

Here we are trying to load data from the 'dab' file in the local filesystem to the 'emp_data' table.

```
> LOAD DATA LOCAL INPATH '/home/dataflair/dab' INTO TABLE
    emp_data;
```

9. SELECT COMMAND :->

The SELECT statement in Hive is similar to the SELECT statement in SQL used for retrieving data from the database.

Syntax :->

```
SELECT col1, col2 FROM tablename;
```

Example :->

```
SELECT * from emp_data;
```

3. INSERT Command :→

The INSERT Command in Hive loads the data into a Hive table. we Can do insert to both the Hive table or partition.

(a) INSERT INTo :→

The INSERT INTO Statement appends the data into existing data in the table or partition. INSERT INTO Statement work's from Hive Version 0.8.

Syntax →

```
[ INSERT INTO TABLE tablename1 [PARTION (partcol1 = Val1,
  Partcol2 = Val2 ...)] Select_statement1 FROM from_Statemet; ]
```

Example :→

Here in this example, we are trying to Insert the data of 'emp-data' table Created above into the table 'example'.

firstly we Create the table :→

```
> CREATE TABLE IF NOT EXISTS example ( id STRING, name
  STRING, dep STRING, state STRING, Salary STRING, year
  STRING);
```

Now Insert Statement to load data into table "example".

```
[ > INSERT INTO TABLE example SELECT emp.emp_id , emp.emp_name
  , emp.emp_dep , emp.state , emp.salary , emp.year_of_joining FROM
  emp_data emp ; ]
```

(b) INSERT OVERWRITE :→

The INSERT OVERWRITE table overwrites the existing data in the table or partition.

Syntax :-

```
[ INSERT OVERWRITE TABLE tablename1 [ PARTITION (partcol1 =
    val1, ...) [ IF NOT EXISTS ]] select_statement FROM
  from_statement ;        ]
```

Example :→

Here we are overwriting the existing data of the table 'example' with the data of table/ Database 'dummy' using INSERT OVERWRITE statement.

```
> INSERT OVERWRITE TABLE example SELECT dmy.enroll, dmy.
  name, dmy.department, dmy.state, dmy.salary, dmy.year
  FROM dummy dmy;
```

By using the SELECT statement we can verify whether the existing data of the table 'example' is overwritten by the data of table 'dummy' or not.

```
set SELECT * from example;
```

(C). INSERT .. VALUES :→

INSERT .. VALUES statement in Hive inserts data into the table directly from SQL. It is available from Hive 0.14.

Syntax →

```
INSERT INTO TABLE tablename [PARTITION (partCol1 [= Val1],
partCol2 [= Val2] ...)] VALUES values_row [, values_row ...]
```

Example :→

Inserting data into the 'student' table using INSERT .. VALUES statement.

```
> INSERT INTO TABLE student VALUES (101, 'Callen', 'IT', '7.8'),
  (103, 'Joseph', 'CS', '8.2'), (105, 'Alex', 'IT', '7.9');
```

```
> SELECT * FROM student ;
```

4. DELETE Command :→

The DELETE statement in Hive deletes the table data. If the WHERE Clause is Specified, then it deletes the rows that satisfy the Condition in where Clause.

The DELETE statement can only be used on the hive tables that Support ACID (ACID Property → Atomicity, Consistency, Isolation, Durability).

Syntax →

```
DELETE FROM tablename [WHERE expression];
```

Example :→

In the below Example, we are deleting the data of the Student from table 'Student' whose roll_no is 105.

```
> DELETE FROM Student WHERE roll_no = 105;
```

By using the SELECT Statement we can Verify whether the data of the Student from table 'Student' whose roll_no is 105 is deleted or not.

```
> SELECT * FROM Student;
```

5. UPDATE Command :→

The update can be performed on the hive tables that support ACID.

The update statement in Hive deletes the table data. If the WHERE clause is specified, then it updates the column of the rows that satisfy the condition in WHERE clause.

Syntax:-

```
    UPDATE tablename SET column = value [, column = value...]
   [WHERE expression];
```

Example:→

In this example, we are updating the branch of the student whose roll_no is 103 in the 'student' table using an ~~update~~ UPDATE statement.

```
> UPDATE student SET branch = 'IT' WHERE roll_no = 103;
```

6. ~~Export~~ EXPORT Command :→

The Hive EXPORT statement exports the table or partition data along with the metadata to the specified output location in the HDFS.

metadata is exported in a _metadata file, and data is exported in a subdirectory 'data'.

Hive Partitions:→

Apache Hive organizes table into partitions. Partitioning is a way of dividing a table into related parts based on the values of particular columns like date, city and department. Each table in the hive can have one or more partition keys to identify a particular partition. Using partition it is easy to do queries on slices of the data.

Syntax:→

Export

```
EXPORT TABLE tablename [PARTITION (part_column="value"
  [, ...])] TO 'export_target_path' [FOR replication
  ('eventid')];
```

Example:→

Here in this example, we are exporting the student table to the HDFS directory "export_from_hive".

```
>EXPORT TABLE student TO 'export_from_hive';
```

The table successfully exported. You can check for the _metadata file and data sub-directory using ls command.

```
$ hadoop fs -ls -R /user/dataflair/export_from_hive/;
```

~~7. Imp~~

7. IMPORT Command :→

The Hive IMPORT Command imports the data from a specified location to a new table or already existing table.

Syntax :→

```
IMPORT [[EXTERNAL] TABLE new_or_original_tablename
 [PARTITION (part_column = "value" [, ...])]]
FROM 'source_path' [LOCATION 'import_target_path'];
```

Example :→

Here in this example, we are importing the data exported in the above example into a new Hive table 'imported_table'.

```
> ~~import~~ IMPORT TABLE imported_table from '/user/dataflair/export_
        from_hive';
```

* Querying and Analyzing Data :->

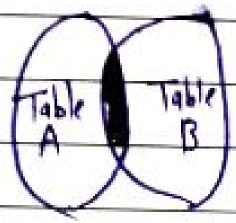
- Hive data types, Hive's DDL and Hive's DML, helps Creating and ~~m~~ managing tables but now we help you explore some HiveQL features for querying and analysing data. We begin by exploring table joins in Hive.

Joining tables with Hive :->

- In relational database modelling, we split the tables for normalization purpose. and use Join operataion get the data when it is required.
- Database normalization is a technique that guards against data loss, redundancy, and other anomalies as data is update and retrieved.
- MapReduce is the engine for joining tables, and the Hadoop file System (HDFS) is the underlying storage.
- Hive table reads and writes via HDFS usually involve very large blocks of data, more data you can manage altogether in one table, the better the overall performance.
- with this background information in mind, we can tackle making joins with Hive. fortunately, the Hive development Community was realistic and understood that users would want and need to join tables with HiveQL.
- Hive supports equijoins, a specific type of join that only uses equality comparisons in the join predicate.
- This ~~restcid~~ restriction is only because of limitations on the underlying MapReduce engine.

Hive Join Example :-

1. Inner Join :→ ~~Full outer Join :→~~



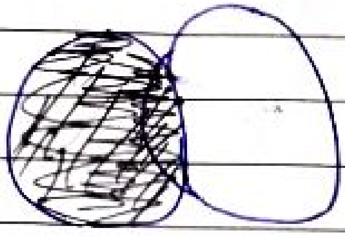
Basically, to Combine and retrieve the records from multiple tables we use Hive Join clause. Moreover, in SQL JOIN is as same as OUTER JOIN. Moreover, by using the primary keys and foreign keys of the tables JOIN Condition is to be raised.
Furthermore, the below query executes JOIN the CUSTOMER and ORDER tables. Then further retrieves the records'

```
> SELECT C.ID, C.Name, C.AGE, O.AMOUNT FROM CUSTOMER
  c JOIN ORDERS o ON (C.ID = O.CUSTOMER_ID);
```

2. Left Outer Join :→

On defining HiveQL Left outer Join, even if there are no matches in the right table it returns all the rows from the left table.
In additional, it returns all the values from the left table. Also, the matched values from the right table, or NULL in case of no matching JOIN predicate.
However, the below query shows LEFT OUTER JOIN between CUSTOMER as well as ORDER tables'

```
> SELECT C.ID, C.Name, O.Amount, O.DATE FROM CUSTOMERS
  C LEFT OUTER JOIN ORDERS O ON (C.ID = O.CUSTOMER
  ID);
```

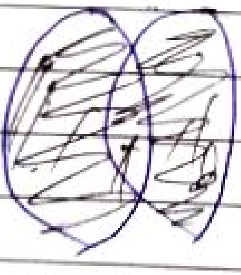


3. Full Outer Join :->

The major purpose of this HiveQL full outer Join is it combines the records of both the left and the right outer tables which fulfills the Hive JOIN condition. Moreover, this Joined table contains either all the records from both the tables or fills in NULL Values for missing matches on either side.

However, the below query shows FULL OUTER JOIN between CUSTOMER as well as ORDER tables:

```
> SELECT C.ID, C.NAME, O.AMOUNT, O.DATE FROM
  CUSTOMERS C FULL OUTER JOIN ORDERS O ON
  (C.ID = O.CUSTOMER_ID);
```



⇒ Improving your Hive queries with Indexes :→

Creating an Index is Commo practice with relational databases when we want to speed access to a Column or set of Columns in your databas without an index, the database System has to read all rows in the table to find the data we have selected. Indexes become even more essential when the tables grow extremely large. Hive supports index Creation on tables.

⇒ Hive Indexes :→

Hive indexes are implemented as tables. ~~This is why we need to first~~ Create the Index table and then build it to populate the table.

Therefore, we Can use indexes in at least two ways.

1. Count on the system to automatically use indexes that You Create.
2. Rewrite some queries to leverage the new index table.

⇒ Windowing in HiveQL :→

The Concept of windowing, introduced in the SQL : 2003 Standard, allows the SQL programmer to Create a frame from the data against which aggrega and other window functions Can operate.

HiveQL now supports windowing per the SQL Standard.

Examples are quite helpful when explaining windowing and aggregate functions.

- we first discovered this data set was, 'what exactly is the average flight delay per day?" so we created a query in Listing 13-19 that produces the average departure delay per day in 2008.

* Other key HiveQL features :→

1. Security :→

Apache Hive provides a security subsystem that can be quite helpful in preventing accidental data corruption or compromise among trusted members of workgroup.

2. Multi-User Locking :→

Hive supports multi-user warehouse access when configured with Apache Zookeeper. without this support, one user may read a table at the same time another user is deleting that table — which is, obviously, unacceptable.

3. functions :→

HiveQL provides a rich set of built-in operators, built-in functions, built-in aggregate functions, and built-in table-generating functions. Several examples in this chapter use built-in operators as well as built-in aggregate function (AVG, MIN, and Count, for example). To list all built-in functions for any particular Hive ~~relase~~ release, use the SHOW FUNCTIONS HIVEQL command. You can also retrieve information about a built-in function by using the HiveQL commands DESCRIBE FUNCTION function-name and DESCRIBE FUNCTION EXTENDED function-name.

4. Com Compression :→

Data Compression Can not only Save Space on the HDFS but also improve performance by reducing the overall size of input / output operations. Additionally, Compression between the Hadoop mappers and reducers Can improve performance, because less data is passed between nodes in the Cluster. Hive supports intermediate Compression between the mappers and reducers as well as table output Compression. Hive also understands how to ingest Compressed data into the warehouse. files Compressed with Gzip or Bzip2 Can be read by Hive's LOAD DATA Command.