NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA
New Horizon Knowledge Park, Ring Road, Bellandur Post, Bengaluru 560 103

DEPARTMENT OF MCA

III Semester MCA

Big Data Analytics Lab

(20MCAL37A)

D PRATHAP REDDY

1NH21MC023

COURSE COORDINATOR

Prof. Neethu Tressa

2022-23

NEW HORIZON
COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA
New Horizon Knowledge Park, Ring Road, Bellandur Post, Bengaluru 560 103

NEW HORIZON COLLEGE OF ENGINEERING

(Autonomous Institution Affiliated to VTU)

Accredited by NAAC with 'A' Grade

BANGALORE

DEPARTMENT OF MCA

CERTIFICATE

This is to certify that **D PRATHAP REDDY (1NH21MC023)** has successfullycompleted the lab work in ***Big Data Analytics Lab (20MCAL37A)*** in partialfulfillment of the III Semester course of MCA for the academic year ***2022-23.***

COURSE COORDINATOR **HEAD OF THE DEPARTMENT**

External Examiners

1)_____

2)_____

INDEX

	c) Performing MAP Reduction operation in Hadoop. Verification of operation results through terminal and browser.
5.	**Hadoop Ecosystem:** **I. Sqoop commands :** a) Import of tables from Mysql database to hdfs b) Export files from hdfs to mysql database **II. Pig commands:** a) Loading and storing - from/to local file system, from/to hdfs b) ii) Diagnostic operator – Dump c) iii) Filter operator **III. Hive Commands:** a) Data Definition Language (DDL): CREATE database/ table/ external table , DROP, ALTER, SHOW, DESCRIBE Statements. b) Data Manipulation Language (DML): LOAD, INSERT Statements- INSERT INTO, INSERT OVERWRITE. **IV. HBase shell Command:** Create table with /without version – a) Put command b) Get command with / without version c) Delete column – column family d) Drop table

MODULE 1

Experiment 1

Introduction to SQL:

Creation of tables with constraints and insertion of values into tables

1. Write SQL statements to create a table "employees" including columns employee_id, first_name, last_name, job_id, salary and make sure that, the employee_id column does not contain any duplicate value at the time of insertion. The foreign key column job_id should be referenced by the column job_id of "jobs" table. "jobs" table can contain columns job_id, job_title, min_salary, max_salary and check whether the max_salary amount exceeding the upper limit 25000. Also, make sure that, the default value for job_title is blank and min_salary is 8000.

```
mysql> create database empjob;

mysql> use empjob;

mysql> create table job(jobid int(20) primary key,jobtitle varchar(20) default ' ',minsalary int(20) check (minsalary>=8000) ,maxsalary int(20) check (maxsalary<=25000));

mysql>mysql> create table employee(empid int(20) primary key,firstname varchar(20) ,lastname varchar(20),jobid int(20) references job(jobid), salary int(20));
```

2. Insert meaningful values into the tables.

```
mysql> insert into job values ('101','QA Engineer','10000','20000');

mysql> insert into job values ('102','Analyst','8000','24000');
```

```
mysql> insert into job values ('103','Manager','15000','25000');

mysql> insert into employee values ('100','MANU','Rajan','103','25000');

mysql> insert into employee values ('111','SHYAM','Raj','102','18000');

mysql> insert into employee values ('112','EKTA','M','102','15000');


mysql> select * from employee;
+_____+________+_______+_____+_____+
| empid | firstname | lastname | jobid | salary |
+_____+________+_______+_____+_____+
|   100 | MANU      | Rajan    |   103 |  25000 |
|   111 | SHYAM     | Raj      |   102 |  18000 |
|   112 | EKTA      | M        |   102 |  15000 |
+_____+________+_______+_____+_____+

mysql> select * from job;
+_____+_________+_______+_______+
| jobid | jobtitle    | minsalary | maxsalary |
+_____+_________+_______+_______+
|   101 | QA Engineer |     10000 |     20000 |
|   102 | Analyst     |      8000 |     24000 |
|   103 | Manager     |     15000 |     25000 |
+_____+_________+_______+_______+
```

Experiment 2

Hands-on DML commands to apply different aggregate functions, Group by-Having-Order by clause, operators

1. Display the minimum, maximum and average salary from "employees" table.

```
mysql> SELECT MIN(salary) FROM employee;
+_____+
| MIN(salary) |
+_____+
|      15000 |
+_____+

mysql> SELECT MAX(salary) FROM employee;
+_____+
| MAX(salary) |
+_____+
|      25000 |
+_____+

mysql> SELECT AVG(salary) FROM employee;
+_____+
| AVG(salary) |
+_____+
|  19333.3333 |
+_____+
```

2. Display the number of employees grouped by salary. Only include employees with salary more than 10000.

```
mysql> SELECT salary FROM employee WHERE salary>=10000 GROUP
BY salary;
+_____+
```

```
| salary |
+_____+
| 25000 |
| 18000 |
| 15000 |
+_____+
```

3. Display the employee details in the descending order of salary.

```
mysql> SELECT * FROM employee ORDER BY salary DESC;
+_____+_______+_______+_____+_____+
| empid | firstname | lastname | jobid | salary |
+_____+_______+_______+_____+_____+
|  100 | MANU      | Rajan    |  103 |  25000 |
|  111 | SHYAM     | Raj      |  102 |  18000 |
|  112 | EKTA      | M        |  102 |  15000 |
+_____+_______+_______+_____+_____+
```

4. Display the details of employees whose salary is 10000 or 20000 using IN operator.

```
mysql> SELECT * FROM employee WHERE salary IN ('10000','20000');
Empty set (0.03 sec)
```

5. Display the details of employees whose salary is between 10000 and 20000.

```
mysql> SELECT * FROM employee WHERE salary BETWEEN 10000
AND 20000;
+_____+_______+_______+_____+_____+
| empid | firstname | lastname | jobid | salary |
+_____+_______+_______+_____+_____+
|  111 | SHYAM     | Raj      |  102 |  18000 |
|  112 | EKTA      | M        |  102 |  15000 |
+_____+_______+_______+_____+_____+
```

6. Display the details of employees whose first name starts with the letter 'M' using LIKE operator.

```
mysql> SELECT * FROM employee where firstname like 'A%';
+_____+________+_______+_____+_____+
| empid | firstname | lastname | jobid | salary |
+_____+________+_______+_____+_____+
|   100 | MANU       | Rajan    |   103 |  25000 |
+_____+________+_______+_____+_____+
```

Experiment 3

Creation of views and working with joins

1. Create a view that shows all employees with salary = 20000. Display the view data.

```
mysql> CREATE VIEW Emp AS SELECT salary FROM employee
WHERE salary=20000;

mysql> select * from Emp;
Empty set (0.04 sec)
```

2. Display employee_id and job-title from the tables employees andjobs. Perform Inner join, Left join, Right Join and Full join for the selection.

```
mysql>  SELECT employee.empid,job.jobtitle FROM employee INNER
JOIN job ON employee.jobid = job.jobid;
+_____+_______+
| empid | jobtitle |
+_____+_______+
|   100 | Manager  |
|   111 | Analyst  |
|   112 | Analyst  |
+_____+_______+

mysql> SELECT employee.empid,job.jobtitle FROM employee LEFT
JOIN job ON employee.jobid = job.jobid;
+_____+_______+
| empid | jobtitle |
+_____+_______+
|   100 | Manager  |
|   111 | Analyst  |
|   112 | Analyst  |
+_____+_______+
```

```
mysql> SELECT employee.empid,job.jobtitle FROM employee RIGHT
JOIN job ON employee.jobid = job.jobid;
+_____+_________+
| empid | jobtitle    |
+_____+_________+
|  NULL | QA Engineer |
|   112 | Analyst     |
|   111 | Analyst     |
|   100 | Manager     |
+_____+_________+
```

Experiment 4

Hadoop Hands on sessions

1. Checking hadoop configuration files.

(a) To verify that all software's are in good health or not

1. go to browser
2. click on “Cloud Manager”
3. enter Username: cloudera , Password: cloudera
4. click on “login”
5. check whether required software is in green color(good health) ; if no, click actions button and restart

(b) To check java path

$ cd /usr/lib/jvm
$ ls

(c) To check hadoop location

$ cd /usr/lib/hadoop-0.20-mapreduce/
$ ls

(d)To verify hadoop installation files

[/lib/hadoop-0.20-mapreduce] $ cd conf
[/lib/hadoop-0.20-mapreduce] $ ls
$gedit core-site.xml (similarly we can open other files)
$ gedit mapred-site.xml
$ edit hdfs-site.xml

Experiment 5

Loading a file from local file system to hadoop file system.

1. **To create a folder or directory in hadoop**
   $hadoop fs –mkdir /user/cloudera/nh001

2. **To verify whether or not the folder is created**
   $ hadoop fs –ls /user/cloudera

3. **To create a file named test in local file system**
   $ gedit test

4. **To verify whether or not the file is created**
   $ls

5. **To put the local file into hadoop file system**
   $ hadoop fs –put /home/cloudera/test /user/cloudera/nh001

6. **To verify whether or not the local file is loaded into hadoop file system**
   $hadoop fs –ls /user/cloudera/nh001

7. **To check the content of loaded file**
   $hadoop fs –cat /user/cloudera/nh001/test

To Verify the loaded file from browser

a) Open the firefox browser in cloudera
b) Click the bookmark: **hdfs Namenode**
c) Click **browse filesystem**
d) Click **user**
e) Click **cloudera**
f) Click **nh001**
g) Click **test**

Experiment 6

Perform analysis on loaded files using hadoop map reduce programs and verify the output using hadoop commands as well as browser.

a) Count
b) Grep

1. To see the list of jar files available in hadoop

$cd /usr/lib/hadoop-0.20-mapreduce/
$ ls

2. To see the content of jar file

$hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples-2.0.0-mr1-cdh4.4.0.jar

(OR)

$hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar

To run word count program on loaded file and creating output file path.

(a) Wordcount:

Syntax :

hadoop jar <jar file path> wordcount <input file path on hdfs><output file path on hdfs with new output directory name>

$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop- examples-2.0.0-mr1-cdh4.4.0.jar wordcount /user/cloudera/nh001/test /user/cloudera/nh001/outwc

- To verify output files

$hadoop fs -ls /user/cloudera/nh001
$ hadoop fs -ls /user/cloudera/nh001/outwc

- To see the content of output file

$ hadoop fs -cat /user/cloudera/nh001/outwc/part-r-00000

(OR)

$ hadoop fs -cat /user/cloudera/nh001/outwc/part*

- **To see the output through browser**
  click on “HDFS Namenode” -> Browse the file system->user->cloudera ->nh001 ->outwc ->part-r-00000

(b) Grep

Syntax :

hadoop jar <jar file path> grep <input file path on hdfs><output file path on hdfs with new output directory name><key_word>

To run Grep program on loaded file with keyword ‘ball’ and creating output file path.

$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples-2.0.0-mr1- cdh4.4.0.jar grep /user/cloudera/nh001/test /user/cloudera/nh001/outgrep ball

(OR)

$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep /user/cloudera/nh001/test /user/cloudera/nh001/outgrep ball

SQOOP Hands on Sessions

Experiment 7

Verifying Sqoop status through cloudera manager and create database and user account in Mysql

For SQOOP

- Open the browser of VM and select "Cloudera Manager"
- Check whether or not SQOOP is in good health
- If no, then restart the SQOOP in "actions" button

For Mysql

- open the terminal

- To start mysql services

$ sudo service mysqld start

- To connect to mysql as root login

$ mysql –u root –p

or

mysql -u root (press enter twice to login)

1) To create your own account

$mysql> create user nh001 identified by '1234';

2) To show the existing users

$mysql> select user from mysql.user;

3)To create Database

$mysql> create database lab;

4) To show the existing data bases

$mysql> show databases;

5)To grant permission to the user for creating tables on that database

$mysql> grant all on lab.* to 'nh001';

6)Exit

$mysql> quit

Steps to be followed after login to your account

1) To login with your account

$ mysql –u nh001 –p

2) To verify the database names

$mysql> show databases;

3) To choose the database you want to use

$mysql> use lab;

4) To create tables

$mysql> create table emp(empno int primary key, ename varchar(10), age int);

$mysql> create table dept(dno int primary key, dname varchar(5));

5) To insert records into the tables

$mysql> insert into emp values(1001,'ram',21), (1002,'sita',22), (1003,'ravi',23), (1004,'teja',21), (1005,'meena',22), (1006,'mona',20), (1007,'sona',21), (1008,'harish',23), (1009,'james',35), (1010,'jacob',30);

$mysql> insert into dept values(10,'cse'), (20,'ise'), (30,'me'), (40,'mba'), (50,'mca'), (60,'au'), (70,'civil'), (80,'ece'), (90,'eee');

6) Exit

$mysql> quit

Experiment 8

Hand-on Practice on various Sqoop basic commands

(a) To List-databases

$ sqoop list-databases --connect "jdbc:mysql://localhost" --username nh001 --password 1234

(b) To list the tables

$ sqoop list-tables --connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234

Experiment 9

Import of tables from Mysql database to hdfs

- Import of all tables to default directory and specific directory
- Import of specific tables to default directory and target directory
- Import of subset of tables using 'where' clause to default and specific directory
- Import as sequence file
- Incremental import

1. Import-all-tables to default directory

- The import-all-tables tool imports all tables from database lab to hdfs. Data from each table is stored in a separate directory in HDFS.
- Each table must have a single-column primary key or **-m 1** option must be used(to make no: of mappers as one)

$ sqoop import-all-tables --connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234

2. To check whether or not tables are imported
$ hadoop fs -ls /user/cloudera

3. To check for a particular table
$ hadoop fs -ls /user/cloudera/dept

4. To see the records of mysql table on hdfs file
$hadoop fs -cat /user/cloudera/dept/part-m-00000

$ hadoop fs –cat /user/cloudera/dept/part-m-00001

$ hadoop fs -cat /user/cloudera/dept/part-m-00002

$ hadoop fs -cat /user/cloudera/dept/part-m-00003
(OR)
$ hadoop fs -cat /user/cloudera/dept/part*

1.1 Import-all-tables to a specific directory

1. To create a specific directory
   $ hadoop fs -mkdir /user/cloudera/jisha

2. To import mysql all tables to newly created directory
   $ sqoop import-all-tables –-connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234 --warehouse-dir /user/cloudera/jisha

3. To check whether or not tables are imported
   $ **hadoop fs -ls /user/cloudera/jisha**

4. To check for a particular table
   $ hadoop fs -ls /user/cloudera/jisha /dept

   $ hadoop fs -ls /user/cloudera/jisha /emp

5. To see the records of mysql table on hdfs file
   $hadoop fs -cat /user/cloudera/jisha /emp/part*

1.2 Import-all-tables to specific directory with only one mapper(-m stands for no:of mappers)

1. To remove already existing tables in the current directory
   $ hadoop fs -rm -R /user/cloudera/jisha /*

$ sqoop import-all-tables --connect jdbc:mysql://localhost/lab" --username nh001 --password 1234 --warehouse-dir /user/cloudera/jisha -m 1

2. To check whether or not tables are imported
   $ hadoop fs -ls /user/cloudera/jisha /

3. To check for a particular table
   $ hadoop fs -ls /user/cloudera/jisha /emp

4. To see the records of mysql table on hdfs file
   $ hadoop fs -cat /user/cloudera/jisha /emp/part-m-00000

2. Import of specific table to default directory of hdfs

1. To remove mysql table's same name file from default directory(to avoid file already exists error)

$ hadoop fs -rm -R /user/cloudera/jisha/*

$ sqoop import --connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234 --table dept

2. To check whether or not tables are imported

$ hadoop fs -ls /user/cloudera/

3. To check for a particular table

$ hadoop fs -ls /user/cloudera/dept

4. To see the records of mysql table on hdfs file

$ hadoop fs -cat /user/cloudera/dept/part*

(2.1) Import of specific table from Mysql to a target directory of hdfs

$ sqoop import --connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234 --table dept --target-dir /user/cloudera/jisha/deptnew

- To check whether or not tables are imported

  $ hadoop fs -ls /user/cloudera/jisha

- To check for a particular table

  $ hadoop fs -ls /user/cloudera/jisha/deptnew

- To see the records of mysql table on hdfs file

  $ hadoop fs -cat /user/cloudera/jisha/deptnew/part*

Experiment 10

Export files from hdfs to Mysql database

- The export tool exports a set of files from HDFS back to an RDBMS. The target table must already exist in the database. The input files are read and parsed into a set of records according to the user-specified delimiters.

- The default operation is to transform these into a set of INSERT statements that inject the records into the database

Step 1: Create a file on local file system using gedit with default delimiter as ' , ' and save it & close.

1. Load this file to hdfs (default or your own directory)

$ gedit sample.csv

111,a
112,b
113,c
114,d
115,e

$ hadoop fs -put /home/cloudera/sample.csv /user/cloudera/jisha

Step 2:login to Mysql and create an empty table with appropriate data type and primary key.

$mysql -u nh001 -p

$mysql> use lab;
$mysql> create table exportsam(value int primary key,name char);

Step 3: Perform Sqoop export command to load the hdfs file to mysql table.

$sqoop export --connect "jdbc:mysql://localhost/lab" --username nh001 --password 1234 --table exportsam --export-dir /user/cloudera/jisha/sample.csv

$mysql> select * from exportsam;

MODULE 2

Pig Latin - I

Experiment 11

I. Loading data stored on hdfs to Pig Storage and store it back to hdfs
II. Loading data stored on local file system to Pig Storage and store it back to local file system

Step 1: Preparing Hadoop hdfs

$ sudo jps

Create a folder with your USN in hadoop

$ hadoop fs -mkdir /user/cloudera/pig_batc3

Step 2: Placing the data on hdfs

- Create a file on local file system with delimiter ' , ' and copy it on to hdfs your directory

  $ gedit sample.txt

  001,Rajiv,Hyderabad
  002,siddarth,Kolkata
  003,Rajesh,Delhi
  004,Preethi,Pune
  005,Trupthi,Bhuwaneshwar
  006,Archana,Chennai.

- Copy it on to hdfs

  $ hadoop fs -put /home/cloudera/sample.txt /user/cloudera/pig_batc3

- Verify the file on hdfs (your folder)

  $hadoop fs -cat /user/cloudera/pig_batc3/sample.txt

Step 3: Run Apache Pig in mapreduce mode

$pig (or) $pig -x mapreduce

Step 4: Load the file

$grunt> student = LOAD '/user/cloudera/pig_batc3/sample.txt' USING PigStorage(',') as(id:int, name:chararray, city:chararray);

$grunt> Dump student;

Step 5: Store the output on Pigstorage after analysis to hdfs

$grunt> STORE student INTO'/user/cloudera/pig_batc3/out'USINGPigStorage (',');

Step 6: verify the file on hdfs

Step 1: Preparing Hadoop hdfs

$ sudo jps

Step 2: Placing the data on hdfs

$ gedit sample.txt

001,Rajiv,Hyderabad
002,siddarth,Kolkata
003,Rajesh,Delhi
004,Preethi,Pune
005,Trupthi,Bhuwaneshwar
006,Archana,Chennai.

Step 3: Run Apache Pig in local mode

$pig -x local

Grunt>

Step 4: load the file

$grunt> student = LOAD '/home/cloudera/sample.txt' USING PigStorage(',') as(id:int, name:chararray, city:chararray);

$grunt> Dump student;

Step 5: Store the output on Pigstorage after analysis to local filesystem desktop

$grunt> STORE student INTO '/home/cloudera/Desktop/out' USING PigStorage (',');

Experiment 12

Utility command and Diagnostic operators

Utility Commands

- Clear Command
  $grunt> clear

- Help Command
  $grunt> help

- history Command
  $grunt> history

- quit Command
  $grunt> quit

Dump Operator

$grunt> doc = LOAD '/user/cloudera/pig_batc3/doctor.txt' USING PigStorage(',') as (name:chararray,id:int,exp:int,hosp:chararray,fees:int);

$grunt> dump doc;

(Milan,1001,5,apollo,500)
(Jay,1002,10,Apollo,500)
(lalit,1003,20,manipal,500)
(Mohit,15,1004,15,)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)
(Jay,102,10,apollo,50)

Describe Operator

```
$grunt> describe doc;
```

Explain Operator

```
$grunt> explain doc;
```

Illustrate Operator

```
$grunt> illustrate doc;
```

Experiment 13

Filtering, Arthematic operators, Comparision operator, Boolean Operators

Three operators are used for filtering:

- FILTER Operator
- FOREACH Operator
- Distinct Operator

FILTER:

$grunt> a = filter doc by exp>10;

$grunt>dump a;

(lalit,1003,20,manipal,500)
(Mohit,1004,15,columbia,600)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)

DISTINCT:

$grunt> dump doc;

(Milan,1001,5,apollo,500)
(Jay,1002,10,apollo,500)
(lalit,1003,20,manipal,500)
(Mohit,1004,15,columbia,600)
(Chauhan,1005,30,narayana,550)
(Suraj,1006,25,manipal,650)
(Jay,1002,10,apollo,500)

$grunt> a = distinct doc;

$grunt> dump a;

(Jay,1002,10,apollo,500)
(Milan,1001,5,apollo,500)
(Mohit,1004,15,columbia,600)
(Suraj,1006,25,manipal,650)
(lalit,1003,20,manipal,500)
(Chauhan,1005,30,narayana,550)

FOREACH:

$**grunt> doc1=foreach doc generate *;**

$grunt> dump doc1;

(Jay,1002,10,apollo,500)
(Milan,1001,5,apollo,500)
(Mohit,1004,15,columbia,600)
(Suraj,1006,25,manipal,650)
(lalit,1003,20,manipal,500)
(Chauhan,1005,30,narayana,550)

$grunt> a = foreach doc generate name,hosp;

$grunt> dump a;

(Milan,apollo)
(Jay,apollo)
(lalit,manipal)
(Mohit,columbia)
(Chauhan,narayana)
(Suraj,manipal)

MODULE 3
HIVE

TO SWITCH OFF SAFE MODE
$ sudo –u hdfs hdfs dfsadmin –safemode leave

Experiment 14

DDL Commands

To Create Two Files On Local Filesystem And Copy It To Hdfs Any Folder

[cloudera@localhost ~]$ gedit emp.txt
1001|hari|d1|chennai|1986-12-10
1002|teja|d1|hyd|1987-01-21
1003|ram|d3|delhi|1986-02-11
1004|milind|d4|bang|1988-03-21
1005|jay|d2|bang|1988-03-22
1006|naveen|d4|hyd|1986-04-12
1007|naser|d1|hyd|1989-11-15
1008|rahul|d3|delhi|1990-12-23

[cloudera@localhost ~]$ gedit d.txt
d1|research|A-block
d2|sales|A-block
d3|testing|B-block
d4|development|C-block

[cloudera@localhost ~]$ hadoop fs -put emp.txt /user/cloudera/batch3

[cloudera@localhost ~]$ hadoop fs -put d.txt /user/cloudera/batch3

CONNECT TO HIVE

[cloudera@localhost ~]$ hive

hive>

TO CREATE DATABSE

hive> create database test;
(OR)
hive> create database if not exists test;

TO LIST OUT DATABSES

hive> show databases;

TO DROP DATABSE

hive> drop database test;
(OR)
hive> drop database if exists test;
(OR)
hive> drop database if exists test cascade;

TO MAKE USE OF THE DATABASE

hive> use test;

TO CREATE TABLE

hive> create table emp(id int,name string,dept string,place string,dob string)
>comment ‘this is employee table’
> row format delimited fields terminated by '|' lines terminated by '\n'
>stored as textfile;

(OR) Type IN Single Line

hive> create table emp(id int,name string,dept string,place string,dob string) comment ‘this is employee table’ row format delimited fields terminated by '|' lines terminated by '\n' stored as textfile;

hive> create table department(did string,dname string,block string) comment 'this is department table' row format delimited fields terminated by '|' lines terminated by '\n' stored as textfile;

TO SEE THE LIST OF TABLES

hive> show tables;

TO SEE THE STRUCTURE OF A TABLE

hive> describe emp;

TO SEE THE STRUCTURE & METADATA INFORMATION OF TABLE

```
hive> describe formatted emp;
hive> show create table emp;
```

TO RENAME TABLE NAME

```
hive> alter table department rename to d;
hive> show tables;
```

TO ADD ONE OR MORE COLUMNS TO THE TABLE

```
hive> alter table d add columns (estb_year int,rating smallint);
hive> describe d;
```

TO CHANGE COLUMN NAME OR ITS DATATYPE OR BOTH

```
hive> alter table d change rating rate string;
hive> describe d;

hive> alter table d change rate rate bigint;
hive> describe d;
```

TO REPLACE COLUMNS

```
hive> alter table d replace columns (did string,dname string, block string);
hive> describe d;

hive> alter table d replace columns (block string);
hive> describe d;

hive> select * from d;

hive> alter table d replace columns (did string,dname string, block string);
hive> desc d;
hive> select * from d;
```

TO DROP THE TABLE

```
hive> drop table if exists d;
(OR)
hive> drop table d;
```

Experiment 15

Load, Insert of data

TO LOAD FROM LOCAL FILESYSTEM

```
hive> load data local inpath '/home/cloudera/emp.txt' into table emp;
hive> select * from emp;
```

TO LOAD FROM HADOOP FILE SYSTEM

```
hive> load data inpath '/user/cloudera/emp.txt' into table emp;
hive> select * from emp;
```

TO LOAD USING OVERWRITE KEYWORD

```
hive> load data local inpath '/home/cloudera/emp.txt' overwrite into table emp;
hive> select * from emp;

hive> load data local inpath '/home/cloudera/d.txt' overwrite into table department;
hive> select * from department;
```

Experiment 16

BUILT-IN RELATIONAL OPERATORS

A=B, A<B, A<=B, A>B, A>=B, A IS NULL, A IS NOT NULL, A LIKE B, A!=B

BUILT-IN ARTHEMATIC OPERATORS

A+B, A-B, A*B ,A/B, A%B,A&B, A|B, A^B, ~A
A&B : bitwise and operationA|B:
bitwise or operation
A^B;bitwise XOR operation
~A: bitwise not operation

BUILT_IN LOGICAL OPERATORS

A AND B, A OR B, NOT A, A || B, A&&B, NOT A, !A

Q) Display details of employee whose employee id is greater than and equal to 1003 and doesn't come from hyd.

```
hive> select * from emp where id >=1003 and place !='hyd';
                        OR
hive> select * from emp where id >=1003 and place not in ('hyd');
```

Q) Display details of department whose id is less than d2 or department name is development.

```
hive> select * from department where did<'d2' or dname='development';
(OR)
hive> select * from department where did<'d2' or dname like 'development';(OR)
hive> select * from department where did<'d2' or dname like 'd%';
```

Q) Display details of department whose department name's second letter is 'e'.

```
hive> select * from department where did<'d2' or dname like '_e%';
```

Q) Display total no:of employees ,minimum of their employee no, avg of their employee no,max of their employee, sum of their employee from employee dataset.

hive> select count(*),min(id),avg(id),max(id),sum(id) from emp;

Q) Display department name,count of employees in each department

hive> select count(*),dept from emp group by dept;

Q) Display department name,count of employees in each department and display rows those have count more than 2.

hive> select count(*),dept from emp group by dept having count(*)>2;

Q) Display department name,count of employees in each department in descending order of count.

hive> select count(*) as c,dept from emp group by dept order by c desc;

Q) Display department name,count of employees in each department in descending order of count and display only first two rows.

hive> select count(*) as c,dept from emp group by dept order by c desc limit 2;

TO QUIT FROM HIVE

hive> quit;

MODULE 4
HBASE

TO START HBASE
[cloudera@localhost ~]$ hbase shell

GENERAL SHELL COMMANDS

hbase(main):001:0> status
1 servers, 0 dead, 2.0000 average load

hbase(main):002:0> version
0.94.6-cdh4.4.0, rUnknown, Tue Sep 3 20:09:51 PDT 2013

hbase(main):003:0> whoami
cloudera (auth:SIMPLE)

hbase(main):004:0> table_help
Help for table-reference commands.

Experiment 17

HBase- CREATE TABLE & VERIFY

hbase(main):007:0> create 'emp','personal data','professional data'
hbase(main):008:0> describe 'emp'

To Verify table
hbase(main):008:0> list
TABLE
emp
hbase(main):014:0> exist 'emp'
Table emp does exist

Experiment 18

HBase- STORE DATA IN TABLE

- **Insertion of data to first row**

```
hbase(main):009:0> put 'emp','row1','personal data:name','raju'
hbase(main):010:0> put 'emp','row1','personal data:city','hyd'
hbase(main):011:0> put 'emp','row1','professional data:designation','manager'
hbase(main):012:0> put 'emp','row1','professional data:salary','50000'
hbase(main):013:0> scan 'emp'
```

- **Insertion of data to second row**

```
hbase(main):013:0> put 'emp','row2','personal data:name','milind'
hbase(main):018:0> put 'emp','row2','personal data:city','chennai'
hbase(main):020:0> put 'emp','row2','professional data:designation','soft Engineer'
hbase(main):015:0> put 'emp','row2','professional data:salary','30000'
```

- **Insertion of data to third row**

```
hbase(main):022:0> put 'emp','row3','personal data:name','anita'
hbase(main):023:0> put 'emp','row3','personal data:city','delhi'
hbase(main):024:0> put 'emp','row3','professional data:designation','tester'
hbase(main):025:0> put 'emp','row3','professional data:salary','40000'
```

Experiment 19

TO READ THE DATA WHICH HAS BEEN ENTERED

SCAN command reads all rows of that table

hbase(main):026:0> scan 'emp'

```
ROW                          COLUMN+CELL
 row1                        column=personal data:city,
timestamp=1523426980414, value=hyd
 row1                        column=personal data:name,
timestamp=1523426797711, value=raju
 row1                        column=professional data:designation,
timestamp=1523426835511, value=manager
 row1                        column=professional data:salary,
timestamp=1523426852045, value=50000
 row2                        column=personal data:city,
timestamp=1523427018284, value=chennai
 row2                        column=personal data:name,
timestamp=1523426910414, value=milind
 row2                        column=professional data:designation,
timestamp=1523427061723, value=soft Engineer
 row2                        column=professional data:salary,
timestamp=1523426965356, value=30000
 row3                        column=personal data:city,
timestamp=1523427129733, value=delhi
 row3                        column=personal data:name,
timestamp=1523427115249, value=anita
 row3                        column=professional data:designation,
timestamp=1523427162000, value=tester
 row3                        column=professional data:salary,
timestamp=1523427187838, value=40000
        3    row(s) in 0.0190 seconds
```

Experiment 20

GET command is used to read data of a particular row or specific column

```
hbase(main):027:0> get 'emp','row3'
COLUMN                          CELL
 personal data:city             timestamp=1523427129733, value=delhi
 personal data:name             timestamp=1523427115249, value=anita
 professional data:designation      timestamp=1523427162000, value=tester
 professional data:salary           timestamp=1523427187838, value=40000

hbase(main):004:0> get 'emp','row3',{COLUMN=>'personal data:city'}
COLUMN                          CELL
 personal data:city             timestamp=1523427129733, value=delhi


hbase(main):020:0> list
TABLE
emp
example
expert
```

TO UPDATE/MODIFY THE EXISTING DATA

```
hbase(main):008:0> put 'emp','row3','personal data:city','2bang'
hbase(main):009:0> put 'emp','row3','personal data:city','3kochi'
hbase(main):010:0> get 'emp','row3',{COLUMN=>'personal
data:city',VERSIONS=>3}

COLUMN                          CELL
 personal data:city             timestamp=1523428351922, value=3kochi
 personal data:city             timestamp=1523428337355,  value=2bang
 personal data:city             timestamp=1523427129733, value=delhi

hbase(main):011:0> put 'emp','row3','personal data:city','4dehradun'
hbase(main):012:0> get 'emp','row3',{COLUMN=>'personal
data:city',VERSIONS=>3}
```

COLUMN	CELL
personal data:city	timestamp=1523428391129, value=4dehradun
personal data:city	timestamp=1523428351922, value=3kochi
personal data:city	timestamp=1523428337355, value=2bang

hbase(main):013:0> get 'emp','row3',{COLUMN=>'personal data:city',VERSIONS=>2}

COLUMN	CELL
personal data:city	timestamp=1523428391129, value=4dehradun
personal data:city	timestamp=1523428351922, value=3kochi

TO DISABLE THE TABLE

hbase(main):024:0> disable 'expert'
hbase(main):020:0> list
TABLE
emp
xample
expert

hbase(main):027:0> scan 'expert'
ROW COLUMN+CELL
ERROR: org.apache.hadoop.hbase.DoNotRetryIOException: expert is disabled.

TO ALTER TABLE to add additional column family ‘xyz with version 5’
hbase(main):017:0> disable 'emp'
hbase(main):018:0> alter 'emp',NAME=>'xyz',VERSIONS=>5
hbase(main):020:0> enable 'emp'
hbase(main):022:0> describe 'emp'
DESCRIPTION
ENABLED
{NAME => 'personal data', DATA_BLOCK_ENCODING => 'NONE', BLOOMFI
LTER => 'NONE', REPLICATION_SCOPE => '0', VERSIONS => '3',
COMPRESSION => 'NONE', MIN_VERSIONS => '0

', TTL => '2147483647', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY => 'false', E
NCODE_ON_DISK => 'true', BLOCKCACHE => 'true'}, {NAME => 'professional data', DATA_BLOCK_ENCODING =>
'NONE', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', MI
N_VERSIONS => '0', TTL => '2147483647', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', IN_MEMO
RY => 'false', ENCODE_ON_DISK => 'true', BLOCKCACHE => 'true'},
{NAME => 'emp', FAMILIES =>[{NAME => 'xyz', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NON true
E', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '5', TTL => '2147483647', MIN_VERSI
ONS => '0', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', ENCODE_ON_DISK => 'true', IN_MEMORY
=> 'false', BLOCKCACHE => 'true'},]}

hbase(main):023:0> put 'emp','row3','xyz:city','5mumbai'
hbase(main):024:0> put 'emp','row3','xyz:city','6jodhpur'
hbase(main):028:0> put 'emp','row3',‘xyz:city','7hubli'
hbase(main):029:0> put 'emp','row3',‘xyz:city','8london'
hbase(main):030:0> put 'emp','row3',‘xyz:city','9shimla'
hbase(main):031:0> put 'emp','row3',‘xyz:city','10kanpur'
hbase(main):032:0> put 'emp','row3',’xyz:city','11nagpur'

hbase(main):033:0> get 'emp','row3',{COLUMN=>'xyz:city',VERSIONS=>5}
COLUMN CELL
xyz:city timestamp=1523431975773, value=11nagpur
xyz:city timestamp=1523431968306, value=10kanpur
xyz:city timestamp=1523431961690, value=9shimla
xyz:city timestamp=1523431953548, value=8london
xyz:city timestamp=1523431947056, value=7hubli
5 row(s) in 0.0080 seconds

hbase(main):034:0> scan 'emp'
ROW COLUMN+CELL
row1 column=personal data:city, timestamp=1523426980414, value=hyd
row1 column=personal data:name, timestamp=1523426797711, value=raju

row1 column=professional data:designation, timestamp=1523426835511, value=manager
row1 column=professional data:salary, timestamp=1523426852045, value=50000
row2 column=personal data:city, timestamp=1523427018284, value=chennai
row2 column=personal data:name, timestamp=1523426910414, value=milind
row2 column=professional data:designation, timestamp=1523427061723, value=soft Engineer
row2 column=professional data:salary, timestamp=1523426965356, value=30000

Experiment 21

TO DELETE A PARTICULAR COLUMN FAMILY

```
hbase(main):037:0> disable 'emp'
hbase(main):038:0> alter 'emp','delete'=>'xyz'
hbase(main):039:0> enable 'emp'
hbase(main):042:0> get 'emp','row3'

COLUMN                          CELL
 personal data:city             timestamp=1523428391129, value=4dehradun
 personal data:name             timestamp=1523427115249, value=anita
 professional data:designation  timestamp=1523427162000, value=tester
 professional data:salary       timestamp=1523427187838, value=40000
4 row(s) in 0.0080 seconds
```

A SPECIFIC CELL IN TABLE

```
hbase(main):045:0> delete 'emp','row3','personal data:name'
hbase(main):046:0> get 'emp','row3'
COLUMN                          CELL
 personal data:city             timestamp=1523428391129, value=4dehradun
 professional data:designation  timestamp=1523427162000,  value=tester
 professional data:salary       timestamp=1523427187838, value=40000
3 row(s) in 0.1870 seconds
```

DELETE COMPLETE ROW IN TABLE

```
hbase(main):049:0> deleteall 'emp','row3'
hbase(main):050:0> get 'emp','row3'
COLUMN                          CELL
0 row(s) in 0.0340 seconds

hbase(main):051:0> scan 'emp'
ROW                             COLUMN+CELL
 row1                           column=personal data:city,
timestamp=1523426980414, value=hyd
```

```
 row1                    column=personal data:name,
timestamp=1523426797711, value=raju
 row1                    column=professional data:designation,
timestamp=1523426835511, value=manager
 row1                    column=professional data:salary,
timestamp=1523426852045, value=50000
 row2                    column=personal data:city,
timestamp=1523427018284, value=chennai
 row2                    column=personal data:name,
timestamp=1523426910414, value=milind
 row2                    column=professional data:designation,
timestamp=1523427061723, value=soft Engineer
 row2                    column=professional data:salary,
timestamp=1523426965356, value=30000
2 row(s) in 0.0390 seconds
```

Experiment 22

TO DROP THE TABLE

To drop , first table has to be disabled.

hbase(main):005:0> list
TABLE
emp
example
exp
expert

hbase(main):024:0> disable 'expert'

hbase(main):001:0> drop 'expert'

hbase(main):002:0> list

TABLE
emp
example

To drop all three table which starts with name ‘ex’

hbase(main):006:0> disable_all 'ex.*'

example
exp
expert

Disable the above 3 tables (y/n)?
y
3 tables successfully disabled

hbase(main):007:0> drop_all 'ex.*'

example
exp

expert

Drop the above 3 tables (y/n)?
y
3 tables successfully dropped

hbase(main):052:0> count 'emp'

2 row(s) in 0.7080 seconds