**Hive - Lab practiced**

Hive – Is a Datawarehouse – ELT (Extract Load Transform)  
Hive QL - SQL interface to Hadoop. SQL on top of Hadoop.

Schema on write – RDBMS supports – created a table with 5cols, then we can only insert 5 cols (match input data with schema)

Schema on read – Hive supports – created a table with 5cols, you can insert 10 cols data. (Schema is not verified while writing. When you query the table only, schema will be verified)

Hive storage – does not have own data storage. It uses Hadoop storage for storing the table info.  
  
Hive doesn’t provide crucial features required for OLTP, Online Transaction Processing. It’s closer to being an OLAP tool, Online Analytic Processing.

**Basics of Hive**

[cloudera@quickstart ~]$ hive

hive> create database Jan25;  
  
**hive> CREATE DATABASE IF NOT EXISTS financials;**

**hive> show databases;**

OK

default

jan25

Time taken: 0.021 seconds, Fetched: 2 row(s)  
  
**hive> SHOW DATABASES LIKE 'h.\*';**

**hive> use Jan25;**  
  
**hive> create table students**(

> name string,

> location string,

> phone int,

> course string)

> row format delimited fields terminated by ',';

**hive> show tables**;

OK

students

Time taken: 0.054 seconds, Fetched: 1 row(s)

**hive> load data local inpath** '/home/cloudera/khasimbabu/HIVE\_Excercise/student.txt' into table students;

Loading data to table jan25.students

Table jan25.students stats: [numFiles=1, totalSize=336]

OK

Time taken: 1.526 seconds

**hive> select \* from students;**

OK

abc\_raghu banglore 12345 BigData

bcd\_babu hyderabad 2345 Spark

cds\_jonny london 34567 Flume

eds\_madhu chennai 67889 DataScience

cds\_jonny london 34567 Flume

bcds\_babu hyd 2345 Spark

Time taken: 0.505 seconds, Fetched: 6 row(s)

**hive> describe students;**

OK

name string

location string

phone int

course string

Time taken: 0.116 seconds, Fetched: 4 row(s)  
  
**hive> describe formatted students;**

OK

# col\_name data\_type comment

name string

location string

phone int

course string

# Detailed Table Information

Database: jan25

Owner: cloudera

CreateTime: Tue Jan 26 01:50:50 PST 2021

LastAccessTime: UNKNOWN

Protect Mode: None

Retention: 0

Location: hdfs://quickstart.cloudera:8020/user/hive/warehouse/jan25.db/students

Table Type: MANAGED\_TABLE

Table Parameters:

COLUMN\_STATS\_ACCURATE true

numFiles 1

totalSize 336

transient\_lastDdlTime 1611655223

# Storage Information

SerDe Library: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

InputFormat: org.apache.hadoop.mapred.TextInputFormat

OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

Compressed: No

Num Buckets: -1

Bucket Columns: []

Sort Columns: []

Storage Desc Params:

field.delim ,

serialization.format ,

Time taken: 0.171 seconds, Fetched: 33 row(s)

**hdfs://quickstart.cloudera:8020/user/hive/warehouse/jan25.db/students  
(Hive is creating all these folders inside Hadoop)**

[cloudera@quickstart ~]$ hdfs dfs -ls /user

Found 9 items

drwxr-xr-x - cloudera cloudera 0 2015-11-18 02:56 /user/cloudera

drwxr-xr-x - hdfs supergroup 0 2020-12-02 09:37 /user/hdfs

drwxr-xr-x - mapred hadoop 0 2015-11-18 02:57 /user/history

drwxrwxrwx - hive supergroup 0 2015-11-18 03:01 /user/hive

drwxrwxrwx - hue supergroup 0 2015-11-18 02:58 /user/hue

drwxrwxrwx - jenkins supergroup 0 2015-11-18 02:58 /user/jenkins

drwxrwxrwx - oozie supergroup 0 2015-11-18 02:59 /user/oozie

drwxrwxrwx - root supergroup 0 2015-11-18 02:58 /user/root

drwxr-xr-x - hdfs supergroup 0 2015-11-18 03:01 /user/spark

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive

Found 1 items

drwxrwxrwx - hive supergroup 0 2021-01-26 01:46 /user/hive/warehouse

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse

Found 1 items

drwxrwxrwx - cloudera supergroup 0 2021-01-26 01:50 /user/hive/warehouse/jan25.db

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/jan25.db

Found 1 items

drwxrwxrwx - cloudera supergroup 0 2021-01-26 02:00 /user/hive/warehouse/jan25.db/students

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/jan25.db/students

Found 1 items

-rwxrwxrwx 1 cloudera supergroup 336 2021-01-26 02:00 /user/hive/warehouse/jan25.db/students/student.txt

**When you load the same students data again, Hive don’t check anything, just append the data to the esiting file.**  
hive> load data local inpath '/home/cloudera/khasimbabu/HIVE\_Excercise/student.txt' into table students;

Loading data to table jan25.students

Table jan25.students stats: [numFiles=2, totalSize=672]

OK

Time taken: 0.384 seconds

hive> select \* from students;

OK

abc\_raghu banglore 12345 BigData

bcd\_babu hyderabad 2345 Spark

cds\_jonny london 34567 Flume

eds\_madhu chennai 67889 DataScience

cds\_jonny london 34567 Flume

bcds\_babu hyd 2345 Spark

abc\_raghu banglore 12345 BigData

bcd\_babu hyderabad 2345 Spark

cds\_jonny london 34567 Flume

eds\_madhu chennai 67889 DataScience

cds\_jonny london 34567 Flume

bcds\_babu hyd 2345 Spark

Time taken: 0.109 seconds, Fetched: 12 row(s)

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/jan25.db/students

Found 2 items

-rwxrwxrwx 1 cloudera supergroup 336 2021-01-26 02:00 /user/hive/warehouse/jan25.db/students/student.txt

-rwxrwxrwx 1 cloudera supergroup 336 2021-01-26 02:15 /user/hive/warehouse/jan25.db/students/student\_copy\_1.txt

[cloudera@quickstart ~]$

**Replace/Overwrite the data (don’t want to append the same data) command**

hive> load data local inpath '/home/cloudera/khasimbabu/HIVE\_Excercise/student.txt' overwrite into table students;

Loading data to table jan25.students

Table jan25.students stats: [numFiles=1, numRows=0, totalSize=336, rawDataSize=0]

OK

Time taken: 0.344 seconds

hive> select \* from students;

OK

abc\_raghu banglore 12345 BigData

bcd\_babu hyderabad 2345 Spark

cds\_jonny london 34567 Flume

eds\_madhu chennai 67889 DataScience

cds\_jonny london 34567 Flume

bcds\_babu hyd 2345 Spark

Time taken: 0.161 seconds, Fetched: 6 row(s)  
  
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/jan25.db/students

Found 1 items

-rwxrwxrwx 1 cloudera supergroup 336 2021-01-26 02:20 /user/hive/warehouse/jan25.db/students/student.txt  
  
**External Table**

**hive> create external table students\_ext**(

> name string,

> location string,

> phone int,

> course string)

> row format delimited fields terminated by ','

> location '/January';  
  
**[cloudera@quickstart ~]$ hdfs dfs -ls /**

Found 6 items

drwxr-xr-x - cloudera supergroup 0 2021-01-26 02:33 /January

drwxrwxrwx - hdfs supergroup 0 2015-11-18 02:57 /benchmarks

drwxr-xr-x - hbase supergroup 0 2021-01-25 08:31 /hbase

drwxrwxrwt - hdfs supergroup 0 2020-12-02 09:37 /tmp

drwxr-xr-x - hdfs supergroup 0 2020-12-02 09:37 /user

drwxr-xr-x - hdfs supergroup 0 2015-11-18 03:00 /var  
  
[cloudera@quickstart ~]$ cd /home/cloudera/khasimbabu/HIVE\_Excercise

[cloudera@quickstart HIVE\_Excercise]$ hdfs dfs -put student.txt /January

[cloudera@quickstart HIVE\_Excercise]$ hdfs dfs -ls /January

Found 1 items

-rw-r--r-- 1 cloudera supergroup 336 2021-01-26 02:39 /January/student.txt  
  
hive> select \* from students\_ext;

OK

abc\_raghu banglore 12345 BigData

bcd\_babu hyderabad 2345 Spark

cds\_jonny london 34567 Flume

eds\_madhu chennai 67889 DataScience

cds\_jonny london 34567 Flume

bcds\_babu hyd 2345 Spark

Time taken: 0.096 seconds, Fetched: 6 row(s)

**Managed Table & External Table**

Managed Table – Data will be stored in the user Hive directory folder structure. When you drop Managed Table, the table will be deleted and the data in the folder is also deleted.

Hive stores the data for these tables in a subdirectory under the directory defined by hive.metastore.warehouse.dir (e.g., /user/hive/warehouse)

External Table – Data will be stored in the user defined location folder structure. When you drop External Table, the table will be deleted but the data will not be deleted.

**Partition in Hive**

/user/hive/warehouse/retail/sales/jan.txt

/user/hive/warehouse/retail/sales/feb.txt

/user/hive/warehouse/retail/sales/mar.txt

Select \* from sales where month=’april’; -> Hive will not know where to check, It will look at all the files of sales folder n scan all the files, accordingly queries will be very slow.

* To overcome this Hive has Partition concept
* Create partitions based on month – Hive will look at the specific folder
* /user/hive/warehouse/retail/sales/JAN/jan.txt
* /user/hive/warehouse/retail/sales/FEB/feb.txt
* /user/hive/warehouse/retail/sales/MAR/mar.txt

Partition is not mandatory. It also depends on the Query.

Eg:- select \* from sales where country=’India’

Bcs current partition is based on the month but the query is based on country

Dynamic -> all the partition is defined by hive and load the data dynamically

Static -> You can define the partition and upload the data.

Partition Modes :

Strict Mode – Defined partition based on the specific query. Allow queries based on partition data.

Non Strict Mode -Hive will allow all the queries.

Bucketing = Clustering in Hive

00000003,06-05-2011,4002199,198.19,Gymnastics,Gymnastics Rings,Milwaukee,Wisconsin,credit

00000004,12-17-2011,4002613,098.81,Team Sports,Field Hockey,Nashville ,Tennessee,credit

00000005,02-14-2011,4007591,193.63,Outdoor Recreation,Camping,Chicago,Illinois,credit

00000006,10-28-2011,4002190,027.89,Puzzles,Jigsaw Puzzles,Charleston,South Carolina,credit

Select something where txnid = ‘00000005’

* Where txnid is unique. In case of unique searches/queries, we can’t do the Partitioning bcs its not possible with unique values.
* For this type of cases we use Bucketing
* I tell hive, I want 10 buckets based on txnid hashed algorithm-> Hive will create 10 buckets(folders), when you try to load the data it looks at the txnid value.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

hive> CREATE TABLE employees (

> name STRING,

> salary FLOAT,

> subordinates ARRAY<STRING>,

> deductions MAP<STRING, FLOAT>,

> address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>);

hive> CREATE TABLE employees\_new (

> name STRING,

> salary FLOAT,

> subordinates ARRAY<STRING>,

> deductions MAP<STRING, FLOAT>,

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

> )

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY '\001'

> COLLECTION ITEMS TERMINATED BY '\002'

> MAP KEYS TERMINATED BY '\003'

> LINES TERMINATED BY '\n'

> STORED AS TEXTFILE;

# The character \001 is the octal code for ^A.’

# The character \002 is the octal code for ^B.

# The character \003 is the octal code for ^C.

# Use '\t' for tab-delimited fields.

hive> CREATE DATABASE IF NOT EXISTS financials;  
  
hive> show databases;

OK

default

financials

jan25

lab

retail  
  
hive> SHOW DATABASES LIKE 'f.\*';

OK

Financials

hive> CREATE DATABASE financials\_new

> COMMENT 'Holds all financial tables'

> LOCATION '/January/warehouse';  
  
[cloudera@quickstart ~]$ hdfs dfs -ls /January

Found 2 items

-rw-r--r-- 1 cloudera supergroup 336 2021-01-26 02:39 /January/student.txt

drwxr-xr-x - cloudera supergroup 0 2021-01-29 17:48 /January/warehouse  
  
hive> describe database financials\_new;

OK

financials\_new Holds all financial tables hdfs://quickstart.cloudera:8020/January/warehouse cloudera USER

Time taken: 0.046 seconds, Fetched: 1 row(s)

**hive> describe database extended financials\_new;**

**OK**

**financials\_new Holds all financial tables hdfs://quickstart.cloudera:8020/January/warehouse cloudera USER**

**Time taken: 0.017 seconds, Fetched: 1 row(s)**

hive> CREATE DATABASE financials1

> COMMENT 'Holds all financial tables'

> WITH DBPROPERTIES ('creator' = 'Mark Test', 'date' = '2021-01-02');  
  
hive> describe database financials1;

OK

financials1 Holds all financial tables hdfs://quickstart.cloudera:8020/user/hive/warehouse/financials1.db cloudera USER

Time taken: 0.024 seconds, Fetched: 1 row(s)

hive> describe database extended financials1;

OK

financials1 Holds all financial tables hdfs://quickstart.cloudera:8020/user/hive/warehouse/financials1.db cloudera USER {date=2021-01-02, creator=Mark Test}

Time taken: 0.015 seconds, Fetched: 1 row(s)  
  
hive> ALTER DATABASE financials1 SET DBPROPERTIES ('edited-by' = 'Joe TEST');

OK

Time taken: 0.102 seconds

hive> describe database extended financials1;

OK

financials1 Holds all financial tables hdfs://quickstart.cloudera:8020/user/hive/warehouse/financials1.db cloudera USER {edited-by=Joe TEST, date=2021-01-02, creator=Mark Test}

Time taken: 0.02 seconds, Fetched: 1 row(s)  
  
**hive> create database mydb;**

**hive> CREATE TABLE IF NOT EXISTS mydb.employees (**

**> name STRING COMMENT 'Employee name',**

**> salary FLOAT COMMENT 'Employee salary',**

**> subordinates ARRAY<STRING> COMMENT 'Names of subordinates',**

**> deductions MAP<STRING, FLOAT>**

**> COMMENT 'Keys are deductions names, values are percentages',**

**> address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT>**

**> COMMENT 'Home address')**

**> COMMENT 'Description of the table'**

**> TBLPROPERTIES ('creator'='me', 'created\_at'='2021-01-02 10:00:00');**

TBLPROPERTIES can be used to express essential metadata about the database connection.

You can also copy the schema (but not the data) of an existing table:   
**CREATE TABLE IF NOT EXISTS mydb.employees2 LIKE mydb.employees;**  
hive> SHOW TABLES 'empl.\*';

hive> create external table stockdata1(

> name string,

> symbol string,

> ymd string,

> price\_open float,

> price\_high float,

> price\_close float,

> volume int)

> row format delimited fields terminated by ','

> location '/January/warehouse';

hive> load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/NYSE\_daily.txt' into table stockdata1;

[cloudera@quickstart ~]$ hdfs dfs -ls /January/warehouse

Found 1 items

-rwxr-xr-x 1 cloudera supergroup 3619 2021-01-30 06:40 /January/warehouse/NYSE\_daily.txt

hive> select \* from stockdata1 limit 5;

OK

NYSE CPO 2009-12-30 0.14 20 50 30 NULL NULL NULL NULL NULL NULL

NYSE CPO 2009-09-28 0.14 210 520 310 NULL NULL NULL NULL NULL NULL

NYSE CPO 2009-06-26 0.14 20 520 30 NULL NULL NULL NULL NULL NULL

NYSE CPO 2009-03-27 0.14 220 50 320 NULL NULL NULL NULL NULL NULL

NYSE CPO 2009-01-06 0.14 20 50 30 NULL NULL NULL NULL NULL NULL

Time taken: 0.277 seconds, Fetched: 5 row(s)

hive> set hive.cli.print.current.db=true;

hive (financials)>

hive (financials)> set hive.cli.print.current.db=false;

hive>

**hive> DROP DATABASE IF EXISTS financials CASCADE;**

* Using the RESTRICT keyword instead of CASCADE is equivalent to the default behavior, where existing tables must be dropped before dropping the database. When a database is dropped, its directory is also deleted.

hive> CREATE TABLE employees (

> name STRING,

> salary FLOAT,

> subordinates ARRAY<STRING>,

> deductions MAP<STRING, FLOAT>,

> address STRUCT<street:STRING, city:STRING, state:STRING, zip:INT> )

> PARTITIONED BY (country STRING, state STRING);

hive> DESCRIBE mydb.employees.salary;   
salary float Employee salary

**Executing Hive Queries from Files**

Hive can execute one or more queries that were saved to a file using the -f file argument. By convention, saved Hive query files use the .q or .hql extension.   
$ hive -f /path/to/file/withqueries.hql

If you are already inside the Hive shell you can use the SOURCE command to execute a script file. Here is an example:   
$ cat /path/to/file/withqueries.hql   
SELECT x.\* FROM src x;   
$ hive   
hive> source /path/to/file/withqueries.hql;

hive> SELECT xpath(\'**b1b2**\',\'//@id\') > FROM src LIMIT 1;   
[foo","bar]  
  
**Query Column Headers**  
hive> set hive.cli.print.header=true;

**Partitioned & Bucketing**

hive (financials)> create table employee(

> first\_name string,

> last\_name string,

> id string)

> **partitioned by (country string, region string)**

hive (financials)> describe employee;

OK

first\_name string

last\_name string

id string

country string

region string

# Partition Information

# col\_name data\_type comment

country string

region string

Time taken: 0.083 seconds, Fetched: 11 row(s)

**Static Partition – (use the user\_info3.txt file)**File(employee.txt) has First\_Name, Last\_Name, Id columns. but we mention the **Partition** while creating the table.  
  
**In static partition**, create table don’t have the Partitioned columns. When we load the data then the columns will be added to the table.  
In Static Partition, we need to specify the partition parameters for each table records. For example when we load file1 data into table 1 with certain partition values & when we load the file2 data into table 1(same table) and specify the different partition values….go on we need to specify different partition values each time when you load the data into same table…..

1. Create an **employee** table with **First\_Name, Last\_Name, Id** columns & **partitioned by (country string, region string)**

**hive> create table employee(**

**> first\_name string,**

**> last\_name string,**

**> id string)**

**> partitioned by(country string, region string)**

**> row format delimited fields terminated by ',';**

1. Load the data into the table.   
   -> Load the **employee.txt** file data into **employee** table & used **partition(country = 'Suriname', region = 'Berlin')  
   ->** This means All the records of the files will add 2 new columns(country and region).

hive> **load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/employee.txt'**

**> into table employee**

**> partition(country = 'Suriname', region = 'Berlin');**

Loading data to table financials.employee partition (country=Suriname, region=Berlin)

Partition financials.employee{country=Suriname, region=Berlin} stats: [numFiles=1, numRows=0, totalSize=1760, rawDataSize=0]

cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/financials.db/employee/country=Suriname

Found 1 items

drwxrwxrwx - cloudera supergroup 0 2021-01-31 09:04 /user/hive/warehouse/financials.db/employee/country=Suriname/region=Berlin

[cloudera@quickstart ~]$

hive> select \* from employee limit 5;

OK

employee.first\_name employee.last\_name employee.id employee.country employee.region

Aidan Estes D765BFC2-1B84-3EA6-F8E1-D05B9F2502DD Suriname Berlin

Aaron Bailey 922B5A56-5A98-2FDB-5B18-ED17E1D1D370 Suriname Berlin

Keelie Jefferson 5A27FEBE-0526-BA14-0A23-661225C098E1 Suriname Berlin

Quynn Mosley FE4BE86B-2736-1267-F6E8-7CB6395B95CA Suriname Berlin

Deirdre Summers D5470391-CDCF-CC18-5CD3-5C2A4766350A Suriname Berlin

1. Load the file2 data into the same employee table using different partition.  
   -> Load the **employee.txt** file data into **employee** table & used **partition(country =** Austria**, region = '**Liguria**')  
   ->** This means All the records of the files will add 2 new columns(country and region).

hive> **load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/user\_info2.txt'**

**> into table employee**

**> partition(country='Austria', region='Liguria');**

Loading data to table financials.employee partition (country=Austria, region=Liguria)

Partition financials.employee{country=Austria, region=Liguria} stats: [numFiles=1, numRows=0, totalSize=572, rawDataSize=0]

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/financials.db/employee/

Found 2 items

drwxrwxrwx - cloudera supergroup 0 2021-01-31 09:38 /user/hive/warehouse/financials.db/employee/**country=Austria**

drwxrwxrwx - cloudera supergroup 0 2021-01-31 09:35 /user/hive/warehouse/financials.db/employee/**country=Suriname**

[cloudera@quickstart ~]$

hive> select \* from employee;

OK

employee.first\_name employee.last\_name employee.id employee.country employee.region

Regan Cole 07FB56A3-5157-A29E-DB5A-66AE79B88CD2 Austria Liguria

Wyoming Ortega D5EDF905-A08E-400D-63B3-0654DBCCA31C Austria Liguria

Forrest Byers F5722E36-AF4A-1185-8793-88D25AF51749 Austria Liguria

Kelsie Singleton 57CF5C7D-CB59-00E2-196E-3DCD58D24734 Austria Liguria

Emerson Bartlett 34B0BE13-F31E-4A9C-9C17-777F32961765 Austria Liguria

Aidan Estes D765BFC2-1B84-3EA6-F8E1-D05B9F2502DD Austria Liguria

Aaron Bailey 922B5A56-5A98-2FDB-5B18-ED17E1D1D370 Austria Liguria

Keelie Jefferson 5A27FEBE-0526-BA14-0A23-661225C098E1 Austria Liguria

Quynn Mosley FE4BE86B-2736-1267-F6E8-7CB6395B95CA Austria Liguria

Deirdre Summers D5470391-CDCF-CC18-5CD3-5C2A4766350A Austria Liguria

Tanek Long 0EF1EA89-7C0C-58E1-855C-C38D53F88DA0 Austria Liguria

Aidan Estes D765BFC2-1B84-3EA6-F8E1-D05B9F2502DD Suriname Berlin

Aaron Bailey 922B5A56-5A98-2FDB-5B18-ED17E1D1D370 Suriname Berlin

Keelie Jefferson 5A27FEBE-0526-BA14-0A23-661225C098E1 Suriname Berlin

Quynn Mosley FE4BE86B-2736-1267-F6E8-7CB6395B95CA Suriname Berlin

Deirdre Summers D5470391-CDCF-CC18-5CD3-5C2A4766350A Suriname Berlin

Tanek Long 0EF1EA89-7C0C-58E1-855C-C38D53F88DA0 Suriname Berlin

**Dynamic Partition – (use the user\_info2.txt file – copy and rename it employee2)**

File(employee2) has First\_Name, Last\_Name, Id columns but we mention the **Partion** while creating the table.  
**In Dynamic partition**, create table(employee2) don’t have the Partioned columns.

We need to create another table(employee3), which has First\_Name, Last\_Name, Id, country, region columns.

We need to load the data into table(employee3), which has all the columns.

Finally, we need to load the data from table(employee3) into table(employee2) using insert command, we need to specify the partition parameters for table(employee2).  
This is onetime insertion, which will create multiple partitions dynamically.

Hive>set hive.exec.dynamic.partition=true;

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

Hive>set hive.exec.max.dynamic.partitions=100;

Hive>set hive.exec.max.dynamic.partitions.pernode=100;

1. Create a **table(employee2) with partition columns.**

hive> create table employee2(

> first\_name string,

> last\_name string,

> id string)

> partitioned by (country string, region string)

> row format delimited fields terminated by ',';

1. Create a **table(employee3) with all required columns.**

hive> create table employee3(

> first\_name string,

> last\_name string,

> id string,

> country string,

> region string)

> row format delimited fields terminated by ',';

1. **Load the data into table(employee3).**

hive> load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/employee3'

> into table employee3;

hive> select \* from employee3 limit 5;

OK

employee3.first\_name employee3.last\_name employee3.id employee3.country employee3.region

Aidan Estes D765BFC2-1B84-3EA6-F8E1-D05B9F2502DD Christmas Island SL

Aaron Bailey 922B5A56-5A98-2FDB-5B18-ED17E1D1D370 Nicaragua São Paulo

Keelie Jefferson 5A27FEBE-0526-BA14-0A23-661225C098E1 Liberia WA

Quynn Mosley FE4BE86B-2736-1267-F6E8-7CB6395B95CA Kiribati C

Deirdre Summers D5470391-CDCF-CC18-5CD3-5C2A4766350A Saint Kitts and Nevis ERM

1. **Load the data into table(employee2) using insert command. It will create all the partitions dynamically.**

hive> insert into table employee2 partition(country,region)

> select first\_name,last\_name,id,country,region from employee3;

hive> select \* from employee2 limit 5;

OK

employee2.first\_name employee2.last\_name employee2.id employee2.country employee2.region

Bertha Hopper 0D4DA087-4C99-7789-651F-44E3958C6E2F "Korea North"

Cyrus Landry 6D6D8383-A36A-9235-7F29-349112F10396 Angola SJ

Lane Wolfe 6591E5E1-24EF-6A51-9423-8EAAA0801780 Austria Liguria

Keelie Nicholson 86E878D9-B1D5-4618-D60C-146923A75776 Belize Idaho

Audra Stephenson 1A6F336D-6E80-36C5-CC4A-F40BF1FF4236 Burundi AL

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/financials.db/employee3

Found 1 items

-rwxrwxrwx 1 cloudera supergroup 1760 2021-01-31 17:42 /user/hive/warehouse/financials.db/employee3/employee3

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/financials.db/employee2

Found 25 items

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=%22Korea

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=Angola

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=Austria

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=Belize

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=Burundi

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48

[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/financials.db/employee2/country='Austria'

Found 1 items

drwxrwxrwx - cloudera supergroup 0 2021-01-31 17:48 /user/hive/warehouse/financials.db/employee2/country=Austria/region=Liguria

[cloudera@quickstart ~]$

hive> **show partitions employee;**

OK

partition

country=Austria/region=Liguria

country=Suriname/region=Berlin

Time taken: 0.042 seconds, Fetched: 2 row(s)

hive> **show partitions employee2;**

OK

partition

country=%22Korea/region= North%22

country=Angola/region=SJ

country=Austria/region=Liguria

country=Belize/region=Idaho

country=Burundi/region=AL

country=Christmas Island/region=SL

country=Gabon/region=Denbighshire

country=Guam/region=Diy

country=Kazakhstan/region=Uttarakhand

country=Kiribati/region=C

country=Latvia/region=SO

country=Liberia/region=WA

country=Luxembourg/region=L

country=Myanmar/region=SJ

country=Namibia/region=ON

country=Nicaragua/region=S�o Paulo

country=Nigeria/region=Pomorskie

country=Oman/region=SJ

country=Saint Kitts and Nevis/region=ERM

country=Suriname/region=Berlin

country=Svalbard and Jan Mayen Islands/region=TX

country=Tanzania/region=Van

country=Togo/region=West Bengal

country=Venezuela/region=OG

country=Venezuela/region=Overijssel

country=Viet Nam/region=British Columbia

Time taken: 0.048 seconds, Fetched: 26 row(s)

hive> **show partitions employee3;**

FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. Table employee3 is not a partitioned table

hive> **show partitions employee2 partition(country='Venezuela');**

OK

partition

country=Venezuela/region=OG

country=Venezuela/region=Overijssel

Time taken: 0.086 seconds, Fetched: 2 row(s)

**External Partitioned Tables**

1. Create external table

hive> **create external table employee\_extn(**

> first\_name string,

> second\_name string,

> id string)

> partitioned by (country string, region string)

> row format delimited fields terminated by ','

> location '/January/EmployeeExt';

1. Load data into external table.

hive> **load data local inpath** 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/user\_info1.txt'

> into table employee\_extn

> partition(country='Austria', region='Liguria');

Loading data to table financials.employee\_extn partition (country=Austria, region=Liguria)

Partition financials.employee\_extn{country=Austria, region=Liguria} stats: [numFiles=1, numRows=0, totalSize=526, rawDataSize=0]

[cloudera@quickstart ~]$ hdfs dfs -ls /January/EmployeeExt

Found 1 items

drwxr-xr-x - cloudera supergroup 0 2021-02-01 02:35 /January/EmployeeExt/country=Austria

[cloudera@quickstart ~]$ hdfs dfs -ls /January/EmployeeExt/country=Austria;

Found 1 items

drwxr-xr-x - cloudera supergroup 0 2021-02-01 02:35 /January/EmployeeExt/country=Austria/region=Liguria

[cloudera@quickstart ~]$

**Creating Tables and Loading Them in One Query**

CREATE TABLE ca\_employees AS SELECT name, salary, address FROM employees WHERE se.state = 'CA';

**Computing with column values**

hive> SELECT upper(name), salary, deductions["Federal Taxes"], > round(salary \* (1 - deductions["Federal Taxes"])) FROM employees;

hive> SELECT count(\*), avg(salary) FROM employees;  
hive> SELECT count(DISTINCT ymd), count(DISTINCT volume) FROM stocks;

**SELECT parse\_url\_tuple(url, 'HOST', 'PATH', 'QUERY') as (host, path, query) FROM url\_table;**

**CASE … WHEN … THEN Statements**hive> create table employeedata(

> first\_name string,

> last\_name string,

> id string,

> salary double)

> row format delimited fields terminated by ',';

OK

Time taken: 0.053 seconds

hive> load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/employeedata.txt' into table employeedata;

Loading data to table financials.employeedata

Table financials.employeedata stats: [numFiles=1, totalSize=631]

hive> **select first\_name,salary,**

**> CASE**

**> when salary<5000 then 'low'**

**> when salary>=5000 and salary<7000 then 'middle'**

**> when salary>=7000 and salary<50000 then 'high'**

**> else 'very low'**

**> end as brackets from employeedata ;**

OK

first\_name salary brackets

Regan 5000.0 middle

Wyoming 6000.0 middle

Forrest 10000.0 high

Kelsie 3000.0 low

Emerson 2000.0 low

Aidan 15000.0 high

Aaron 1000.0 low

Keelie 6500.0 middle

Quynn 7000.0 high

Deirdre 4000.0 low

Tanek 25000.0 high

Time taken: 0.203 seconds, Fetched: 11 row(s)

hive> SELECT year(ymd), avg(price\_close) FROM stocks   
> WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'   
> GROUP BY year(ymd);

hive> SELECT year(ymd), avg(price\_close) FROM stocks

> WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'

> GROUP BY year(ymd)

> HAVING avg(price\_close) > 50.0;

hive> SELECT s2.year, s2.avg FROM

> (SELECT year(ymd) AS year, avg(price\_close) AS avg FROM stocks

> WHERE exchange = 'NASDAQ' AND symbol = 'AAPL'

> GROUP BY year(ymd)) s2

> WHERE s2.avg > 50.0;

**JOIN Statements**  
**Inner JOIN:**

hive> create table nyse\_dividend(

> exchangename string, symbol string, datetime string, price float)

> row format delimited fields terminated by '/t';

hive> load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/NYSE\_dividends.txt' into table nyse\_dividend;

hive> select \* from nyse\_dividend limit 5;

OK

nyse\_dividend.exchangename nyse\_dividend.symbol nyse\_dividend.datetime nyse\_dividend.price

NYSE CPO 2009-12-30 0.14 NULL NULL NULL

NYSE CPO 2009-09-28 0.14 NULL NULL NULL

NYSE CPO 2009-06-26 0.14 NULL NULL NULL

NYSE CPO 2009-03-27 0.14 NULL NULL NULL

NYSE CPO 2009-01-06 0.14 NULL NULL NULL

hive> create table nyse\_stock(

> exchangetype string, symbol string, price float, price\_low float, price\_high float, price\_close float)

> row format delimited fields terminated by '/t';

hive> load data local inpath 'file:///home/cloudera/khasimbabu/HIVE\_Excercise/NYSE\_daily.txt' into table nyse\_stock;

hive> select \* from nyse\_stock limit 5;

OK

nyse\_stock.exchangetype nyse\_stock.symbol nyse\_stock.price nyse\_stock.price\_low nyse\_stock.price\_high nyse\_stock.price\_close

NYSE CPO 2009-12-30 0.14 20 50 30 NULL NULL NULL NULL NULL

NYSE CPO 2009-09-28 0.14 210 520 310 NULL NULL NULL NULL NULL

NYSE CPO 2009-06-26 0.14 20 520 30 NULL NULL NULL NULL NULL

NYSE CPO 2009-03-27 0.14 220 50 320 NULL NULL NULL NULL NULL

NYSE CPO 2009-01-06 0.14 20 50 30 NULL NULL NULL NULL NULL

Time taken: 0.051 seconds, Fetched: 5 row(s)

hive> SELECT a.ymd, a.price\_close, b.price\_close > FROM stocks a JOIN stocks b ON a.ymd = b.ymd > WHERE a.symbol = 'AAPL' AND b.symbol = 'IBM';

hive> SELECT s.ymd, s.symbol, s.price\_close, d.dividend > FROM stocks s JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol > WHERE s.symbol = 'AAPL';

SELECT s.ymd, s.symbol, s.price\_close, d.dividend FROM stocks s JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol WHERE s.symbol = 'AAPL';

L**EFT OUTER JOIN**   
The left-outer join is indicated by adding the LEFT OUTER keywords:   
hive> SELECT s.ymd, s.symbol, s.price\_close, d.dividend   
> FROM stocks s **LEFT OUTER JOIN** dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol   
> WHERE s.symbol = 'AAPL';

**RIGHT OUTER JOIN**

hive> SELECT s.ymd, s.symbol, s.price\_close, d.dividend > FROM dividends d **RIGHT OUTER JOIN** stocks s ON d.ymd = s.ymd AND d.symbol = s.symbol > WHERE s.symbol = 'AAPL';

**FULL OUTER JOIN**

hive> SELECT s.ymd, s.symbol, s.price\_close, d.dividend > FROM dividends d **FULL OUTER JOIN** stocks s ON d.ymd = s.ymd AND d.symbol = s.symbol > WHERE s.symbol = 'AAPL';

**LEFT SEMI-JOIN**  
SELECT s.ymd, s.symbol, s.price\_close FROM stocks s WHERE s.ymd, s.symbol IN (SELECT d.ymd, d.symbol FROM dividends d);

hive> SELECT s.ymd, s.symbol, s.price\_close > FROM stocks s LEFT SEMI JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol;

**Cartesian Product JOINs**  
SELECTS \* FROM stocks JOIN dividends;  
  
hive > SELECT \* FROM stocks JOIN dividends > WHERE stock.symbol = dividends.symbol and stock.symbol='AAPL';

**ORDER BY and SORT BY**  
hive> select exchangename,symbol,price from nyse\_dividend ORDER BY price ASC;  
hive> select exchangetype,symbol,price,price\_close from nyse\_stock **ORDER BY** price\_close ASC, symbol DESC;

hive> select exchangetype,symbol,price,price\_close from nyse\_stock **SORT BY** price\_close DESC, symbol ASC;

**DISTRIBUTE BY**

We can use DISTRIBUTE BY to ensure that the records for each stock symbol go to the same reducer, then use SORT BY to order the data the way we want.

hive> SELECT s.ymd, s.symbol, s.price\_close

> FROM stocks s

> DISTRIBUTE BY s.symbol

> SORT BY s.symbol ASC, s.ymd ASC;

**CLUSTER BY**

hive> SELECT s.ymd, s.symbol, s.price\_close

> FROM stocks s

> CLUSTER BY s.symbol;

**Casting**  
SELECT name, salary FROM employees WHERE **cast(salary AS FLOAT)** < 100000.0;

**Bucketing**

hive> CREATE TABLE numbers\_bucketed (number int) CLUSTERED BY (number) INTO 3 BUCKETS; hive> SET hive.enforce.bucketing=true; hive> INSERT OVERWRITE TABLE numbers\_bucketed SELECT number FROM numbers;

**UNION ALL**   
UNION ALL combines two or more tables.

SELECT log.ymd, log.level, log.message FROM ( SELECT l1.ymd, l1.level, l1.message, 'Log1' AS source FROM log1 l1 UNION ALL SELECT l2.ymd, l2.level, l2.message, 'Log2' AS source FROM log1 l2 ) log SORT BY log.ymd ASC;

**View**  
A view allows a query to be saved and treated like a table.  
When a query becomes long or complicated, a view may be used to hide the complexity by dividing the query into smaller, more manageable pieces; similar to writing a function in a programming language.  
  
FROM ( SELECT \* FROM people JOIN cart ON (cart.people\_id=people.id) WHERE firstname='john' ) a SELECT a.lastname WHERE a.id=3;

CREATE VIEW shorter\_join AS SELECT \* FROM people JOIN cart ON (cart.people\_id=people.id) WHERE firstname='john';

SELECT lastname FROM shorter\_join WHERE id=3;  
  
hive> CREATE TABLE employee (firstname string, lastname string, > ssn string, password string, department string);   
hive> CREATE VIEW techops\_employee AS > SELECT firstname,lastname,ssn FROM userinfo WERE department='techops';

CREATE TABLE shipments2 LIKE shipments;

DROP VIEW IF EXISTS shipments;

ALTER VIEW shipments SET TBLPROPERTIES ('created\_at' = 'some\_timestamp');

**Indexes**

Hive has limited indexing capabilities. There are no keys in the usual relational database sense, but you can build an index on columns to speed some operations. The index data for a table is stored in another table.

CREATE TABLE employees ( name STRING, salary FLOAT, subordinates ARRAY, deductions MAP, address STRUCT ) PARTITIONED BY (country STRING, state STRING);  
  
CREATE INDEX employees\_index ON TABLE employees (country) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD IDXPROPERTIES ('creator = 'me', 'created\_at' = 'some\_time') IN TABLE employees\_index\_table PARTITIONED BY (country, name) COMMENT 'Employees indexed by country and name.';

**Bucketing Table Data Storage**

Partitions offer a convenient way to segregate data and to optimize queries. However, not all data sets lead to sensible partitioning, especially given the concerns raised earlier about appropriate sizing.

Bucketing is another technique for decomposing data sets into more manageable parts.

hive> CREATE TABLE weblog (url STRING, source\_ip STRING)

PARTITIONED BY (dt STRING, user\_id INT);

hive> FROM raw\_weblog   
 INSERT OVERWRITE TABLE page\_view PARTITION(dt='2012-06-08', user\_id)

SELECT server\_name, url, source\_ip, dt, user\_id;

hive> CREATE TABLE weblog (user\_id INT, url STRING, source\_ip STRING)

PARTITIONED BY (dt STRING)

CLUSTERED BY (user\_id) INTO 96 BUCKETS;

hive> SET hive.enforce.bucketing = true;

hive> FROM raw\_logs

INSERT OVERWRITE TABLE weblog

PARTITION (dt='2009-02-25')

SELECT user\_id, url, source\_ip WHERE dt='2009-02-25';

**Adding Columns to a Table**   
Hive allows the definition of a schema over raw data files, unlike many databases that force the conversion and importation of data following a specific format. A benefit of this separation of concerns is the ability to adapt a table definition easily when new columns are added to the data files.

hive> CREATE TABLE weblogs (version LONG, url STRING)   
 PARTITIONED BY (hit\_date int)   
 ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';   
  
hive> ! cat log1.txt   
1 /mystuff   
1 /toys   
  
hive> LOAD DATA LOCAL INPATH 'log1.txt' int weblogs partition(20110101);   
hive> SELECT \* FROM weblogs;   
1 /mystuff 20110101   
1 /toys 20110101

hive> ! cat log2.txt   
2 /cars bob   
2 /stuff terry   
  
hive> ALTER TABLE weblogs ADD COLUMNS (user\_id string);   
hive> LOAD DATA LOCAL INPATH 'log2.txt' int weblogs partition(20110102);   
hive> SELECT \* from weblogs   
1 /mystuff 20110101 NULL   
1 /toys 20110101 NULL   
2 /cars 20110102 bob   
2 /stuff 20110102 terry