Core Java+SQL --> MR 1.x 2.x
--> HDFS basic commands -> Pig -> Pig Basics -> PigAdvance --> Hive Basics -> Hive Advance --> Sqoop ->Sqoop Adv --> Flume --> HBase --> Oozie --> Project --> Spark --> Scala --> Kafka ---> Hadoop Admin --> All above installations.. Standalone or single node or

---

Java - OOP
Abstraction - Hiding of data..Cars, Remote, Laptop, Mobile, TV
Encapsulation - Mechanism to binds together code, parts to keep it safe. Its a wrapping of data. Ex: capsule, To achieve the abstratction we do encapsulation.
Polymorphism - One interface and multi methods/behaviours. Static and Dynamic.

Overloading and Overriding..

Shape --> area()
--> Traingle --> area (b*h)
--> Circle --> area (pi*r*r)

Inheritance - Acquires a properties from one obj to another..Reusability..

Hadoop revise

1.x and 2.x (Yarn - yet another resource negotiator)

4k nodes, Integrate other frameworks...

Yarn - has cluster management capabilities and map reduce.

Job tracker is divided into three services:
Resource manager- Persistent YARN service that receives and run applications on cluster.
JobHistoryServer- To provide information about jobs and status.
Application Master- To manage each MR job and terminates when job completes.

TaskTracker has been replaced with NodeManager, that manages resources and deployment on node. Also responsible for launching container that could be a map or reduce task.

Hadoop - Open source framework
Hdfs - Storage - Hadoop Distributed File System
Processing engine - MapReduce

HDFS commands
MR program - wordcount
PIG- ETL
HIVE- DW
SQOOP-Import/Export
OOZIE- Job Scheduler to automate jobs
HBASE- NoSQL Database

HDFS basic commands:

Windows - Folder
Linux - Directory

File system: Root /
Shell - Communicate with hardware..
hdfs - inherited from unix root /

ls -List
ls -lrta - listing with attributes for all files/directories
cp - Copy files/directory
pwd- Present working directory
cd - Change directory
sudo - Super user do
chmod - Change mode of a file/directory 541 (u,g,o)7 -
(4(read)+2(write)+1(exec))
mkdir - Make new directory
rm - Remove a file
rm -r -> Remove directory with recursion
mv - Move the file/directory
cat - Catout the containts of a file
touch - Creating an empty file
gedit - editor top open a file for writing
clear - Clear the screen (Ctrl+L)

HDFS commands:

hadoop fs -ls /

18-12-2018

Installation CDH, Shared Folder, Virtual box

1. Virtual Box
2. Merging
3. Unzip
4. Click on .ovf file and import .vmdk file.
5. Save the machine and Close the virtual box
6. Open virtual box and click on Saved machine..

Shared folder:

1. Create a folder named SharedFolder in main OS anywhere..
2. Open virtual box and make sure ur CDH is poweredOFF
3. Goto setting-> SharedFolder Option --> Click + sign -->Folder path-
>Select ur folder-->Choose--> Enable AutoMount->OK
4. Start your CDH now.
5. You must see drive of shared folder on CDH desktop
6. Now Open a terminal > sudo gedit /etc/group
Add cloudera at last after vboxsf:474:
Save the file
7. Termianal > sudo shutdown -r now
8. Make Clipboard bidirectional:
On top Vbox -> Devices --> SharedClipboard--> Enable Bidirectional Option

---

HDFS - storage
MR - Processing Engine

Java can write plenty of code...

Tools -
Apache Pig - ETL (Extract tranform and load) - Pig Latin -
MR Program - I S M S R O (Java code)
Apache Hive - DW (Data warehousing) - Hive QL - OLAP - History
Apache Sqoop - Import/export - Commands - RDBMS <------>
HDFS/Hive/Hbase
Apache Oozie - Job Scheduler - Workflows
Hbase - NoSQL DB - Queries -OLTP - Real time

Mysql -> Web API -> .log, .csv on server
|
|
| --> Web/Third Party tool --> Transaction Data

Fraud Detection -

Case : Bank 2 years data.. Data formats -

.log files (2TB) - Web logs
mysql (100MB) - Personal data
.csv files (1TB) - Transactional data

Generate report of all customers who have done more than 10L online transaction in last 6 months from laptop and mozila browser.

Apache Pig:

Pig is an abstraction over MapReduce. Yahoo 2006,2007,2010 to Apache, Hive FB --> Apache, HDFS concept is from Google FS (GFS)

Pig provides High level lang - Pig Latin

Why to use:
ETL
Can perform MR tasks without having complex java code.
Multi query approach and optimization
Extensability
UDF's (User Defined Functions)

Apache Pig Architecture:

Pig Latin Scripts ---> Apache Pig [Pig Sever --> Parser --> Compiler --
>Execution Engine] ----> Map Reduce ---> HDFS

Parser -> Syntax, type checking of pig code Output of parser goes as -->
DAG (Directed Acyclic Graph) - Logical plan --->

Compiler -> create series of map reduce jobs --> Pig Execution Engine
Submit to MR engine
--->HDFS

Pig Latin Data Model:

Atom - Field/Any single value Ex: 25, 'abcd'
Tuple - Formed by an ordered set of Fields. Ex: (25,'abcd'),(26,'xyz')
Bag - Unordered set of Tuples. Ex: {(25,'abcd'),('xyz',26,9822)}
Relation/aliase - This bag is also called as Relation which holds this bag.
Map - Set of key-value pairs. Ex: [name#abcd,age#30],[name#xyz,age#35]

Pig Execution Mechanism:

1. Interactive Mode - Grunt Shell -> Local mode (Local system) and Map reduce mode (HDFS)

2. Batch/Script Mode- .pig extention file. Ex: demo.pig --> Submit to Pig

3. Embeded UDF - Using java will create UDF i.e. .jar and register to pig

---

Invoke Grunt Shell:

Local Mode:
Terminal > pig -x local

Press CTRL+C to exit from Grunt Shell OR Type > quit

MapReduce Mode:
Terminal > pig -x mapreduce
OR
Terminal > pig

Pig Latin statements:
Statement works with relations. They include epxressions and schemas.
It ends with ';'

Every statement take a relation as input and produce another relation as output.

Data types:
int, long,float,double,chararray,bytearray,boolean,datetime

Default data type is always a bytearray

Complex types: Tuple, Bag, Map

Operators in Pig:

Reading a data - LOAD operator

Syntax: Relation_name = LOAD 'inputfile_path' USING function_name AS schema;

Displaying a relation - DUMP operator
Syntx: DUMP Relation_name;

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);

describe employee;

DUMP employee;
```

STORE Operator:

```
STORE relationname INTO 'directoryPath' USING functionname;

STORE employee INTO 'Dekstop/emp_output' USING PigStorage(',');
```

1. success (empty file)
2. part-m-00 (data)

Diagnostic Operators:

Dump - To run the pig statement and display the result on screen.
Describe - View the schema of a relation.
Explain - is used to display the logical,physical and map reduce execution of a relation.
Illustrate - gives you step-by-step execution of a sequence of statements.

Grouping and Joining:

GROUP Operator:
is used to group the data in one or more relations. It collects the data having same key.

```
Syntax: groupRelationName = GROUP ExistingRelation BY key;

groupdno = GROUP employee BY dno;

describe groupdno;
dump groupdno;
```

Multiple columns grouping:

```
groupdnosal = GROUP employee BY (dno,esal);
describe groupdnosal;
dump groupdnosal;


groupalldata = GROUP employee ALL;
describe groupalldata;
dump groupalldata;

(all,{ })
```

Cogroup operator:

is used for two or more relations.

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);


dept = LOAD 'Desktop/dept.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);

cogroupdata = COGROUP employee BY dno, dept BY dno;

describe cogroupdata;

dump cogroupdata;

(1,{(e1.1),(e1.2)},{(d1.1)})
(2,{(e2.1),(e2.2)},{(d2.1)})


Take two files emp.log and emp.txt on hdfs and create cogroup on these
based on sal
Store this data into hdfs directory name /user/cloudera/cogroup_sal

hadoop fs -put Desktop/emp.log /user/cloudera/
hadoop fs -put -f Desktop/emp.log /user/cloudera/

employee1 = LOAD '/user/cloudera/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
employee2 = LOAD '/user/cloudera/emp.log' USING PigStorage('\t') AS
(eid:int,ename:chararray,city:chararray,country:chararray,esal:int);

cogroupdata = COGROUP employee1 BY esal,employee2 BY esal;
dump cogroupdata;

STORE cogroupdata INTO '/user/cloudera/cogroup_sal' USING
PigStorage(',');

Joins:

Combine records from two or more relations with matched key.

Equi/Inner
Left/Right/Full outer join

Syntax inner join:
newJoinedRealtion = JOIN Relation1 BY Key, Relation2 BY key;


employee = LOAD 'Desktop/Data/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
dept = LOAD 'Desktop/Data/dept.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);

empdept = JOIN employee BY dno,dept BY dno;


describe empdept;
dump empdept;

8 tuples with 9 columns

Left Outer:
```

```
empdept1 = JOIN employee BY dno LEFT OUTER,dept BY dno;

Right Outer:
empdept2 = JOIN employee BY dno RIGHT OUTER,dept BY dno;

Full Outer:
empdept3 = JOIN employee BY dno FULL OUTER,dept BY dno;


CROSS operator:

Relationname = CROSS Relation1,Relation2;
crossdata = CROSS employee,dept;

UNION:
Merge the contents of two relations. Rule: Their column and domains must
be identical.

dept1 = LOAD 'Desktop/dept.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
dept2 = LOAD 'Desktop/dept1_1.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
dept = UNION dept1,dept2;
dump dept;

describe dept;


SPLIT Operator:

Split a relation into two or more relations.

SPLIT relationename INTO relation1 IF (condition1), relation2 IF
(condition2);

SPLIT dept INTO ls35k IF dsal<35000, gt35k IF dsal>35000;
dump ls35k;
dump gt35k;


Q1. Display dno and location of dept having sal=35k


FILTER Operator:
select the required tuples from a relation based on a condition.

deptdata = FILTER dept BY dsal == 35000;
DUMP deptdata;

FOREACH Operator:
dnoloc = FOREACH deptdata GENERATE dno,dloc;
DUMP dnoloc;

Assignment: emp.txt and dept.txt join them on dno and display ename,dname
with
emp salary < 28000.


DISTINCT:
Used to remove redundant/duplicate tuples.
```

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empdata = DISTINCT employee;
dump empdata;

ORDER BY:
empsalorder = ORDER empdata BY esal DESC;
dump empsalorder;

topsalrecord = LIMIT empsalorder 1;
dump topsalrecord;
```

MapReduce Mode:

1. Display lowest salary of emp.txt only.
Output: 25000

2. Display the eid,ename,doj having that lowest salary.

3. Display ename and dloc having highest salary.

4. Store eid and doj having highest salary and dept location india.
HDFS Location: /user/cloudera/empid

27-12-2018

Built in Functions:

eval functions:

AVG,COUNT,COUNT_STAR,SUM,MAX,MIN,SIZE,SUM,TOKENIZE,FLATTEN

Display max salary from emp?

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empgroup = GROUP employee ALL;
empsal = FOREACH empgroup GENERATE MAX(employee.esal) as maxsal;
empdata = FILTER employee BY esal == empsal.maxsal;
dump empdata;
```

Local mode:
count the employees having min salary?

Map Reduce Mode: HDFS
Store employee information having min salary without limit operator?
File Location: /user/cloudera/emp.txt
Store Location: /user/cloudera/empminsal

1. hadoop fs -put Desktop/emp.txt /user/cloudera/

2. Terminal > pig
OR

2. Terminal > pig -x mapreduce

3.
```
employee = LOAD '/user/cloudera/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empgroup = GROUP employee ALL;
empsal = FOREACH empgroup GENERATE MIN(employee.esal) as minsal;
empdata = FILTER employee BY esal == empsal.minsal;
emp1 = GROUP empdata ALL;
emp2 = FOREACH emp1 GENERATE COUNT(empdata.eid);
dump emp2;
```

Terminal > hadoop fs -cat /user/cloudera/empminsal/part*

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empdata = FOREACH employee GENERATE SIZE(ename) as sizename;
dump empdata;
```

```
empmaxsal = FILTER employee BY esal == empsal.maxsal;
empgroupmax = GROUP empmaxsal ALL;
empmaxsalcount = FOREACH empgroupmax GENERATE COUNT(empmaxsal.eid);
dump empmaxsalcount;
```

Now: 1. Display no. of employees having min salary without limit operator?

Home Assignment :
2. Display employee information having second max salary?
3. Map Reduce Mode - Give 10% increment to emp having sal is less than avg salary and store details to /user/cloudera/empavgsal

---

DIFF : Its used to compare two bags or fields in a tuple.

Syntax: DIFF (expre1,expre2)

```
dept1 = LOAD 'Desktop/dept.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
dept2 = LOAD 'Desktop/dept2.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
cogroupdept = COGROUP dept1 BY dno,dept2 BY dno;
```

```
diff_dept = FOREACH cogroupdept GENERATE DIFF(dept1,dept2);
dump diff_dept;
```

SUBTRACT:
subtract two bags. It returns a bag which contains tuples of first bag that are not in second bag.

```
sub_dept = FOREACH cogroupdept GENERATE SUBTRACT(dept1,dept2);
dump sub_dept;
```

TOKENIZE:
Its used to Split a string in a single tuple and returns a bag.

Flatten: Unbag the tuples from a bag.

Words count from a file:

```
lines = LOAD 'Desktop/Data/wordcount.txt' as (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grp = GROUP words BY word;
wordcount = FOREACH grp GENERATE group, COUNT(words);
dump wordcount;
```

Date and Time Functions:

```
ToDate,CurrentTime,GetDay,GetHour,GetMilliSecond,GetMinute,GetMonth,GetSe
cond,
GetWeek,GetYear,ToString,DaysBetween,HoursBetween,MilliSecondsBetween,Min
utesBetween, MonthsBetween,SecondsBetween,WeeeksBetween,YearsBetween
```

Get Month from doj of emp:

```
employee = LOAD 'Desktop/emp.txt' USING PigStorage(',') AS
(eid:int,ename:chararray,esal:int,dno:int,doj:chararray);
empdate = FOREACH employee GENERATE ToDate(doj,'yyyy/MM/dd HH:mm:ss') as
(dojdate:DateTime);
serviceyears = FOREACH empdate GENERATE
YearsBetween(CurrentTime(),dojdate);
dump serviceyears;
```

Local Mode:
Display employee eid,ename who have done maximum service in my company?
Count the people who joined my company this year 2018?

MapReduce Mode:
Give 10% increment to people spend more than 3 years and store this result on hdfs.
Give 5% increment to people spend months in between 30 and 35 and store this result on hdfs.

Piggybank, XML and json handling

matches

Check mobile starts with 98.

9822334455
9955866456

```
filter somedata by (chararray) mobile matches '98.*';
```

Piggybank for csv,xml,json (java standard object notation)

csv vs .log,.txt

a.csv
name,mobile,address
'acd,acde',123,tyty

a.txt/.log
name,mobile,address
acd,acde,123,tyty

Online Avertisement --> API Application ---> Spring/Hibernate ---> Click --> .log,.csv,.json ---> Process and store --> table (Final statistic table) --> Report

Piggybank .. its a jar file
Its a standard with Pig.

CSV file:

Register that jar file to pig..

```
> Register jarfile_location;
```

```
cars =  LOAD 'Desktop/Data/cars.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',') AS
(buying:chararray,maint:chararray,doors:chararray,persons:chararray,lug_b
oot:chararray,safety:chararray,remark:chararray);
```

XML file: Xpath way

```
Register piggybank.jar;

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();
a = LOAD 'Desktop/Data/testXML.xml' USING
org.apache.pig.piggybank.storage.XMLLoader('document') as (x:chararray);
b = FOREACH a GENERATE
XPath(x,'document/url'),XPath(x,'document/category'),XPath(x,'document/us
ercount');
c = LOAD 'Desktop/Data/testXML.xml' USING
org.apache.pig.piggybank.storage.XMLLoader('review') as (r:chararray);
d = FOREACH c GENERATE XPath(r,'review');
e = CROSS b,d;
dump e;
```

JSON Handling:

Simple json:

```
firstjson = LOAD 'Desktop/Data/first.json' USING
JsonLoader('food:chararray,person:chararray,amount:int');
dump firstjson;

secondjson = LOAD 'Desktop/Data/second.json' USING
JsonLoader('recipe:chararray,ingredients:{(name:chararray)},inventor:(nam
e:chararray,age:int)');
dump secondjson;

thirdjson = LOAD 'Desktop/Data/third.json' USING
JsonLoader('recipe:chararray,ingredients:{(name:chararray)},inventor:(nam
e:chararray,age:int)');
dump thirdjson;
```

UDF (User Defined Function) using Java:

Eclipse -> Create a project --> build Pig jars --> extends a class --> Export as a jar
--> Register jar --> Use that function as some name

Demo: 4th power of number FourthPower(3) => 3*3*3*3 => 27*3 => 81
Assignment: Factorial of number function--> Facto(5) => 120

Extend EvalFunc in Pig jars..

Eclipse -> New Project -> Right click on project --> new Folder ->
Copy some jars from /usr/lib/pig into folder created
Build path => Export as jar on Desktop.

Implementation:

```
REGISTER 'Desktop/MyPigUDF.jar';
dept = LOAD 'Desktop/Data/dept.txt' USING PigStorage(',') AS
(dno:int,dname:chararray,dloc:chararray,dsal:int);
dnopower = FOREACH dept GENERATE myudf.Facto(dno);
-----------------------------------------------------
dnofact = FOREACH dept GENERATE myudf.Facto(dno);
```

Home Assignment:
1. NoOfVowels(String) Ex: NoOfVowels('abcde') => 2
2. Give 30% hike to emp whose salary is maximum and years spend are more and Save this data to hdfs /user/cloudera/emp30max/
3. Count the people whose day of joining is odd number. Use own UDF.

10 Time efforts -> Note down

Map Reduce Program .. Demo word count...

Java - 30-40 lines
Pig: 4-5 lines
Spark Scala: 1-2 line
Python: 2-3 lines

https://bit.ly/2CQfhhH

commons-cli-1.2.jar
hadoop-common-2.6.0-cdh5.13.0.jar
hadoop-common.jar
hadoop-core-2.6.0-mr1-cdh5.13.0.jar
hadoop-core-mr1.jar

```
hadoop fs -put Desktop/demo.txt /user/cloudera/

sudo hadoop jar Desktop/MRProgram.jar PackageDemo.WordCount
/user/cloudera/demo.txt
/user/cloudera/MROutput
```

---

demo.pig

```
pig -x local demo.pig
pig -x mapreduce demo.pig
```

HIVE

Apache Hive: DW tool to process the data in hadoop. It helps us to make queries and analyzing.

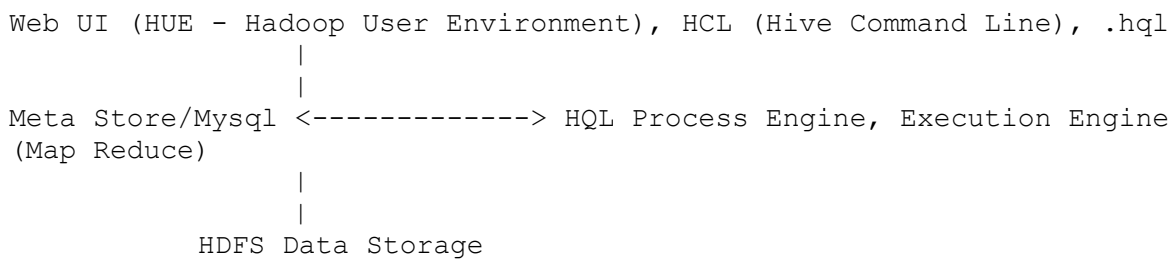
It was developed by FB and given to Apache Software Foundation.

Hive is not:
RDBMS, OLTP, Real time queries and row level updates.

Normalization - RDBMS (ML,DDL,DCL) + Normalized (Relation - Contraints)
ACID Properties

Features of Hive:
It stores schema in database (mysql) and process data into hdfs.
By default there is Derby DB with hive. You can not have multiple sessions work.
It is designed for OLAP.
It provides SQL type lang callled HQL.
Fast, Scalable and extensible.

Architecture of Hive:



HCL Terminal > hive

1. CREATE DATABASE [IF NOT EXISTS] <databasename>

```
hive > create database if not exists mydb;
hive > show databases;
hive > use mydb;
hive > set hive.cli.print.current.db=true
```

```
hive (mydb) > use default;
hive (default) >

2. DROP DATABASE [IF EXISTS] <databasename> [CASCADE]
hive (default) > drop database mydb;

      hive > create database if not exists mydb;
      hive > use mydb;
      hive (mydb) >

3. CREATE [TEMPORARY][EXTERNAL] TABLE [IF NOT EXISTS]
[dbname.]<tablename>
<col1 datatype,col2 datatype....>
[COMMENT tablecomment]
[ROW FORMAT DELIMITED]
[FIELDS TERMINATED BY '']
[LINES TERMINATED BY '']
[STORED AS fileformat]

ex: emp
hive >
CREATE TABLE IF NOT EXISTS mydb.emp (eid int,ename string, esal int,city
string,dno int)
COMMENT 'this is my emp table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

hive > describe emp;

4. LOAD DATA STATEMENT:
Syntax: LOAD DATA [LOCAL] INPATH <'filepath'> [OVERWRITE] INTO TABLE
<tablename> [PARTITION (partCol1,..)];

ex: hive > LOAD DATA LOCAL INPATH 'Desktop/emp1.txt' INTO TABLE emp;
    hive > select * from emp;
    hive > LOAD DATA INPATH '/user/cloudera/emp1.txt' INTO TABLE emp;
    hive > LOAD DATA INPATH '/user/cloudera/emp1.txt' OVERWRITE INTO
TABLE mydb.emp;

hadoop fs -ls /user/hive/warehouse/mydb.db/emp

have hdfs  --> loading this to hive --> hive will move this file to above
location.

Similarly try with dept.txt (tab seperate)

Types of tables :
3 types : Temporaray, Internal/managed, External table

By default : table is internal or managed
Data and table are together..
If drop this table - table/schema and data will get dropped.

hdfs > /user/hive/warehouse/db.dbname/tablename

hive --> Drop table emp;   <---> hdfs  --> schema --> mysql

Temporay: upto sessionc only.
```

```
CREATE TEMPORARY TABLE IF NOT EXISTS emptemp (eid int,ename string, esal
int,city string,dno int)
COMMENT 'this is my emp table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Scope of this table is for current session only. Hive sessios closed this table will get dropped.

```
LOAD DATA LOCAL INPATH 'Desktop/emp1.txt' INTO TABLE emptemp;
```

External Table:
External to data in hive..

HDFS flipkartlogs/10:30/1021.log

```
101,Laptop,Mozila,Ubuntu,16,978.15.62.31,india,pune,fcroad,10:21:50:500,s
hoes
102,Mobile,Crhome,Andoid,4,978.15.62.30,india,pune,kothrud,10:21:40:500,c
loths
105,Mobile,Crhome,Andoid,4,,india,pune,kothrud,10:22:40:500,cloths
```

HDFS flipkartlogs/10:30/1022.log

```
103,Laptop,Mozila,Ubuntu,16,978.15.62.25,india,pune,fcroad,10:22:10:500,s
hoes
104,Mobile,Crhome,Andoid,4,978.15.62.38,india,pune,kothrud,10:22:30:500,w
atch
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS userlogext (uid int,device
string,browser string,os string, osversion int,ip string,country
string,city string,street string,tt string,product string)
COMMENT 'this is my ext table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/cloudera/10_30/';
```

```
CREATE TABLE finalstat (device string,os string,interest int)   <--
product wise partition
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

INSERT INTO finalstat select street,product,count(*) from userlogext
group by street,product;
```

Give me the report of today morning 10.30 who have visited our site for shoes.

```
select sum(interest) from finalstat where product='shoes';
```

Give me the report of today morning 10.30 who have visited our site from laptop for cloths?

Terminal > hive -f Desktop/flip.hql

Assignment:
XML file --> Pig---> store /user/cloudera/mypigdata/ --> Hive External table --> final table

ETL + DW

Query optimizations...
Partitioning and JOINS:

Divide table into different partitions...It will create different folders in table as per partition column/field. Paritioning can be done on one more columns.

Total 10Gb data..
yearly..

2014 -2
2015 -2
2016 -2
2017 -2
2018 -2

select count(*) from thisdata where year='2018';
10Gb data processing and then filtering...

More cardinatlity..

Two types of partitioning:

Static - We specifies the partition value.
Load statement for loading a file
Saves your time

Dynamic - We dont specify the partition value.
Insert statement from external table to final table.

10 Gb of mixed data.. millions of rows with year...
Dynamic

2 Gb file .. 2014
Static

```
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.exec.dynamic.partition=true;
CREATE TABLE IF NOT EXISTS userlogpart (uid int,device string,browser string,os string, osversion int,ip string,country string,city string,street string,tt string,product string, day int)
PARTITIONED BY (month int,year int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Static Way:

```
LOAD DATA LOCAL INPATH 'Desktop/301219.log' INTO TABLE userlogpart
PARTITION(month=12,year=2019);

Dynamic Way:
CREATE EXTERNAL TABLE IF NOT EXISTS userlogext(uid int,device
string,browser string,os string, osversion int,ip string,country
string,city string,street string,tt string,product string, day int, month
int, year int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/user/umesh/10_30/';
INSERT INTO userlogexe PARTITION(month,year) select * from userlogpart;


hadoop fs -ls /user/hive/warehouse/userlogpart/


HDFS flipkartlogs/10:30/1021.log

101,Laptop,Mozila,Ubuntu,16,978.15.62.31,india,pune,fcroad,10:21:50:500,s
hoes
102,Mobile,Crhome,Andoid,4,978.15.62.30,india,pune,kothrud,10:21:40:500,c
loths
105,Mobile,Crhome,Andoid,4,,india,pune,kothrud,10:22:40:500,cloths


HDFS flipkartlogs/10:30/1022.log
103,Laptop,Mozila,Ubuntu,16,978.15.62.25,india,pune,fcroad,10:22:10:500,s
hoes
104,Mobile,Crhome,Andoid,4,978.15.62.38,india,pune,kothrud,10:22:30:500,w
atch


Create finalstat table with PRODUCT wise partitioning..Ignore if ip
address in not there
with columns (device string,os string,interest int)

select * from finalstat;
Laptop,Ubuntu,2,shoes
Mobile,Android,1,cloths
Mobile,Android,1,watch



hive -f my.hql
```