HIVE (HQL)

SESSION 1

```
show databases;
create database trendy;
use trendy;

create table customers (
id bigint,name string,address string);
show tables;
describe customers;

describe formatted customers -- Describe table with extra information

insert into customers values (1111,"John","WA");
insert into customers values (2222,"emily","MF");
insert into customers values (3333,"damon","CA");
insert into customers values (4444,"stefan","LV");

select * from customers;

hadoop fs -ls /user/hive/warehouse == Actual Data = HDFS
hadoop fs -ls /user/hive/warehouse/trendy.db/customers/files
hadoop fs -ls /user/hive/warehouse/trendy.db/customers/*
hadoop fs -cat /user/hive/warehouse/trendy.db/customers/000000_0

files = Inside these customers your data files will be created
No of files = No of times you insert the data.
```

There is no default database in HDFS.If you create orders table inside default database then it will be directly created inside warehouse.
/user/hive/warehouse/orders

Only cat is used when you want to see single file
cat /* is used when there are multiple files inside one particular directory.

SESSION 2

select * from customers where address = "WA";

select distinct address from customer;

select name, address from customers order by address;

select count(*) from customers;

select address ,count(*) from customers group by address;

This all will trigger Mapreduce job.

To come out of hive = exit

CREATE TABLE WITH IF NOT EXIST STATEMENT

create table if not exists orders (

id bigint,

product_id string,

customer_id string,

quantity int,

amount double);

CONNECTING TO BEELINE

beeline -u jdbc:hive2://

To come out of beeline = !q

RUNNING THE QUERY WHEN YOU ARE NOT CONNECTED TO BEELINE

beeline -u jdbc:hive2:// -e "select * from trendy.customers"

CRATING BEELINE SCRIPT FILE

create a file myquries .hql using gedit terminal

COMMAND = gedit myquries.hql

It will now open a gedit terminal

Inside this terminal write the script and save it

cat myquries.hql

EXECUTE THE FILE FROM TERMINAL

beeline -u jdbc:hive2:// -f /home/cloudera/myquries.hql

EXECUTE THE BEELINE SCRIPT FEOM BEELINE ITSELF

source /home/cloudera/myquries.hql;

SESSION 3

METADATA LOCATION

Mysql-->show databases-->metastore ====== Here your metadata will be stored.

use metastore;

Here in Metastore database there will be TBLS Table

Inside TBLS Table all the schema of tables will be stored which we created in HIVE.

TYPES OF TABLES

1. MANAGED TABLE

To see where data is stored in HDFS from HIVE

dfs -ls /user/hive/warehouse/trendy.db;

To see where hive data is stored in HDFS from HDFS

hadoop fs -ls /user/hive/warehouse/trendy.db

To see the type of table in HIVE use command

describe formatted orders;

When we create table in HIVE

hadoop fs -ls /user/hive/warehouse == Actual Data = HDFS

Mysql-->show databases-->metastore-->use metastore-->Inside TBLS-->You will schema of the table created in HIVE

Now, in case of Managed Table when you drop the Managed Table your both Data and Metadata will be deleted.

2. EXTERNAL TABLE

```
create external table products5 (
id string,
title string,
cost float)
location '/data/';
```

We give path where actually data resides.

First put the file in /data location.

It will put all the data in first column because we did not give any delimiter.

if you, drop table products4

-- Your data will be intact in HDFS.

-- Only Metadata will get deleted.

```
create external table products8 (
id string,
title string,
cost float)
row format delimited
fields terminated by ','
location '/data/';
```

Now, in case of External Table when you drop the External Table Only Metadata will be deleted and Data will be intact in HDFS.

SESSION 4 - LOADING DATA FROM FILES

FROM LOCAL TO HIVE TABLE

1. Create a managed Table

```
create table if not exists products_managed(id string,title string,cost float)
row format delimited
fields terminated by ','
stored as textfile;
```

2. File should be available in your local enviornment.

/home/cloudera/Desktop/shared1/products.csv

3.Load data into Managed table from a local path

```
load data local inpath '/home/cloudera/Desktop/shared1/products.csv' into
table products_managed;
```

Run this command in HIVE.

Data should have been loaded in products_managed.

4.The table we created is a managed table

We have not specified the path of data. That means data will be kept in default path (/user/hive/warehouse/trendy.db)

so loading from local, products.csv is copied from local to hdfs (/user/hive/warehouse/trendy.db)

Loading data from local to HIVE table is copy paste operation.

LOAD DATA FROM HDFS TO HIVE TABLE

1.First copy the file from local to HDFS

hadoop fs -copyFromLocal /home/cloudera/Desktop/shared1/products.csv /data/

2.Load data into managed Table from HDFS.

load data inpath '/data/products.csv' into table products_managed;

Loading data from HDFS to HIVE table is copy paste operation.

OVERWRITE A HIVE TABLE

load data local inpath '/home/cloudera/Desktop/shared1/products.csv' overwrite into table products8;

By default, it will append the data into table if you do not provide overwrite command.

DATA LOADING USING TABLE TO TABLE METHOD

insert into table products_managed2

select * from products_managed;

products_managed2 = New empty table with same schema

products_managed = Table containing data.

SUBQURIES IN FROM CLAUSE

```
create external table products8 (
id string,
title string,
cost float)
row format delimited
fields terminated by ',';

load data local inpath '/home/cloudera/Desktop/shared1/products.csv' into
table products;

create external table products5 (
id string,
title string,
cost float)
row format delimited
fields terminated by ',';

load data local inpath '/home/cloudera/Desktop/shared1/freshproducts.csv'
into table freshproducts;

select * FROM (
 select id as product_id from products
UNION ALL
select id as product_id from freshproducts) t;
```

SUBQURIES IN WHERE CLAUSE

IN / NOT IN

```
select name from customers
WHERE customer.id IN
( select customer_id from orders);
```

```
select name from customers
WHERE customer.id  NOT IN
( select customer_id from orders);
```

EXISTS / NOT EXISTS

```
select id from customers WHERE EXISTS
(select sutomer_id from orders
where orders.customer_id = customers.id);
```

```
select id from customers WHERE NOT EXISTS
(select sutomer_id from orders
where orders.customer_id = customers.id);
```

VIEWS IN HIVE

```
CREATE VIEW customer_purchases
AS
SELECT customer_id,product_id,address
FROM customers
JOIN orders
WHERE customers.id = orders.customer_id;

show tables;

dscribe fromatted customer_purchases;

select * from customer_purchases;
```

HIVE EXPLODE

```
create table author_details(
author_name string,
book_names array<string>)
row format delimited
fields terminated by ','
collection items terminated by ":" ;
```

Collection items terminated by ":" == Means Array items are separated by : in the file.

```
load data local inpath '/home/cloudera/Desktop/shared1/authors.csv' into table author_details;
```

Example -

```
select * from author_details;
```

```
author_name  | book_names
Salman Rushide        | ["Grimus","Shame","Fury"]
Thomas Otway          | ["Don Carlos","The Orphan"]
Ben Jonson            | ["Volpone","Epicene"]
John Milton           | ["Arcades","Comus"]
```

Now if you want list of books in each line then we need to use explode column.

select explode(book_names) from author_details;

Ouput -

col

Grimus

Shame

Fury

Don Carlos

The Orphan

Volpone

Epicene

Arcades

Comus

The limitation with this is we can only see it as shown above

Example -

author1, [book1,book2,book3]

book1

book2

book3

We can't do it like below with explode command.

author1 book1

author1 book2

author1 book3

LATERAL VIEW

the exploded column should be made as a virtual table (lateral view) and this is joined with the actual table.

select author_name, b_name from author_details lateral view explode (book_names) book_table as b_name;

table1 == author_details
join == Lateral view
table2 == explode(book_names)
alias name of table 2 is book_table
b_name is the column name in virtual table (book_table)

COMPLEX DATATYPES IN HIVE

ARRAY

```
create table mobilephones ( id string,
title string,
cost float,
colors array<string>,
screen_size array<float>
);

insert into table mobilephones
select "redminote7", "Redmi Note 7", 300, array ("white", "silver",
"black"),array(float(4.5))
UNION ALL
select "motoGplus", "Moto G Plus", 200, array ("black", "gold"),
array (float (4.5), float (5.5));

select id,color[0] from mobilephones;

create table mobilephones_new (
id string,
title string,
cost float,
colors array<string>,
screen_size array<float>
)
row format delimited
fields terminated by ','
collection items terminated by '#';
```

load data local inpath '/home/cloudera/Desktop/shared1/mobilephones.csv' into table mobilephones_new;

MAP

create table mobilephones (

id string,

title string,

cost float,

colors array<string>,

screen_size array<float>,

features map<string, boolean>

)

row format delimited

fields terminated by ','

collection items terminated by '#'

map keys terminated by ':';

Example of Map –

features map<string, boolean>

String Boolean

Key Value

Camera True

Dualsim False

load data local inpath '/home/cloudera/Desktop/shared1/mobilephones.csv' into table mobilephones;

STRUCT

```
create table mobilephones (
id string,
title string,
cost float,
colors array<string>,
screen_size array<float>,
features map<string, boolean>,
information struct<battery: string, camera:string>
)
row format delimited
fields terminated by ','
collection items terminated by '#'
map keys terminated by ':';

load data local inpath '/home/cloudera/Desktop/shared1/mobilephones.csv'
into table mobilephones;

select id,features['camera'],information.battery from mobilephones;
```

UDF (USER DEFINED FUNCTION)

https://www.linkedin.com/pulse/hive-udfs-amanjit-singh/

To Download the Jars

https://mvnrepository.com/artifact/org.apache.hive/hive-exec/1.2.2

UDF JAVA CODE

```
package udf_example;

import org.apache.hadoop.hive.ql.exec.UDF;
public class DataStandardization extends UDF {

public String evaluate (String input) {

if(input==null)
{
return null;
}
return (input.toUpperCase());
}
}
```

```
create table sample_table (name string,count int)
row format delimited
fields terminated by ','
collection items terminated by '\n';

load data local inpath '/home/cloudera/Desktop/shared1/sample_data.txt'
into table sample_table;

ADD JAR /home/cloudera/Desktop/udf_example.jar;

create temporary function standardize as
'udf_example.DataStandardization';

select standardize(name) from sample_table;

hadoop fs -put /home/cloudera/Desktop/udf_example.jar /user/cloudera

CREATE FUNCTION standardize_permanent AS
'udf_example.DataStandardization' using JAR
'hdfs://localhost:8020/user/cloudera/udf_example.jar';
```