BIG DATA

LABORATORY MANUAL

Department of Computer Science and Engineering
GITAM School of Technology
GITAM Deemed to be University
Hyderabad Campus - 502329
Academic Year 2022-23

Sno.	Name of the Experiment
1.	Installation of Hadoop in Linux and Windows
2.	File Management Tasks in Hadoop
3.	Case Study on Hadoop Commands
4.	Map Reduce using Python (word count)
5.	Analyzing Data with Pig Latin Commands a) Loading and Storing commands b) Diagnostic Operators c) Grouping and Joining commands
6.	Analyzing Data with Pig Latin Commands a) Combining and Splitting b) Sorting commands c) Filtering commands
7.	a) Map Reduce using Pig Latin (word count) b) Pig Latin String functions
8.	Hive Commands
9.	Problems based on Hive Commands
10.	a) Spark Commands b) Map Reduce using Spark (word count)

1.1 Hadoop Installation (Linux)

Prerequisite Test

==========

sudo apt update sudo apt install

openjdk-8-jdk -y

java -version; javac -version sudo apt install

openssh-server openssh-client -y sudo adduser

hdoop su - hdoop ssh-keygen -t rsa -P '' -f

~/.ssh/id_rsa cat ~/.ssh/id_rsa.pub >>

~/.ssh/authorized_keys chmod 0600

~/.ssh/authorized_keys ssh localhost

Downloading Hadoop (Please note link is updated to new version of hadoop here on 6th May 2022)

==========

wget https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz tar

xzf hadoop-3.2.3.tar.gz

Editng 6 important files

==========

1st file

==========

sudo nano .bashrc - here you might face issue saying hdoop is not sudo user if this issue comes then su - aman sudo adduser hdoop sudo

sudo nano .bashrc

#Add below lines in this file

#Hadoop Related Options export

HADOOP_HOME=/home/hdoop/hadoop-3.2.3 export

HADOOP_INSTALL=$HADOOP_HOME export

HADOOP_MAPRED_HOME=$HADOOP_HOME export

HADOOP_COMMON_HOME=$HADOOP_HOME export

HADOOP_HDFS_HOME=$HADOOP_HOME export

YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_OPTS"-Djava.library.path=$HADOOP_HOME/lib/nativ"

source ~/.bashrc

2nd File

========== sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh

#Add below line in this file in the end

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

3rd File

========== sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml #Add below lines in this file(between "<configuration>" and "<"/configuration>")

```
  <property>
      <name>hadoop.tmp.dir</name>
      <value>/home/hdoop/tmpdata</value>
      <description>A base for other temporary directories.</description>
   </property>
   <property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
```

<description>The name of the default file system></description>

</property>

4th File

========== sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

```
<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hdoop/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
```

5th File

==========

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

6th File

========== sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

#Add below lines in this file(between "<configuration>" and "<"/configuration>")

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>
<value>127.0.0.1</value>

</property>

<property>

<name>yarn.acl.enable</name>

<value>0</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>

</property>

Launching Hadoop

========== hdfs

namenode -format

./start-dfs.sh

Reference: https://www.youtube.com/watch?v=Ih5cuJYYz6Y

1.2 Hadoop Installation (Windows) Cloudera

Quick Start VM Installation

Step1:

Download & Install VM Ware Workstation Player

Step2:

Download Cloudera quick start VM

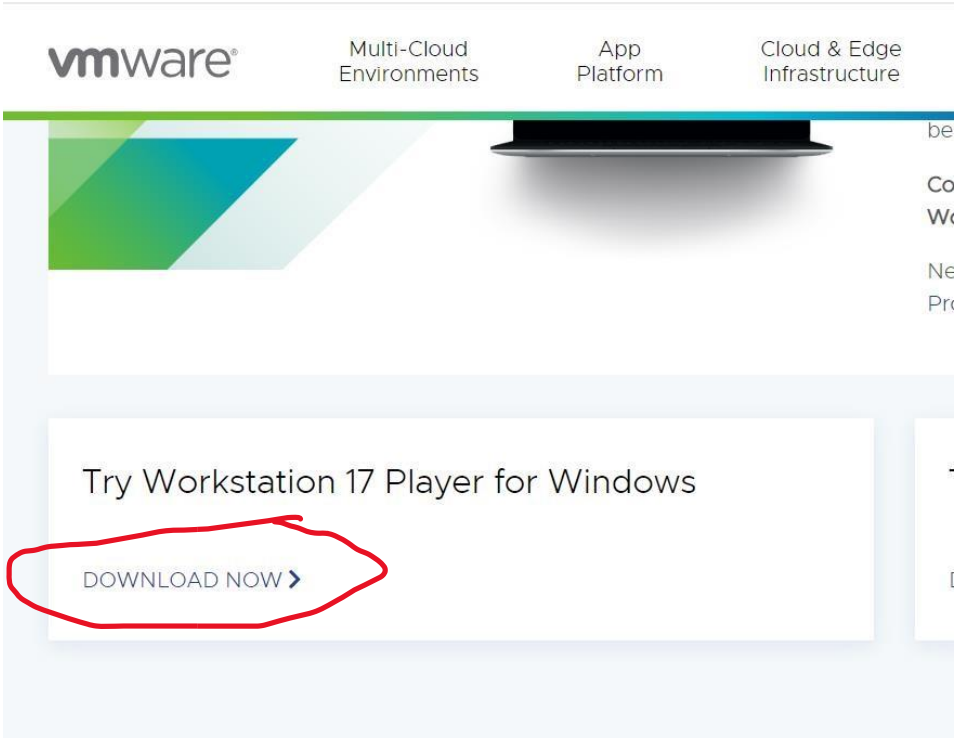
Step3:

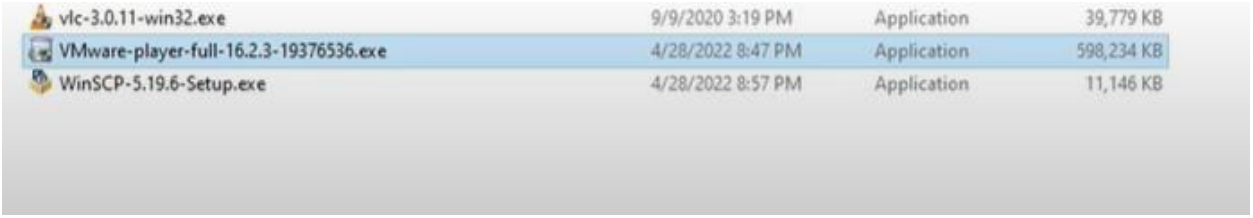
Install Cloudera quick start VM on VM Ware workstation

1) Download & Install VM Ware Workstation Player

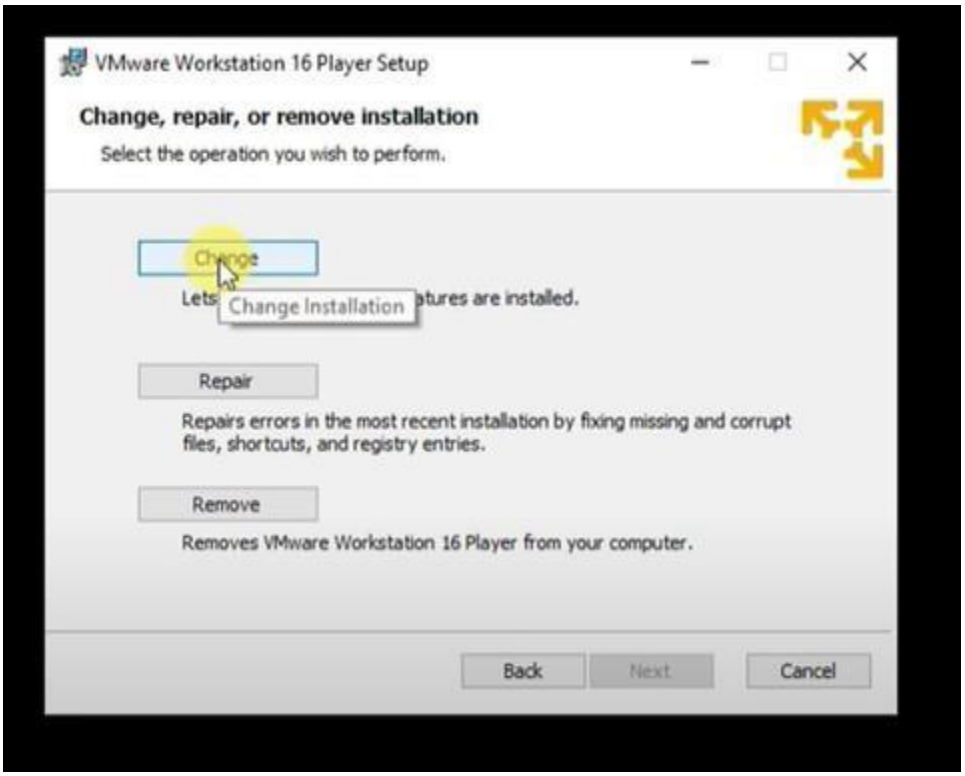
Download VM Ware from the given link

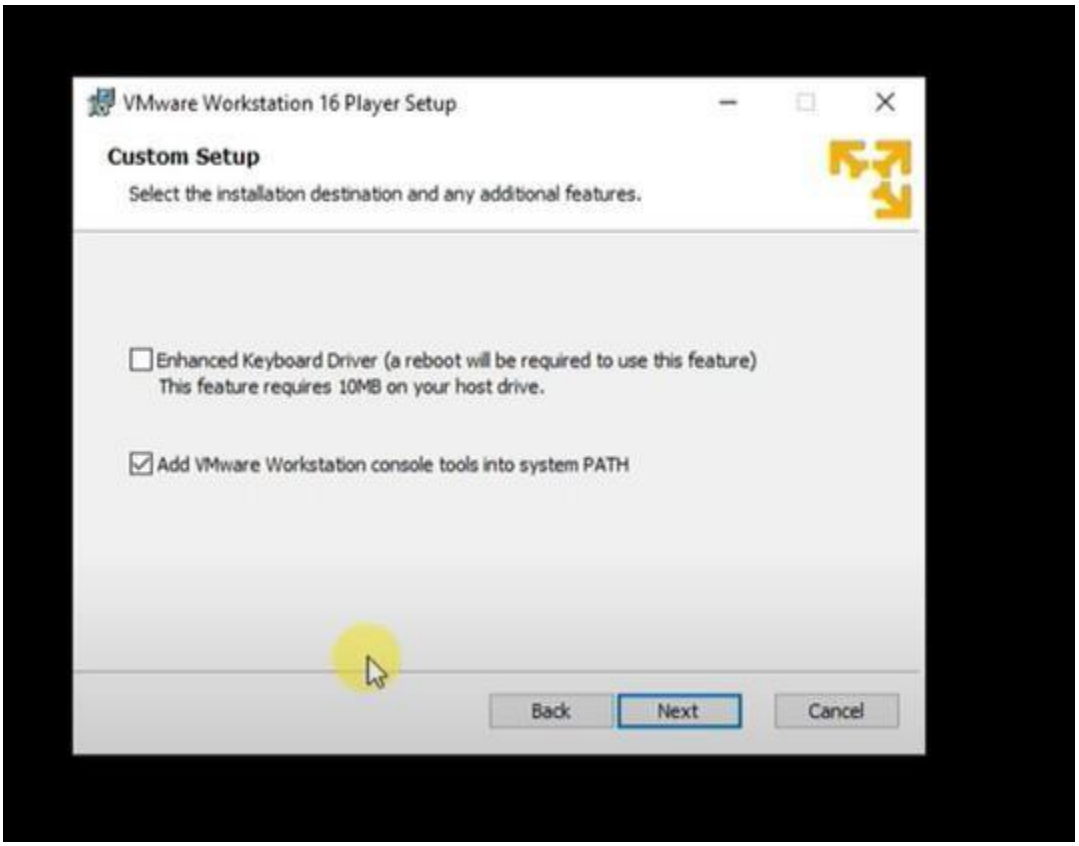
https://www.vmware.com/in/products/workstationplayer/workstation-player-evaluation.html

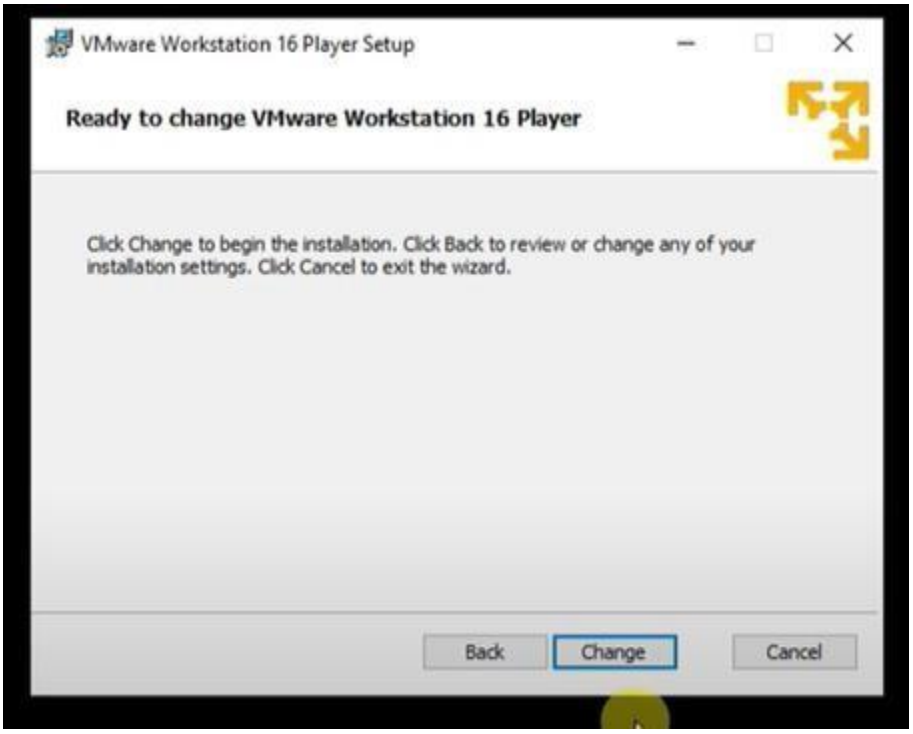

vmware
Multi-Cloud Environments
App Platform
Cloud & Edge Infrastructure
Try Workstation 17 Player for Windows
DOWNLOAD NOW


vlc-3.0.11-win32.exe 9/9/2020 3:19 PM Application 39,779 KB
VMware-player-full-16.2.3-19376536.exe 4/28/2022 8:47 PM Application 598,234 KB
WinSCP-5.19.6-Setup.exe 4/28/2022 8:57 PM Application 11,146 KB


VMware Workstation 16 Player Setup
VMWARE
WORKSTATION
PLAYER™ 16
Welcome to the VMware Workstation 16 Player Setup Wizard
The Setup Wizard allows you to change the way VMware Workstation 16 Player features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel


VMware Workstation 16 Player Setup
Change, repair, or remove installation
Select the operation you wish to perform.
Change
Change Installation
Repair
Repairs errors in the most recent installation by fixing missing and corrupt files, shortcuts, and registry entries.
Remove
Removes VMware Workstation 16 Player from your computer.
Back
Next
Cancel

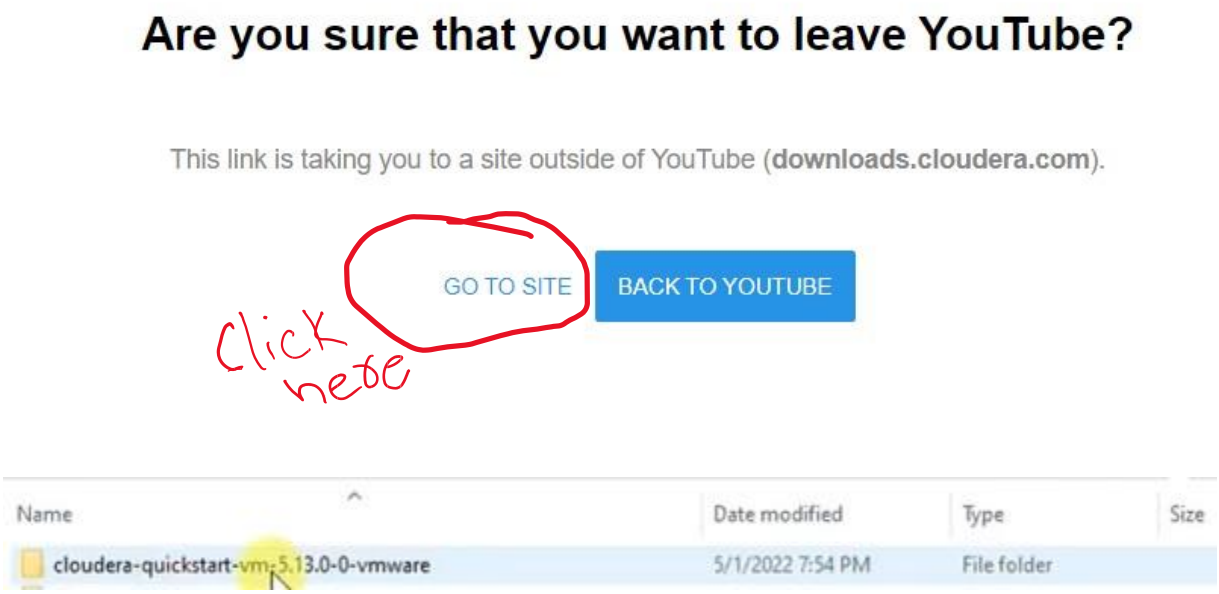

VMware Workstation 16 Player Setup
Custom Setup
Select the installation destination and any additional features.
Enhanced Keyboard Driver (a reboot will be required to use this feature)
This feature requires 10MB on your host drive.
Add VMware Workstation console tools into system PATH
Back
Next
Cancel



Step2:

Download Cloudera quick start VM

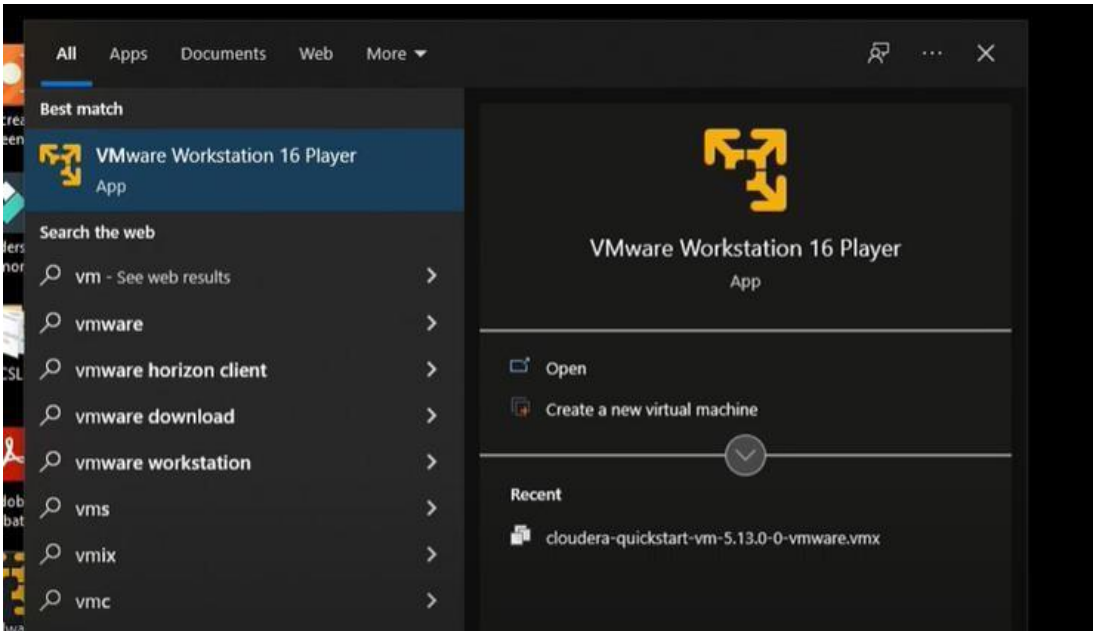
https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa2w5bUc2YkNzNHhJZnN2cjJPUHl2blRBWnFDZ3xBQ3Jtc0tsNms5X2ZEU3Blb0dERWdEa1h0NDRiZEMxYXdOOS1DLUlxNGo0NEhNcVFGcGpTY1djNmtzTFh5dEF0YjJKSnV0aDBvbzlFMjRMWXB4LWk5aldreTN1MkhJd1MyWkRIZU1xVmJWQ1lsTDlOc2R5NG1IUQ&q=https%3A%2F%2Fdownloads.cloudera.com%2Fdemo_vm%2Fvmware%2Fcloudera-quickstart-vm5.13.0-0-vmware.zip&v=0LmLMur_MSo

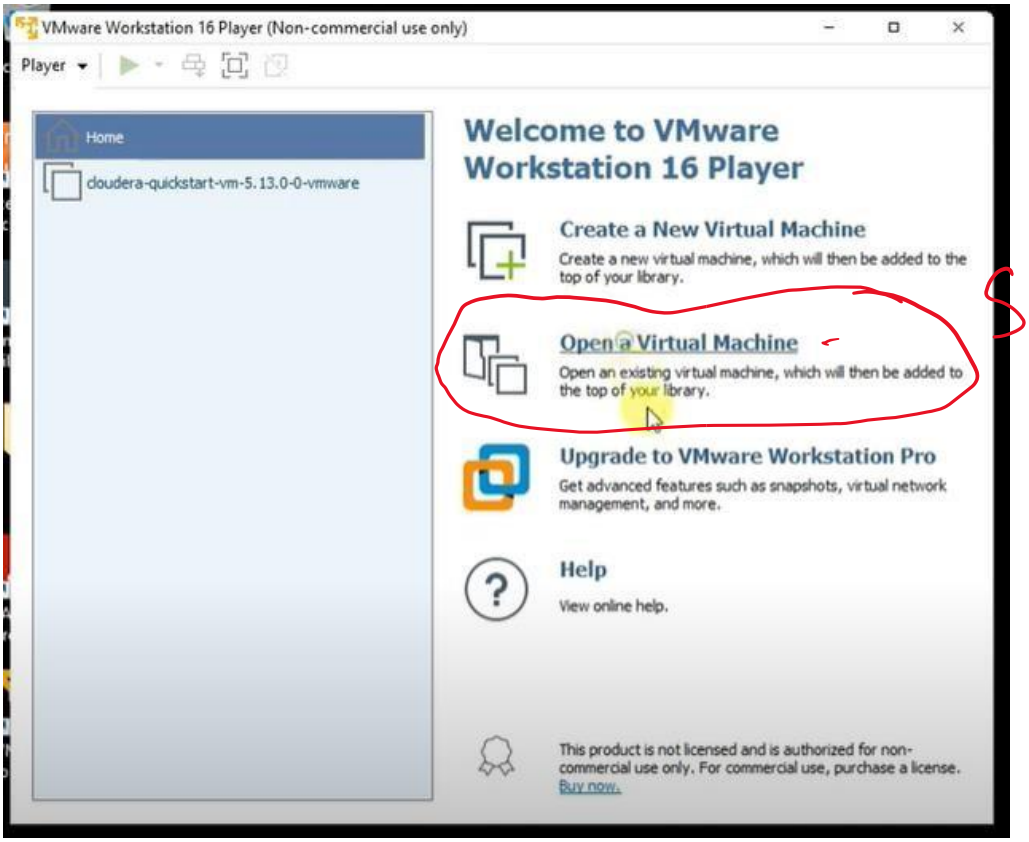


Extract file by using 7-zip or winrar

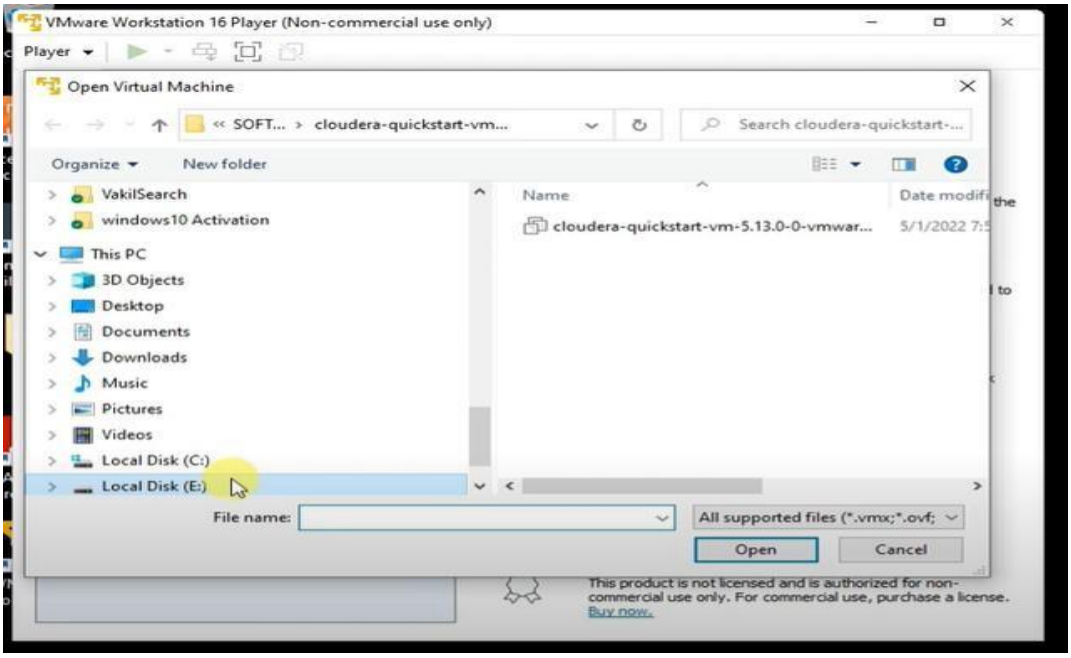
Step3:

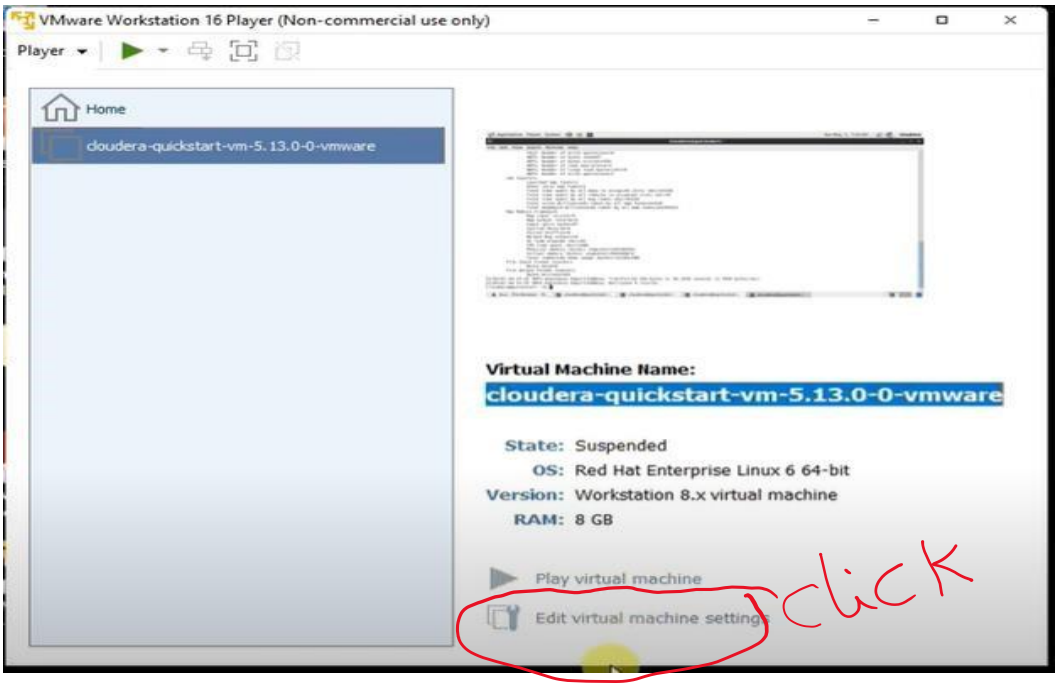
Install Cloudera quick start VM on VM Ware workstation

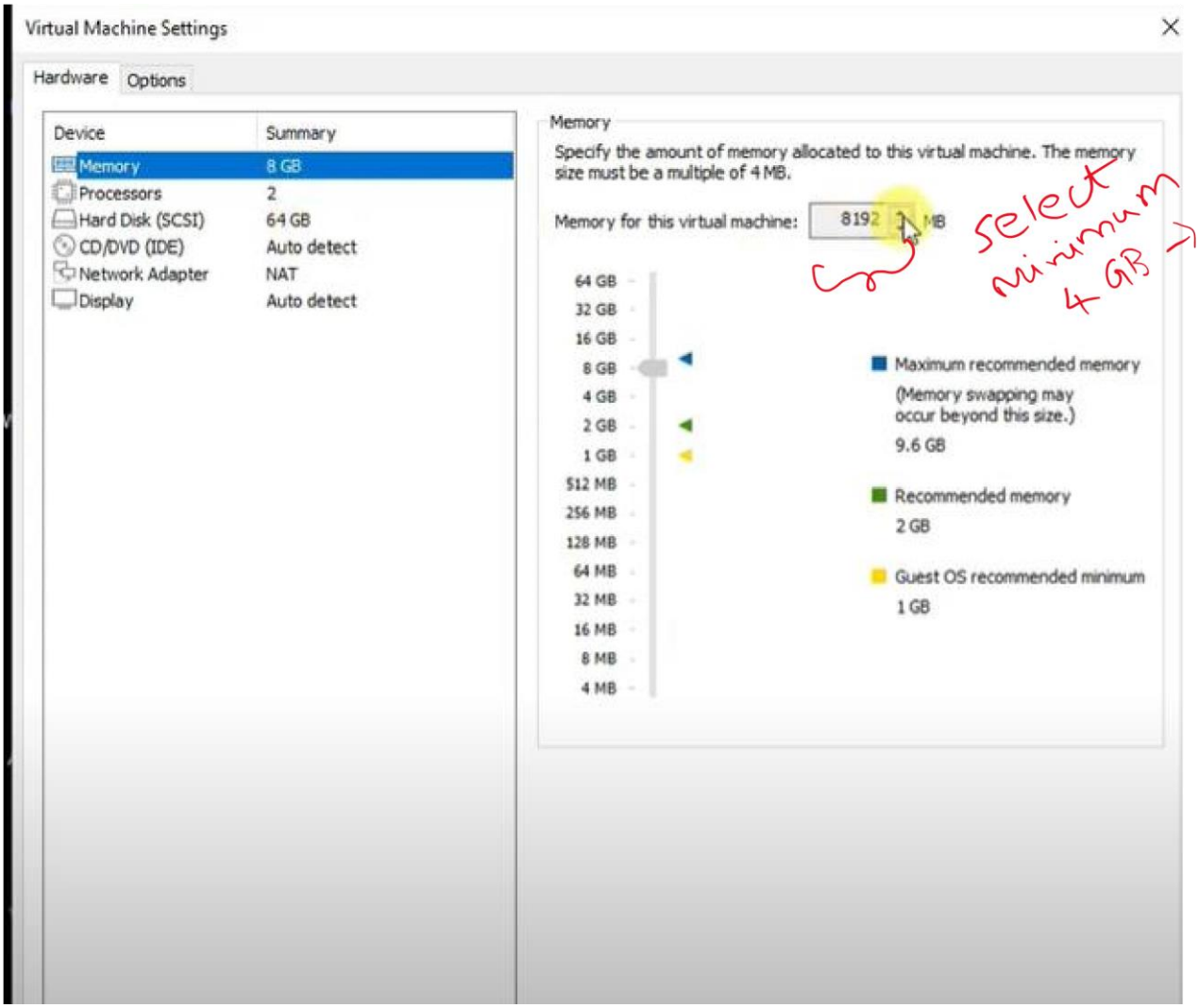

All
Apps
Documents
Web
More
Best match
VMware Workstation 16 Player
App
Search the web
vm - See web results
vmware
vmware horizon client
vmware download
vmware workstation
vms
vmix
vmc
VMware Workstation 16 Player
App
Open
Create a new virtual machine
Recent
cloudera-quickstart-vm-5.13.0-0-vmware.vmx

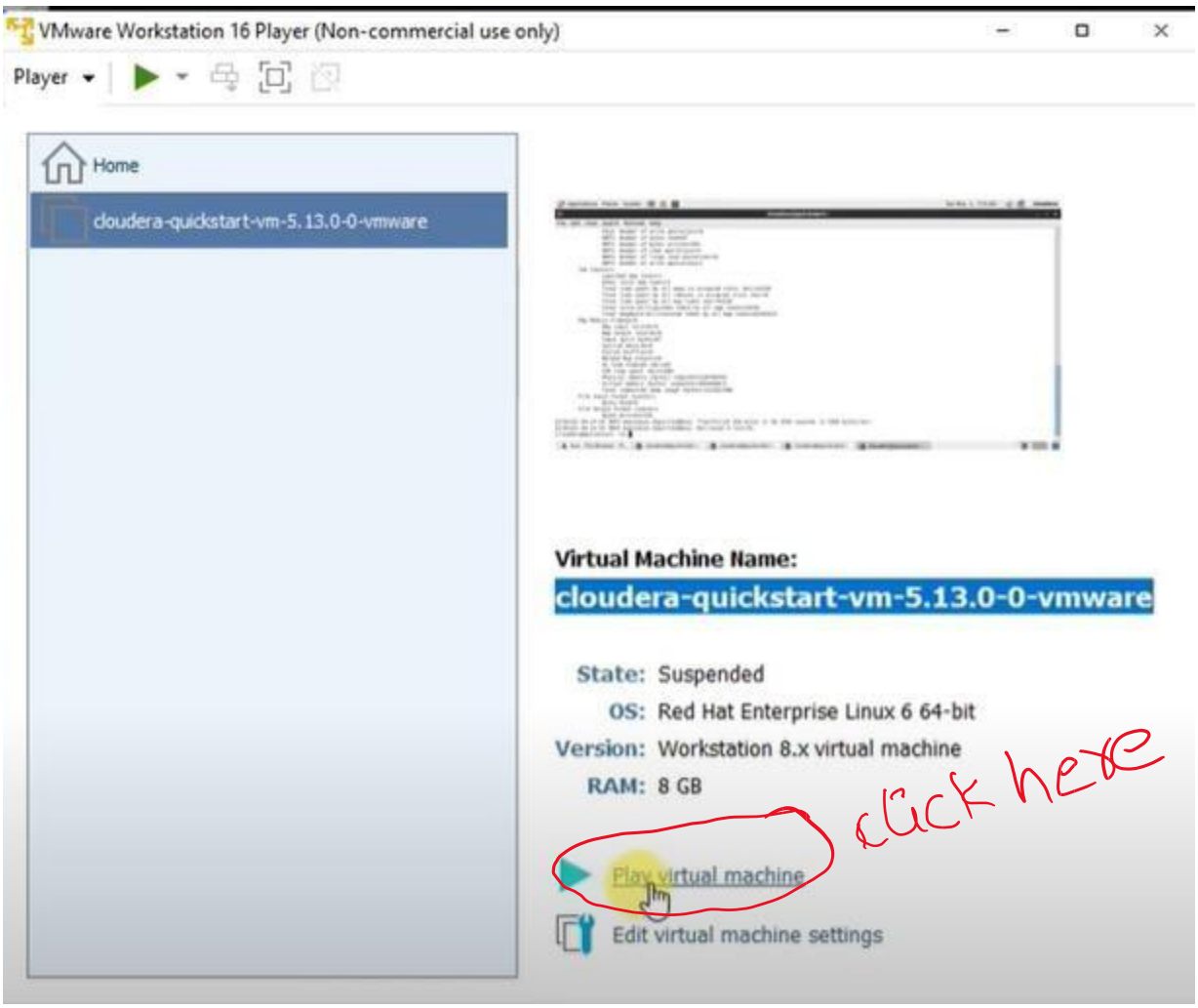

VMware Workstation 16 Player (Non-commercial use only)
Player
Home
cloudera-quickstart-vm-5.13.0-0-vmware
Welcome to VMware Workstation 16 Player
Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.
Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.
Upgrade to VMware Workstation Pro
Get advanced features such as snapshots, virtual network management, and more.
Help
View online help.
This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license.
Buy now.

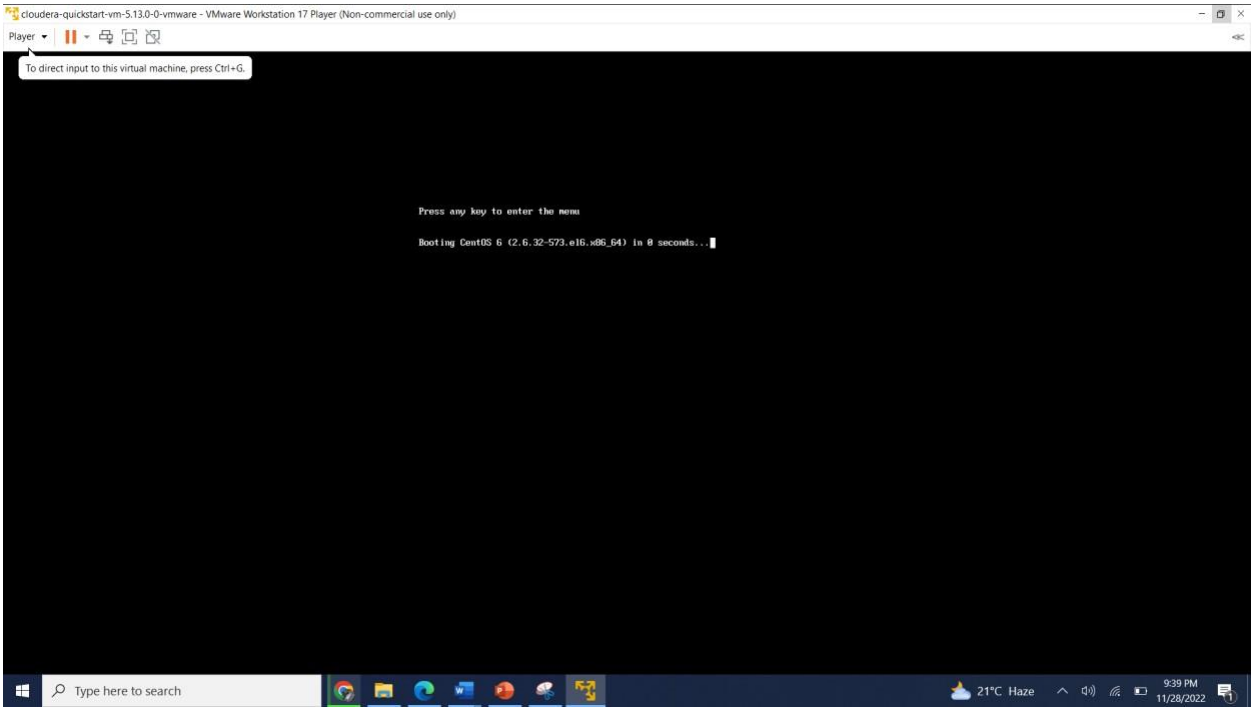
Select

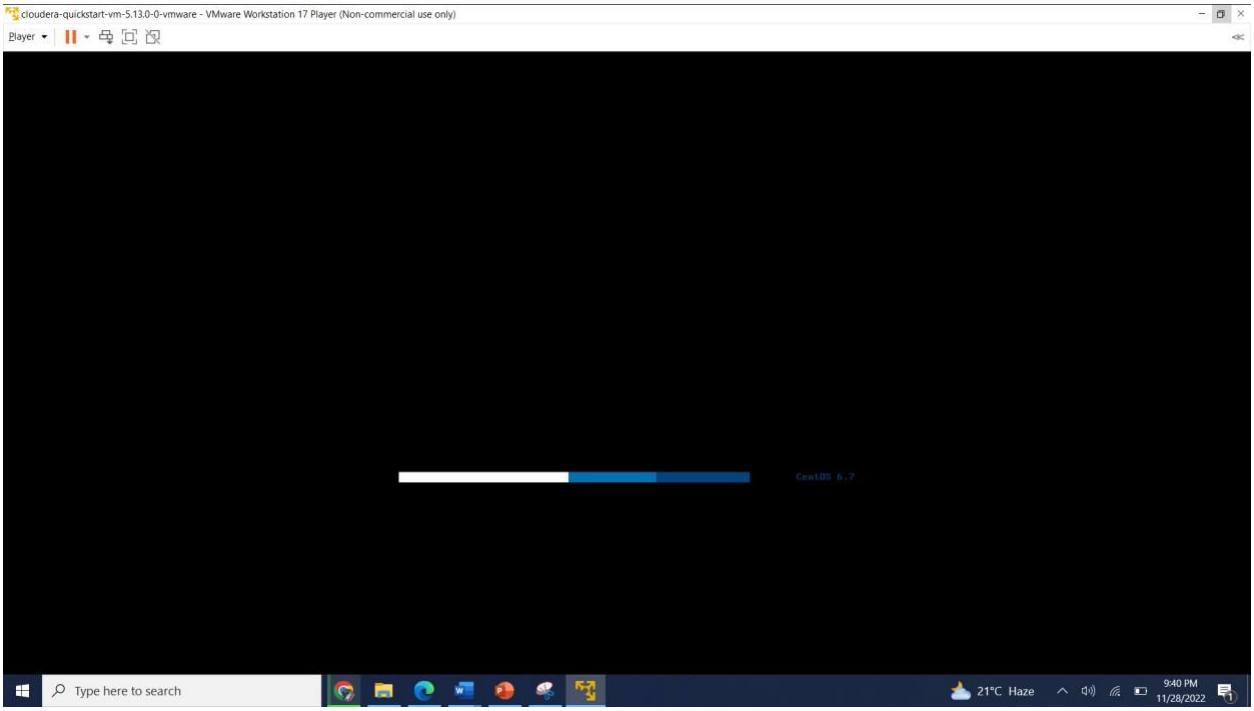

VMware Workstation 16 Player (Non-commercial use only)
Player
Open Virtual Machine
« SOFT... › cloudera-quickstart-vm...
Search cloudera-quickstart-...
Organize
New folder
VakilSearch
windows10 Activation
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (E:)
Name
Date modifi
cloudera-quickstart-vm-5.13.0-0-vmwar...
5/1/2022 7:5
File name:
All supported files (*.vmx;*.ovf;
Open
Cancel
This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license.
Buy now.


VMware Workstation 16 Player (Non-commercial use only)
Player
Home
cloudera-quickstart-vm-5.13.0-0-vmware
Virtual Machine Name:
cloudera-quickstart-vm-5.13.0-0-vmware
State: Suspended
OS: Red Hat Enterprise Linux 6 64-bit
Version: Workstation 8.x virtual machine
RAM: 8 GB
Play virtual machine
Edit virtual machine settings
click

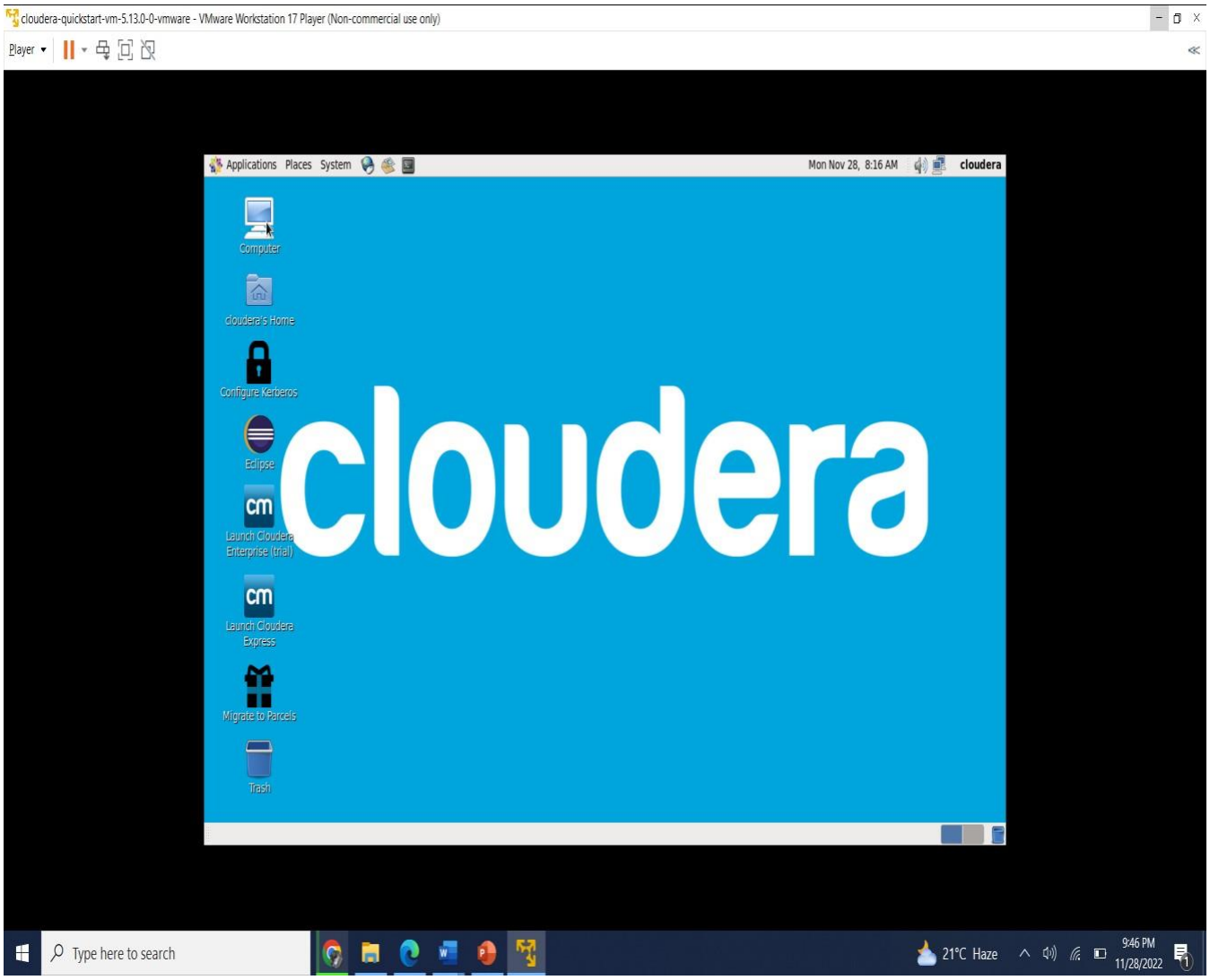

Virtual Machine Settings
Hardware
Options
Device
Summary
Memory
8 GB
Processors
2
Hard Disk (SCSI)
64 GB
CD/DVD (IDE)
Auto detect
Network Adapter
NAT
Display
Auto detect
Memory
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.
Memory for this virtual machine:
8192
MB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
1 GB
512 MB
256 MB
128 MB
64 MB
32 MB
16 MB
8 MB
4 MB
Maximum recommended memory
(Memory swapping may occur beyond this size.)
9.6 GB
Recommended memory
2 GB
Guest OS recommended minimum
1 GB
select minimum 4 GB => 8 GB or 16GB


VMware Workstation 16 Player (Non-commercial use only)
Player
Home
cloudera-quickstart-vm-5.13.0-0-vmware
Virtual Machine Name:
cloudera-quickstart-vm-5.13.0-0-vmware
State: Suspended
OS: Red Hat Enterprise Linux 6 64-bit
Version: Workstation 8.x virtual machine
RAM: 8 GB
click here
Play virtual machine
Edit virtual machine settings



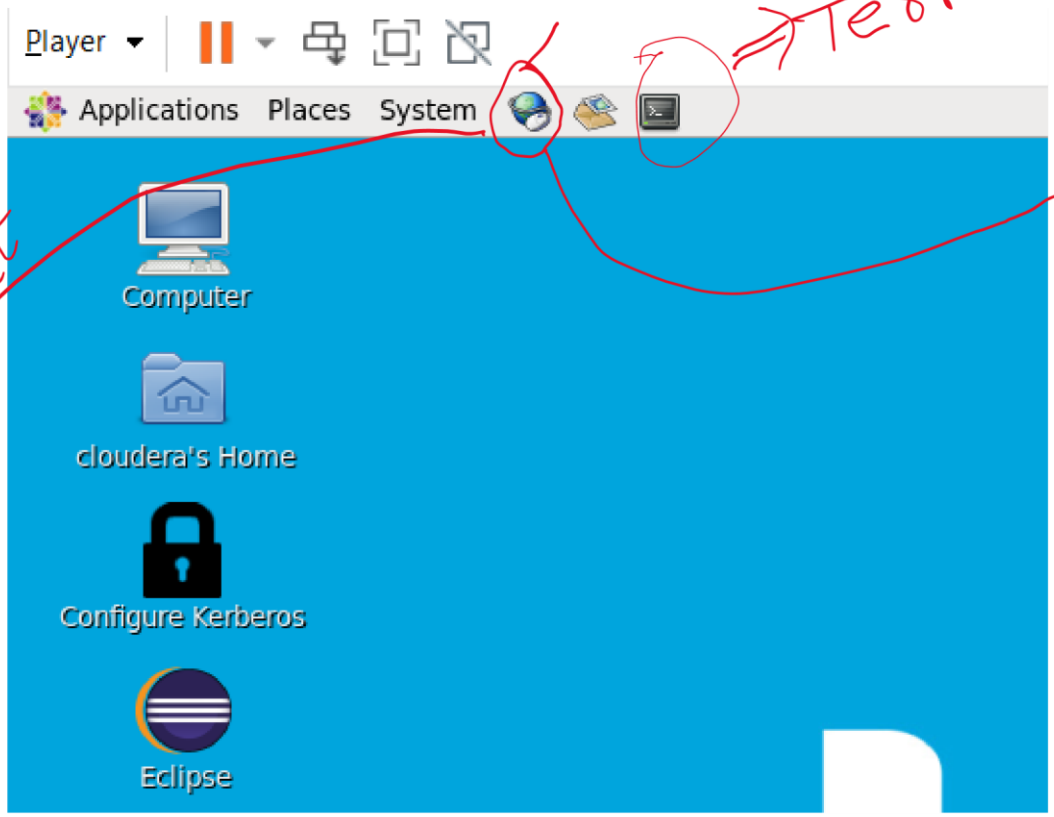


(NOTE: It takes time to load-depends on system performance)



(Desktop View)

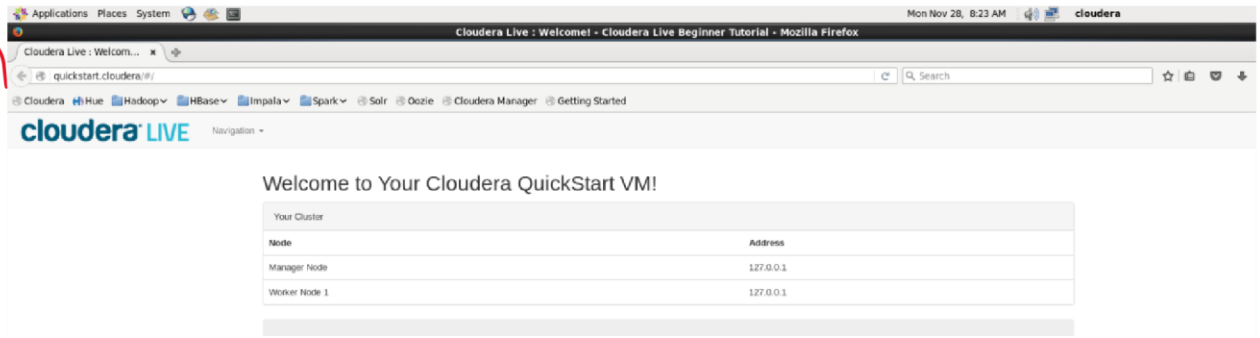
Cloudera Exploration:



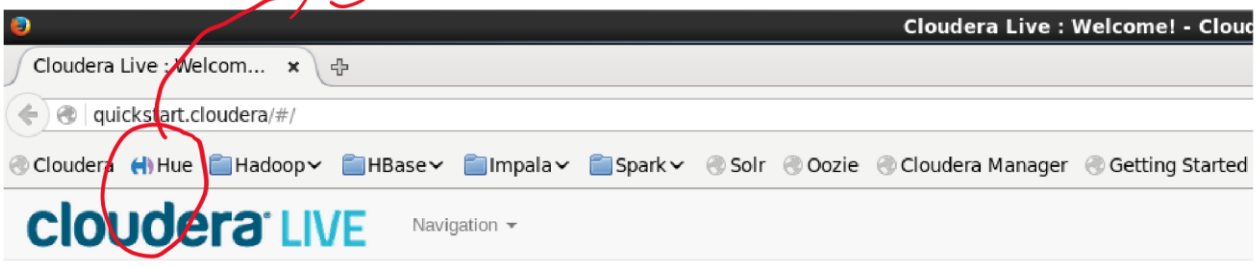
Terminal

Browsing & To open Big Data Technologies Like PIG HIVE, SPARK etc..

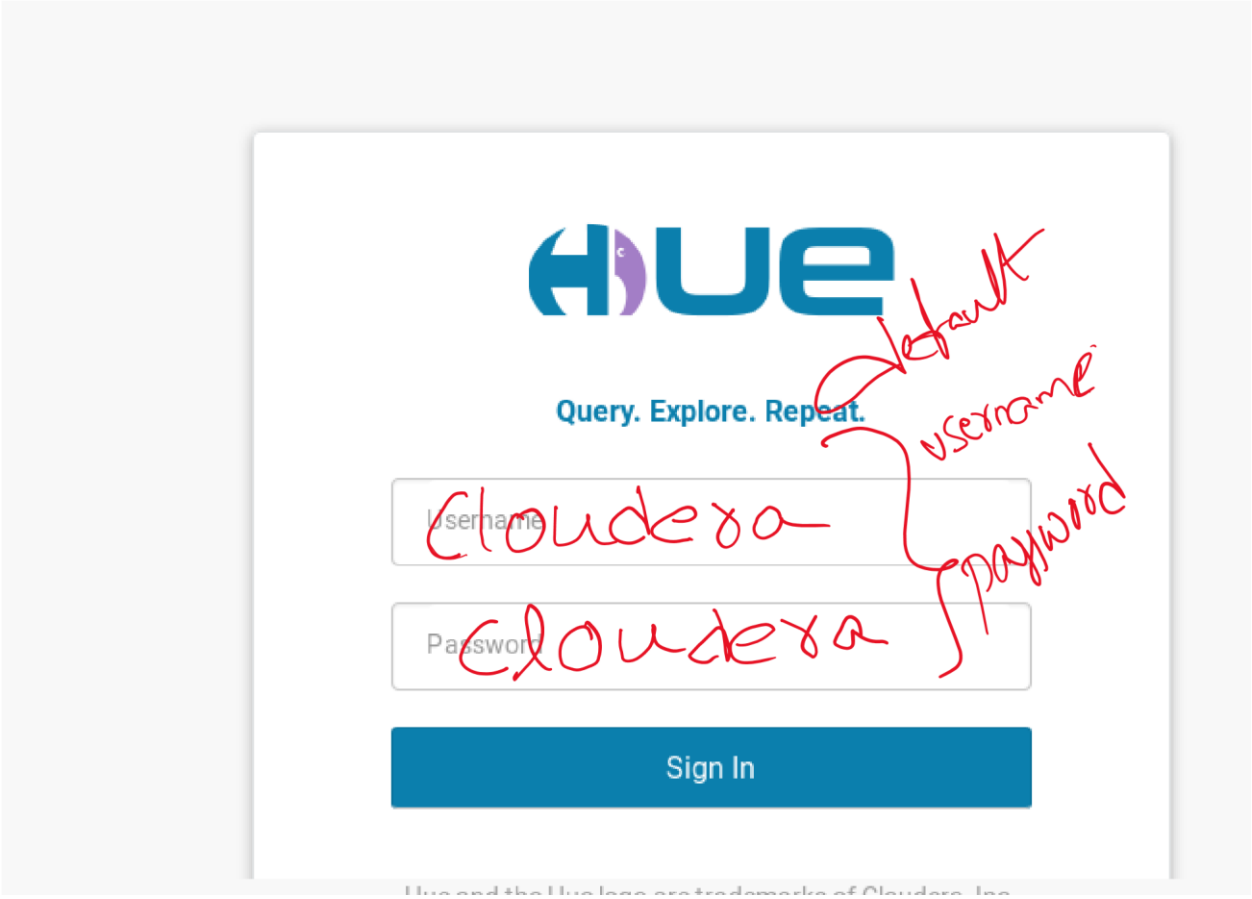
Select

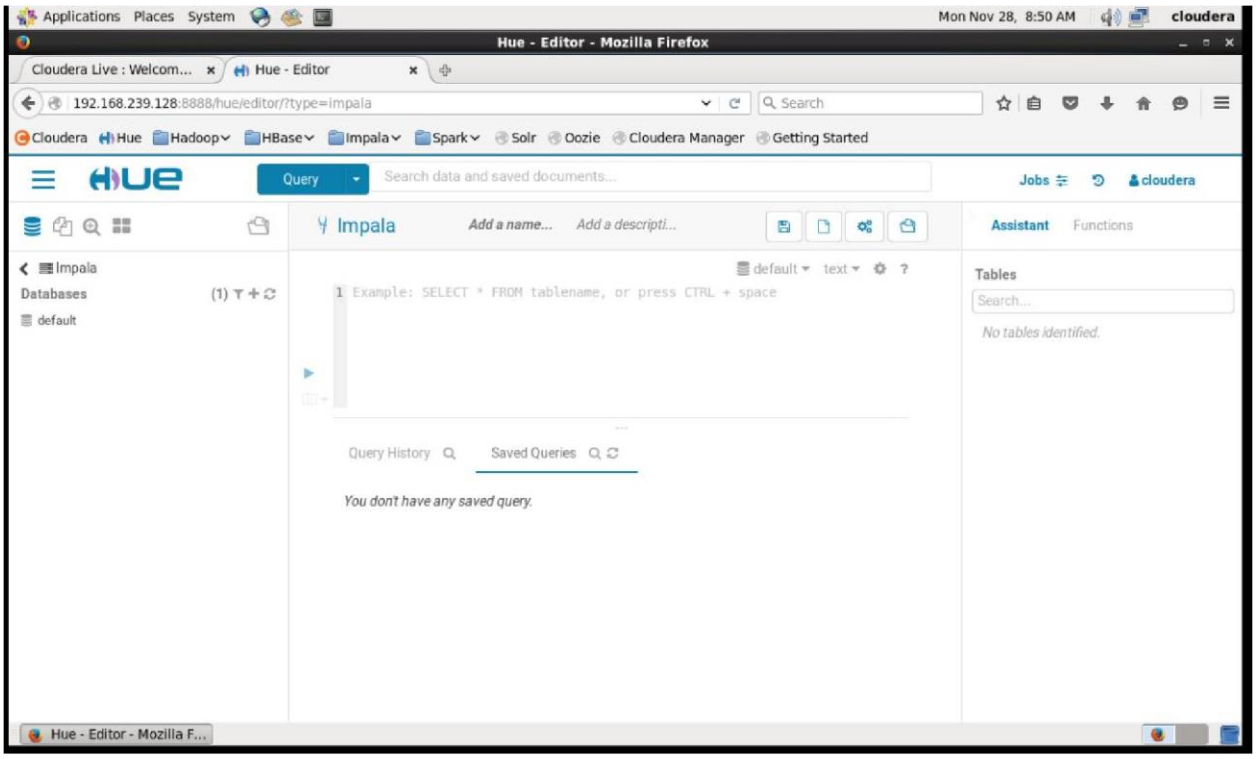


Select HUE ⇒ Hadoop user experience



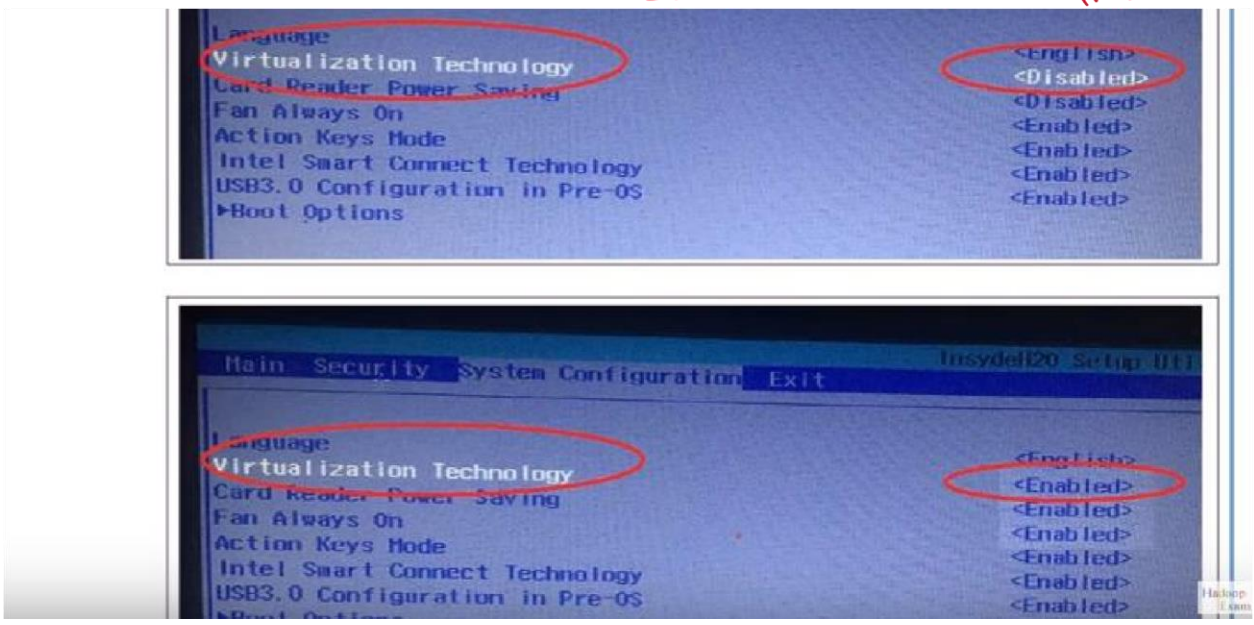
Welcome to Your Cloudera QuickStart V


Query. Explore. Repeat.
Username
Password
Sign In
default
username
password
cloudera
cloudera



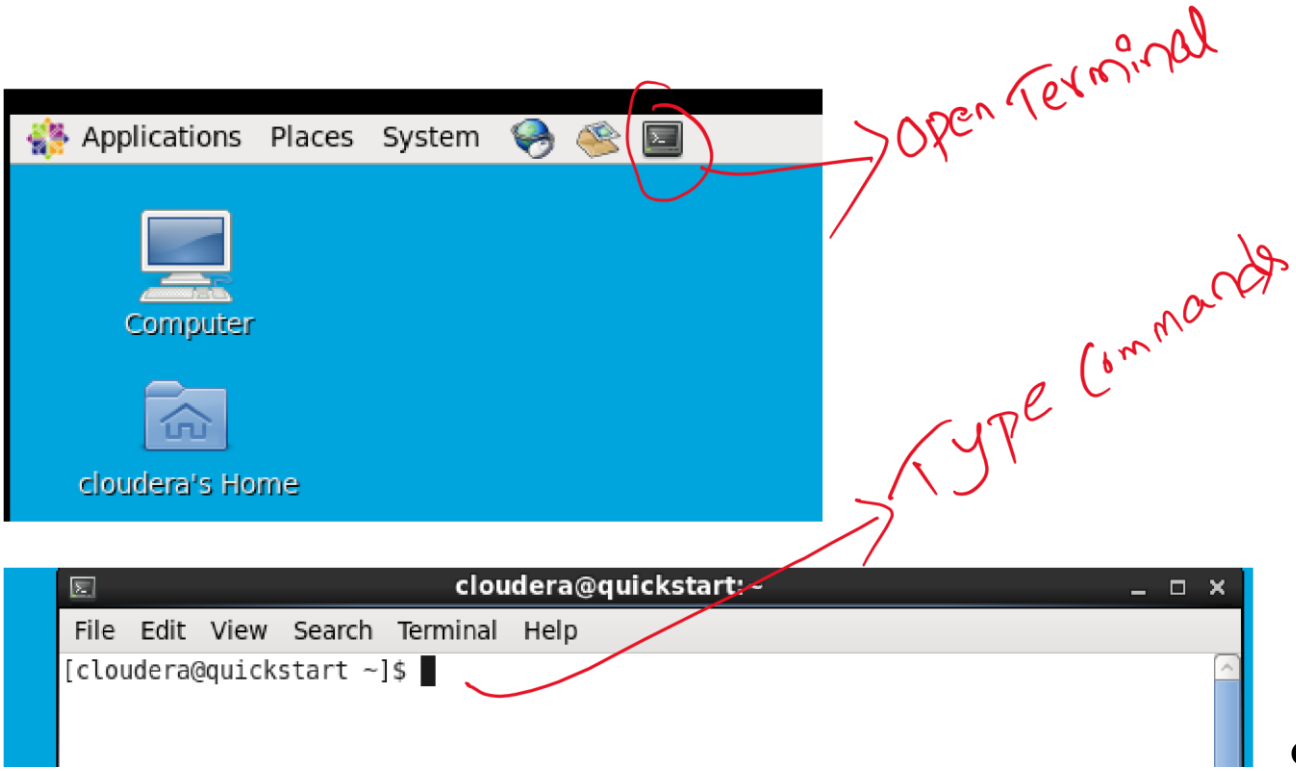
GUI based query editor

If Virtualization Technology is Disabled in BIOS setup } => Throws installation error.



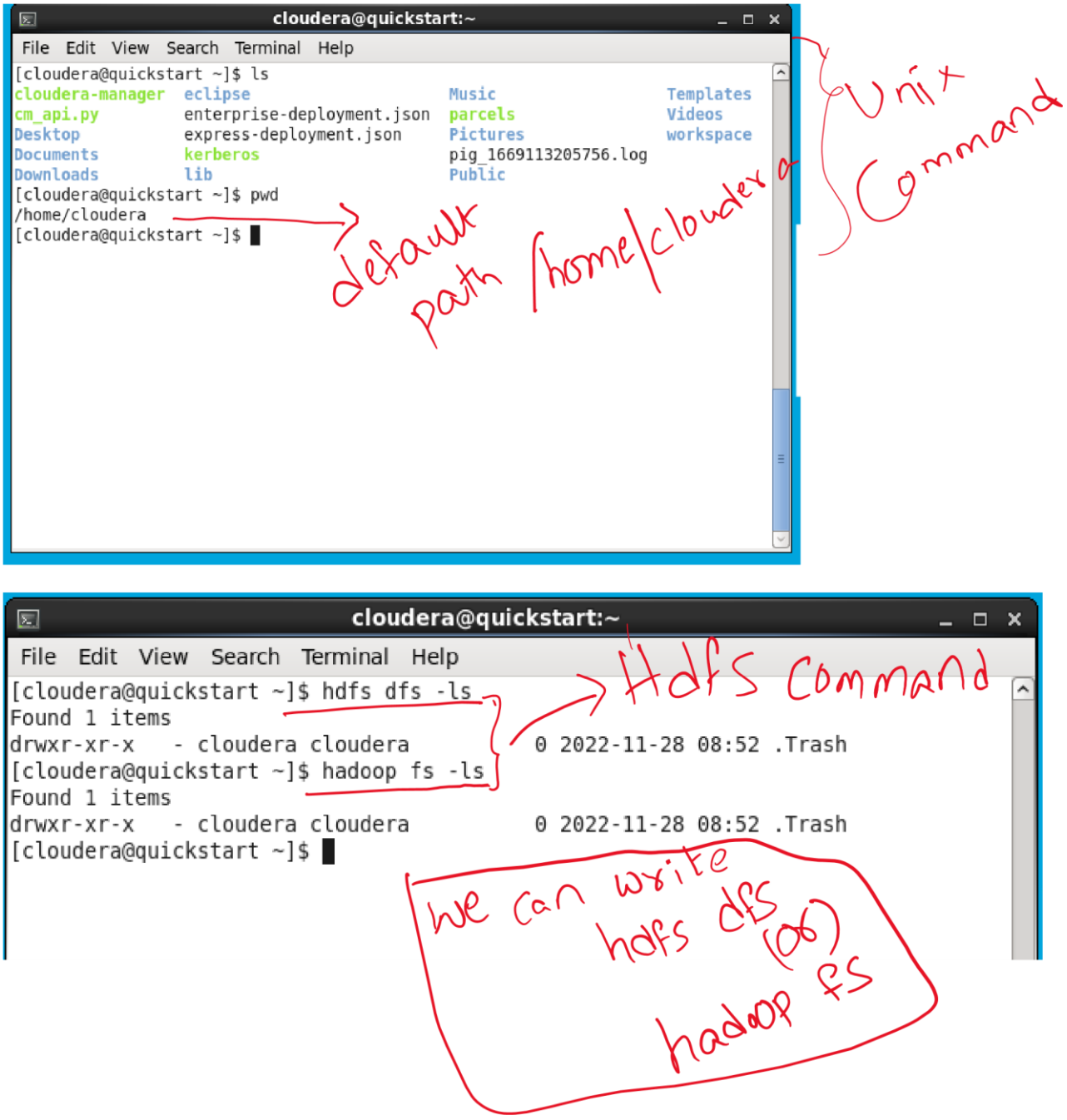
2 Hadoop Commands

- Hadoop is a open-source distributed framework that is used to store and process a large set of datasets. To store data, Hadoop uses HDFS, and to process data, it uses MapReduce & Yarn.
- hadoop fs or hdfs dfs are file system commands to interact with HDFS.



- These commands are very similar to Unix Commands.

Unix Commands	Hadoop Commands
default path /home/cloudera	**default path /user/cloudera**
a) ls command: This command is used to list all the files. ex: [cloudera@quickstart ~]$ ls	[cloudera@quickstart ~]$ **$hdfs dfs ls or** [cloudera@quickstart ~]$ **$hadoop fs -ls**



NOTE: From the same terminal, we can type both Unix and HDFS commands.

b) mkdir: creates directory

[cloudera@quickstart ~]$ cd demoLocal

[cloudera@quickstart demoLocal]$cd .. → previous directory.

[cloudera@quickstart ~]$ clear --- to clear the screen

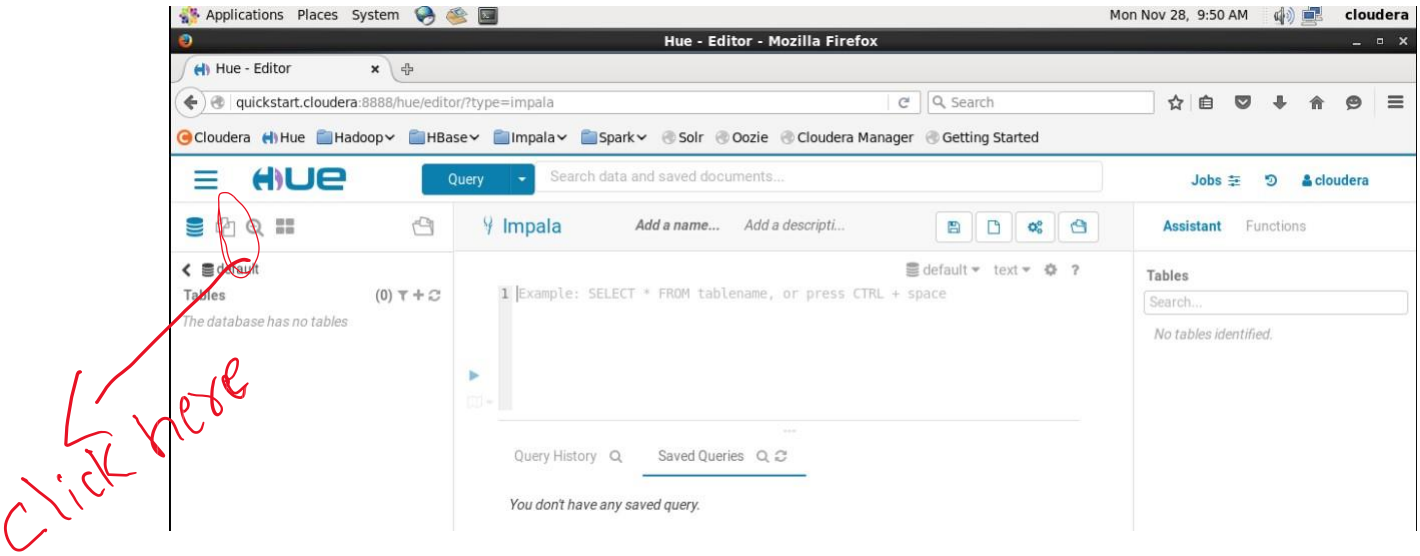
Note: Default path to HDFS files /user/cloudera

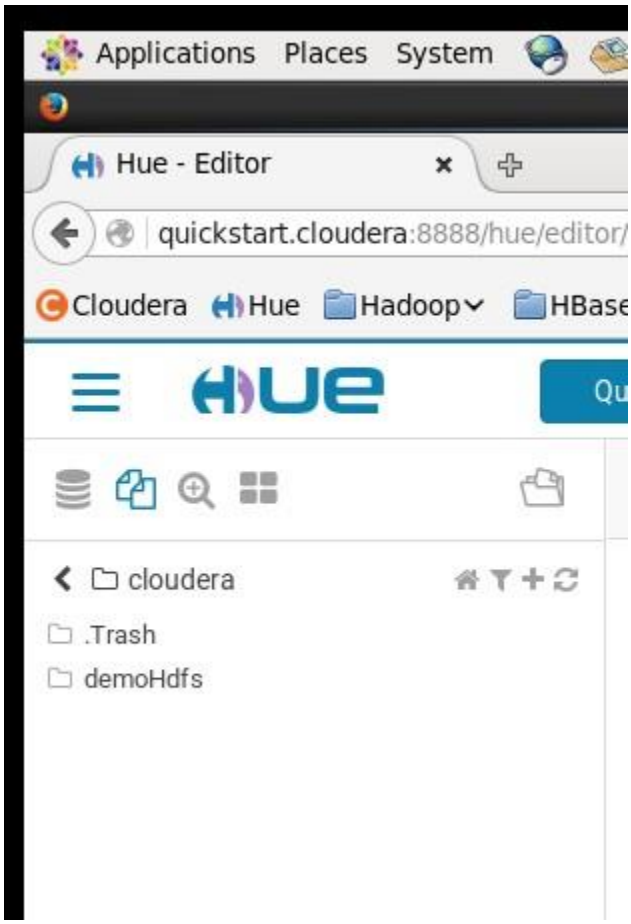
[cloudera@quickstart ~]$hdfs dfs -mkdir demoHdfs

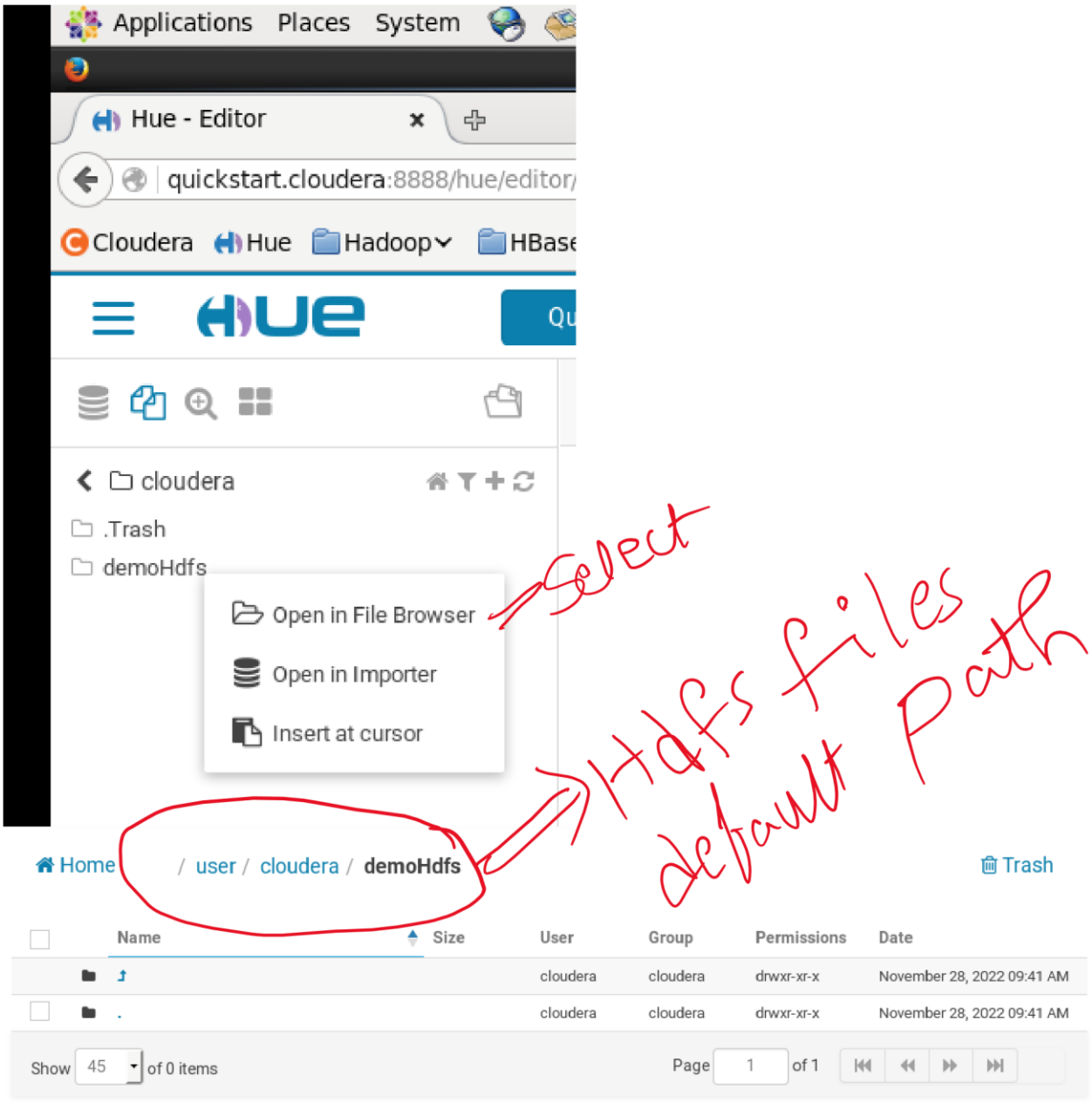
[cloudera@quickstart ~]$ hdfs dfs -ls

To check whether demoHdfs directory created or not, do the following steps:

i) **open browser -> click HUE (username and password: cloudera)**


Applications Places System
Mon Nov 28, 9:50 AM
cloudera
Hue - Editor - Mozilla Firefox
Hue - Editor
quickstart.cloudera:8888/hue/editor/?type=impala
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Query
Search data and saved documents...
Jobs
cloudera
Impala
Add a name...
Add a descripti...
Assistant
Functions
default
Tables
(0)
The database has no tables
default
text
1 Example: SELECT * FROM tablename, or press CTRL + space
Tables
Search...
No tables identified.
Query History
Saved Queries
You don't have any saved query.
click here


Applications
Places
System
Hue - Editor
quickstart.cloudera:8888/hue/editor/
Cloudera
Hue
Hadoop
HBase
HUE
cloudera
.Trash
demoHdfs



[cloudera@quickstart ~]$ hadoop version
Hadoop 2.6.0-cdh5.13.0
Subversion http://github.com/cloudera/hadoop -r 42e8860b182e55321bd5f5605264da4adc8882be
Compiled by jenkins on 2017-10-04T18:08Z Compiled with protoc 2.5.0
From source with checksum 5e84c185f8a22158e2b0e4b8f85311

This command was run using /usr/lib/hadoop/hadoopcommon-2.6.0-cdh5.13.0.jar
[cloudera@quickstart ~]$

Big Data Laboratory
Week - 2

1)Create a directory in Local server:
mkdir
Usage: hdfs dfs -mkdir [-p] <paths>
Takes path uri's as argument and creates directories.

```
[cloudera@quickstart ~]$ mkdir BD8044
```

2) To view files:
ls
Usage: hdfs dfs -ls [-R] <args>
Options:
The -R option will return stat recursively through the directory structure.

3) To Enter into directory:

```
[cloudera@quickstart ~]$ cd BD8044
```

4) To view the Present Working Directory(pwd):

```
[cloudera@quickstart BD8044]$ pwd
/home/cloudera/BD8044
```

5) To create a file in a directory:(Input the data and Press Ctrl+D to exit)
Cat
Usage: hdfs dfs -cat URI [URI ...]
Copies source paths to stdout.
Example:

- hdfs dfs -cat hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2
- hdfs dfs -cat file:///file3 /user/hadoop/file4

Exit Code:

Returns 0 on success and -1 on error.

```
cloudera@quickstart BD8044]$ cat abc.txt
cloudera@quickstart BD8044]$ gedit abc.txt
cloudera@quickstart BD8044]$ cat abc.txt
gitam
gitam
```

6)To view the content in the file created

```
cloudera@quickstart BD8044]$ cat abc.txt
gitam
gitam
```

7) To remove a file from directory

rm

Usage: hdfs dfs -rm [-f] [-r|-R] [-skipTrash] URI [URI ...]

Delete files specified as args.

```
cloudera@quickstart BD8044]$ rm /home/cloudera/BD8044/abc1.txt
m: remove regular empty file `/home/cloudera/BD8044/abc1.txt'? yes
```

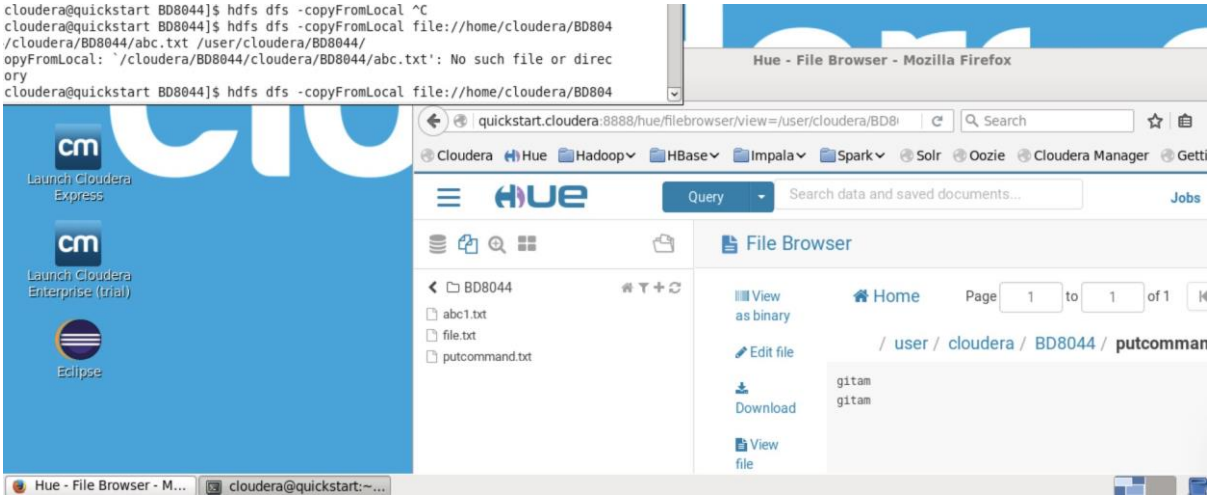
8) Check whether the files are created in directory

```
cloudera@quickstart BD8044]$ ls
abc1.txt  abc.txt  abc.txt~
```

11) To copyFromLocal

Usage: hdfs dfs -copyFromLocal <localsrc> URI

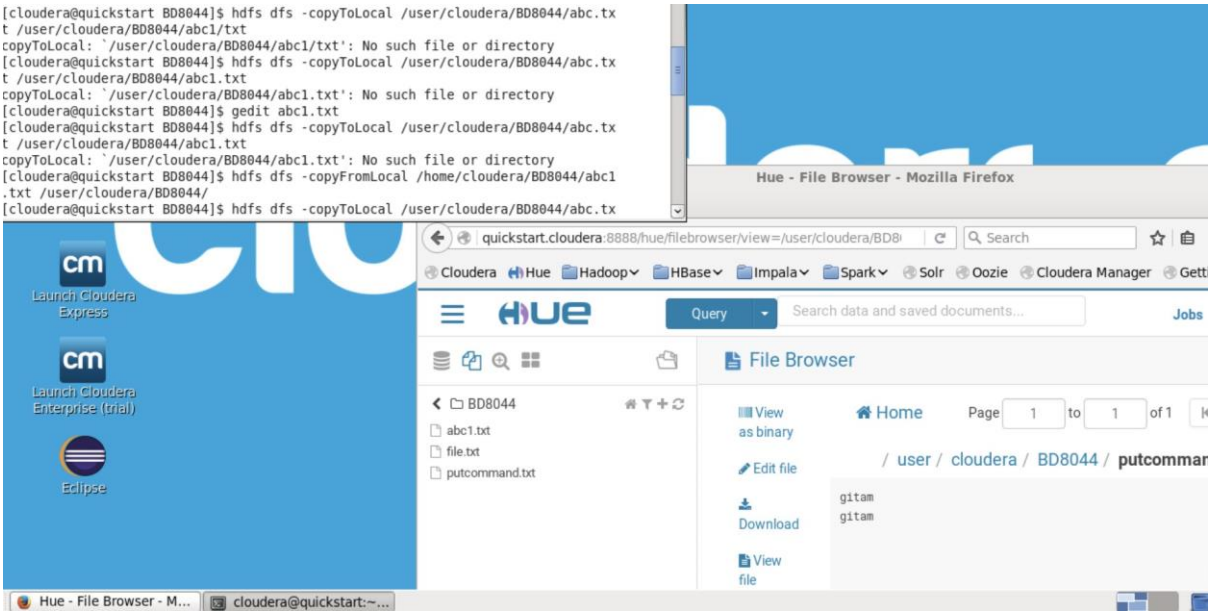
Similar to put command, except that the source is restricted to a local file reference.



12) To copyToLocal

Usage: hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

Similar to get command, except that the destination is restricted to a local file reference.



13) To move

mv

Usage: hdfs dfs -mv URI [URI ...] <dest>

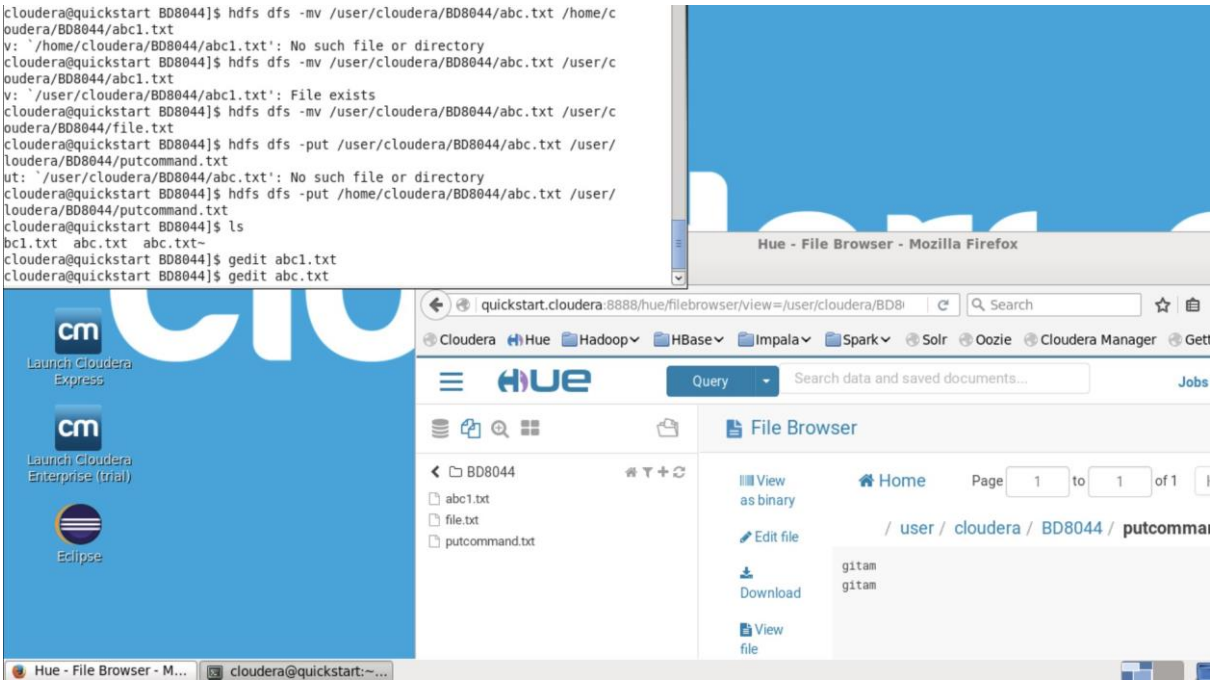
Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory. Moving files across file systems is not permitted.

Example:

- hdfs dfs -mv /user/hadoop/file1 /user/hadoop/file2
- hdfs dfs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1

Exit Code:

Returns 0 on success and -1 on error.



14) put command

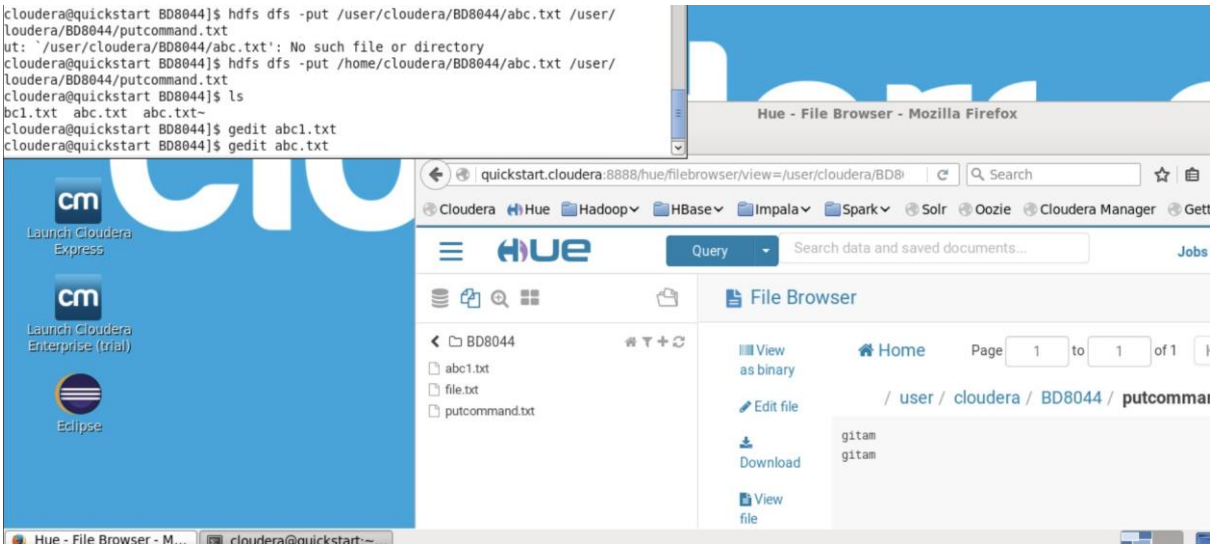
Usage: hdfs dfs -put <localsrc> ... <dst>

Copy single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and writes to destination file system.

- hdfs dfs -put localfile /user/hadoop/hadoopfile
- hdfs dfs -put localfile1 localfile2 /user/hadoop/hadoopdir
- hdfs dfs -put localfile hdfs://nn.example.com/hadoop/hadoopfile
- hdfs dfs -put - hdfs://nn.example.com/hadoop/hadoopfile Reads the input from stdin.

Exit Code:

Returns 0 on success and -1 on error.



Week – 3 BD LAB

Creating Directories Locally

[cloudera@quickstart ~]$ mkdir 8044

[cloudera@quickstart ~]$ ls

221910308044_1 cm_api.py Desktop eclipse kerberos parcels Templates

8044 demoLocal Documents enterprise-deployment.json lib Pictures Videos

cloudera-manager demoLocal_1 Downloads express-deployment.json Music Public workspace

[cloudera@quickstart ~]$ cd 8044

[cloudera@quickstart 8044]$ mkdir 8th_sem

[cloudera@quickstart 8044]$ mkdir placements

[cloudera@quickstart 8044]$ mkdir projects

[cloudera@quickstart 8044]$ ls

8th_sem placements projects

[cloudera@quickstart 8044]$ cd 8th_sem

[cloudera@quickstart 8th_sem]$ cat >BigData.txt

Big Data is structure , unstructrued and semi-structured in nature.

Difficult for computing systems due to high speed and volume.

Traditional data management, warehousing and analysis fizzle to analyze the high speed of data.

Hadoop by Apache is widely used for storing an managing Big data.

According to IBM, everyday we create 2.5 quintillion bytes of data - so much that 90% of the world today has been crecreated in the last two years alone. data- sensor data, climate data , gps data,bank data to name a few.

Big data is a high volume, velocity and variety information assets

Big data demand cost-effective, innovative forms of information processing for better decision making

big data is a collection of data that is huge in volume.

Yet big data growing exponentially with time

It is a data with so large size and complexity that none of traditional data managemnet.[cloudera@quickstart 8th_sem]$ cat>uhv.txt

Conformity

Tradition

Security

Power

Achievement

Hedonism

Stimulation

Self Direction

Universalism

Benevolance

[cloudera@quickstart 8th_sem]$ ls

BigData.txt uhv.txt

[cloudera@quickstart 8th_sem]$ pwd

/home/cloudera/8044/8th_sem

[cloudera@quickstart 8th_sem]$ cd

[cloudera@quickstart ~]$ pwd

/home/cloudera

[cloudera@quickstart ~]$ cd 8044

[cloudera@quickstart 8044]$ ls

8th_sem placements projects

[cloudera@quickstart 8044]$ cd placements

[cloudera@quickstart placements]$ cat >python.txt

Python

1

2

3

4

5

6

7

8

9

0

[cloudera@quickstart placements]$ cat >java.txt

Java

1

2

3

4

5

6

7

8

9

0

[cloudera@quickstart placements]$ cd

[cloudera@quickstart ~]$ cd 8044

[cloudera@quickstart 8044]$ cd projects

[cloudera@quickstart projects]$ cat

>miniproject.txt documentation presentation

implementation execution research verification

validation deployment sampling segregration

[cloudera@quickstart projects]$ cat >majorproject.txt aa abb abc abd abe abf abg abh

abi

abj

abk

abl

Creating Directories on Hadoop

[cloudera@quickstart projects]$ mkdir test

[cloudera@quickstart 8044]$ cd ..

[cloudera@quickstart ~]$ hdfs dfs -mkdir 8044

[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/cloudera/8044/Internal

[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/cloudera/8044/External

1) copy java.txt and mainproject.txt to “Internal”

[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal /home/cloudera/8044/placements/java.txt /user/cloudera/8044/Internal

[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal /home/cloudera/8044/projects/majorproject.txt /user/cloudera/8044/Internal

2) Check the contents of majorproject.txt at “Internal” folder

[cloudera@quickstart ~]$ hdfs dfs -cat

/user/cloudera/8044/Internal/majorproject.txt aa abb abc abd abe

abf

abg

abh

abi

abj

abk

abl

3) Move uhv.txt to “TEST” folder

[cloudera@quickstart 8th_sem]$ mv /home/cloudera/8044/8th_sem/uhv.txt /home/cloudera/8044/projects/test

[cloudera@quickstart 8th_sem]$ cd ..

[cloudera@quickstart 8044]$ cd projects

[cloudera@quickstart projects]$ cd

test [cloudera@quickstart test]$ ls

uhv.txt

[cloudera@quickstart test]$ cat uhv.txt

Conformity

Tradition

Security

Power

Achievement

Hedonism

Stimulation

Self Direction

Universalism

Benevolance

4) Copy python.txt to “External” folder

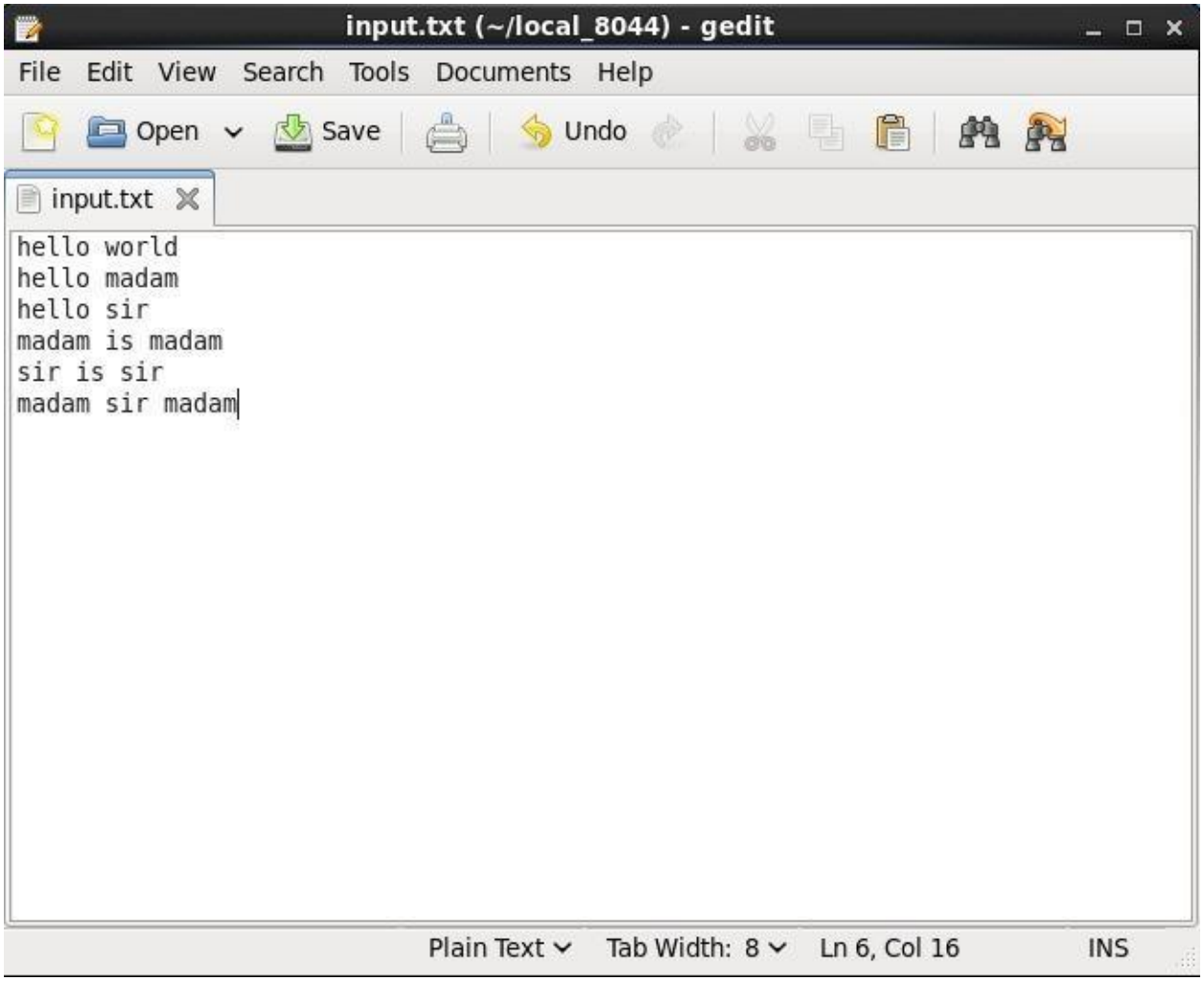
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal /home/cloudera/8044/placements/python.txt /user/cloudera/8044/External

5) Copy all files from “Internal” to “External

[cloudera@quickstart ~]$ hdfs dfs -cp /user/cloudera/8044/Internal /user/cloudera/8044/External

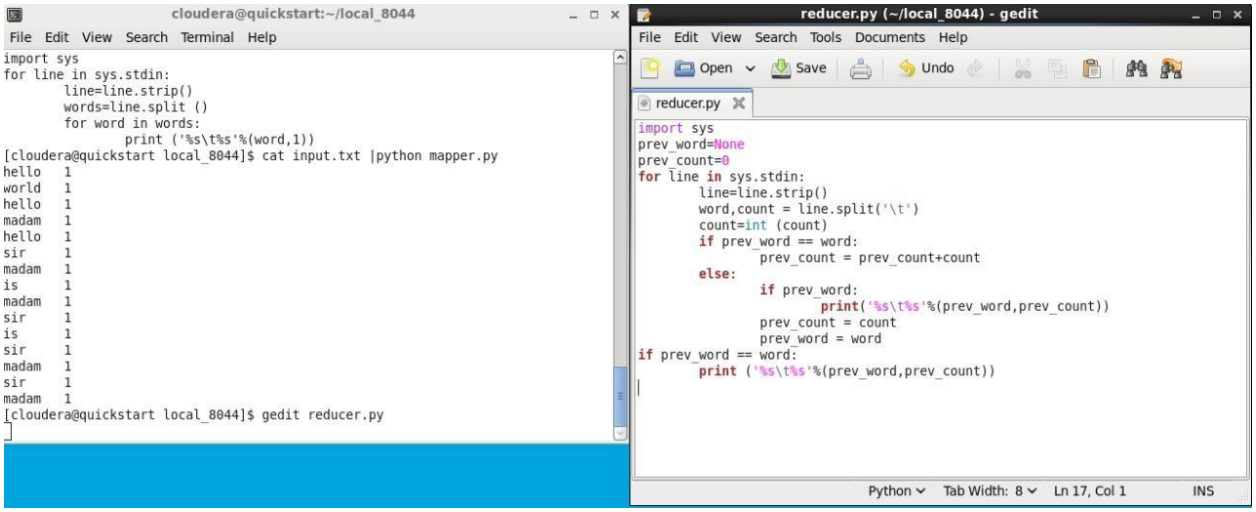
WEEK- 4

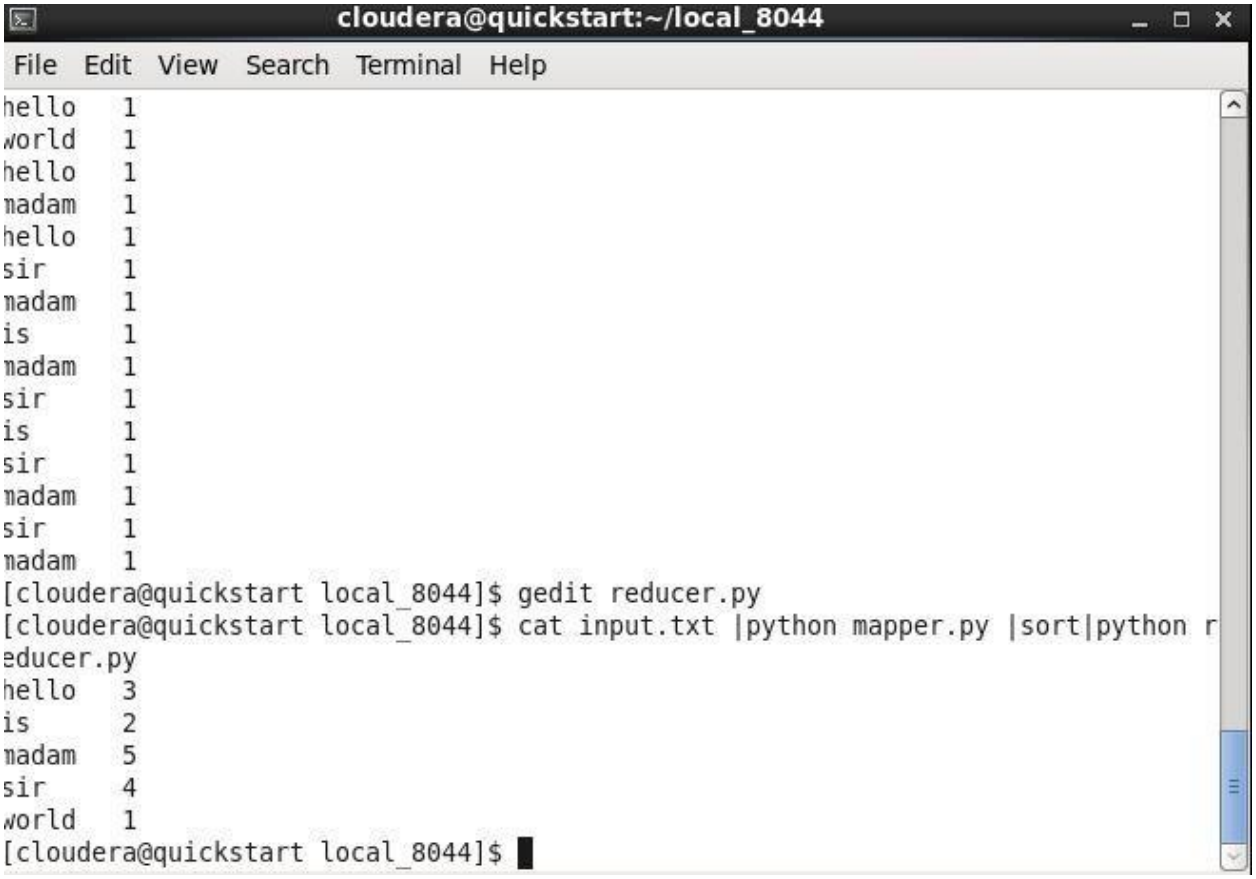
```
[cloudera@quickstart ~]$ mkdir local_8044
[cloudera@quickstart ~]$ cd local_8044
[cloudera@quickstart local_8044]$ gedit input.txt
```



```
S[cloudera@quickstart local_8044]$python
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[cloudera@quickstart local_8044]$ gedit mapper.py
[cloudera@quickstart local_8044]$ cat mapper.py
import sys
for line in sys.stdin:
        line=line.strip()
        words=line.split ()
        for word in words:
                print ('%s\t%s'%(word,1))
[cloudera@quickstart local_8044]$
```

```
[cloudera@quickstart local_8044]$ cat input.txt |python mapper.py
hello   1
world   1
hello   1
madam   1
hello   1
sir     1
madam   1
is      1
madam   1
sir     1
is      1
sir     1
madam   1
sir     1
madam   1
[cloudera@quickstart local_8044]$
```


cloudera@quickstart:~/local_8044
File Edit View Search Terminal Help
import sys
for line in sys.stdin:
line=line.strip()
words=line.split ()
for word in words:
print ('%s\t%s'%(word,1))
[cloudera@quickstart local_8044]$ cat input.txt |python mapper.py
hello 1
world 1
hello 1
madam 1
hello 1
sir 1
madam 1
is 1
madam 1
sir 1
is 1
sir 1
madam 1
sir 1
madam 1
[cloudera@quickstart local_8044]$ gedit reducer.py
reducer.py (~/local_8044) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
reducer.py
import sys
prev_word=None
prev_count=0
for line in sys.stdin:
line=line.strip()
word,count = line.split('\t')
count=int (count)
if prev_word == word:
prev_count = prev_count+count
else:
if prev_word:
print('%s\t%s'%(prev_word,prev_count))
prev_count = count
prev_word = word
if prev_word == word:
print ('%s\t%s'%(prev_word,prev_count))
Python Tab Width: 8 Ln 17, Col 1 INS


cloudera@quickstart:~/local_8044
File Edit View Search Terminal Help
hello 1
world 1
hello 1
madam 1
hello 1
sir 1
madam 1
is 1
madam 1
sir 1
is 1
sir 1
madam 1
sir 1
madam 1
[cloudera@quickstart local_8044]$ gedit reducer.py
[cloudera@quickstart local_8044]$ cat input.txt |python mapper.py |sort|python r
educer.py
hello 3
is 2
madam 5
sir 4
world 1
[cloudera@quickstart local_8044]$

WEEK - 5

grunt> [cloudera@quickstart 8044_pig]$ gedit employee.txt
to enter into pig local mode :
[cloudera@quickstart 8044_pig]$ pig -x local

grunt> student1= LOAD '/home/cloudera/8044_pig/student.txt' USING PigStorage(',') AS (id:int,firstname:chararray, lastname:chararray, phone:int, city:chararray);

grunt> STORE student1 INTO'/home/cloudera/8044_pig/pigOutput/'USING PigStorage(',');
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 19:41:51 2023-01-31 19:41:57 UNKNOWN

Success!

Job Stats (time in seconds):

JobId Alias Feature Outputs
Job_local875657594_0002 student1 MAP_ONLY /home/cloudera/8044_pig/pigOutput,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "/home/cloudera/8044_pig/pigOutput"

Job DAG:
job_local875657594_00022023-01-31 19:42:03,935 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher Success!

grunt> describe student1;
2023-01-31 19:43:10,373 [main] INFO org.apache.hadoop.conf.Configuration.deprecation fs.default.name is deprecated. Instead, use fs.defaultFS
2023-01-31 19:43:10,373 [main] INFO org.apache.hadoop.conf.Configuration.deprecation mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2023-01-31 19:43:10,374 [main] INFO org.apache.hadoop.conf.Configuration.deprecation io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum student1: {id: int,firstname: chararray,lastname: chararray,phone: int,city: chararray} **grunt> explain student1;**

grunt> illustrate student1;

```
-------------------------------------------------------------------------------------------------------------|
student1 | id:int | firstname:chararray | lastname:chararray | phone:int | city:chararray
|
-------------------------------------------------------------------------------------------------------------
|                | 82             | ss             | b              | 8710           | hyd    |
-------------------------------------------------------------------------------------------------------------
```

grunt> dump student1

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 19:52:13 2023-01-31 19:52:19
UNKNOWN
Success!

Job Stats (time in seconds):

JobId Alias Feature Outputs

job_local625885912_0003 student1 MAP_ONLY
file:/tmp/temp-2056100116/tmp1505697807,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-

2056100116/tmp1505697807" Job DAG: job_local625885912_0003 **grunt>**

group_data = GROUP student1 BY city; grunt> DUMP group_data

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 19:57:23 2023-01-31 19:57:29
GROUP_BY
Success!

Job Stats (time in seconds):

JobId Alias Feature Outputs

job_local1327638202_0004 group_data,student1 GROUP_BY file:/tmp/temp-
2056100116/tmp1106574879,

Input(s):

Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-2056100116/tmp1106574879"

Job DAG:
job_local1327638202_0004(hyd,{(10,Pranav,Reddy,8100,hyd),(9,Vasista,Babu,8150,hyd),(8,Dheeraj,Babu,8710,hyd),(7,Abhishek,Tillu,8054,hyd),(6,Srujan,Kolla,8027,hyd),(5,Irfan,Sk,8200,hyd),(4,Karthik,Hari,8201,hyd),(3,Dinesh,Daki,8110,hyd),(2,Ram,Charan,8015,hyd),(1,Aditya,Charan,8100,hyd)})

grunt> describe group_data; group_data: {group: chararray,student1: {(id: int,firstname: chararray,lastname: chararray,phone: int,city: chararray)}}

grunt> illustrate group_data;
---|student1 | id:int | firstname:chararray | lastname:chararray | phone:int | city:chararray|

| | 4 | ss | Ha | 8201 | hyd |
| | 10 | Vaa | oud | 8100 | hyd |

---| group_data | group:chararray | student1:bag{:tuple(id:int,firstname:chararray,lastname:chararray,phone:int,city:chararray)}|
---| | hyd | {(4, ..., hyd), (10, ..., hyd)} |
---

grunt> group_multiple = GROUP student1 BY (lastname,city);
grunt> dump group_multiple;

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 20:10:30 2023-01-31 20:10:43 GROUP_BY
Success!

Job Stats (time in seconds): JobId Alias Feature Outputs
job_local1033964090_0001 group_multiple,student1 GROUP_BY file:/tmp/temp-599394969/tmp480589619,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-599394969/tmp480589619"

Job DAG:
job_local1033964090_0001
((Sk,hyd),{(5,Irfan,Sk,8200,hyd)})
((Babu,hyd),{(9,Vasista,Babu,8150,hyd),(8,Dheeraj,Babu,8710,hyd)})
((Daki,hyd),{(3,Dinesh,Daki,8110,hyd)})
((Reddy,hyd),{(10,Pranav,Reddy,8100,hyd)})
((Hari,hyd),{(4,Karthik,Hari,8201,hyd)})
((Kolla,hyd),{(6,Srujan,Kolla,8027,hyd)})
((Tillu,hyd),{(7,Abhishek,Tillu,8054,hyd)})
((Charan,hyd),{(2,Ram,Charan,8015,hyd),(1,Aditya,Charan,8100,hyd)})

grunt> group_all = GROUP student1 All; grunt> dump

group_multiple;

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 20:10:30 2023-01-31
20:10:43 GROUP_BY Success!

Job Stats (time in seconds): JobId Alias Feature Outputs
job_local1033964090_0001 group_multiple,student1 GROUP_BY
file:/tmp/temp-599394969/tmp480589619,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-599394969/tmp480589619"

Job DAG:
job_local1033964090_0001
((Sk,hyd),{(5,Irfan,Sk,8200,hyd)})((Babu,hyd),{(9,Vasista,Babu,8150,hyd),(8,Dheeraj,Babu,8710
,
hyd)})((Daki,hyd),{(3,Dinesh,Daki,8110,hyd)})((Reddy,hyd),{(10,Pranav,Reddy,8100,hyd)})((Hari
,
hyd),{(4,Karthik,Hari,8201,hyd)})((Kolla,hyd),{(6,Srujan,Kolla,8027,hyd)})((Tillu,hyd),{(7,Abhishe
k,Tillu,8054,hyd)})((Charan,hyd),{(2,Ram,Charan,8015,hyd),(1,Aditya,Charan,8100,hyd)})
grunt> group_all = GROUP student1 All;

grunt> dump group_all;

2023-01-31 20:13:04,702 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-01-31 20:12:52 2023-01-31 20:13:04 GROUP_BY

Success!

Job Stats (time in seconds): JobId Alias Feature Outputs
job_local907292916_0002 group_all,student1 GROUP_BY
file:/tmp/temp-599394969/tmp-1410081782,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-599394969/tmp-1410081782"
Job DAG: job_local907292916_0002
(all,{(10,Pranav,Reddy,8100,hyd),(9,Vasista,Babu,8150,hyd),(8,Dheeraj,Babu,8710,hyd),(7,Abhishek,Tillu,8054,hyd),(6,Srujan,Kolla,8027,hyd),(5,Irfan,Sk,8200,hyd),(4,Karthik,Hari,8201,hyd),(3,Dinesh,Daki,8110,hyd),(2,Ram,Charan,8015,hyd),(1,Aditya,Charan,8100,hyd)})

grunt> customers = LOAD '/home/cloudera/8044_pig/join/customers.txt' USING PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int); grunt> orders = LOAD '/home/cloudera/8044_pig/join/orders.txt' USING PigStorage(',')as (oid:int, date:chararray, customer_id:int, amount:int); grunt> coustomer_orders = JOIN customers BY id, orders BY customer_id; grunt> dump coustomer_orders;

HadoopVersion PigVersion UserId StartedAtFinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-02-21 19:52:02 2023-02-21 19:52:15 HASH_JOIN

Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local1229324401_0022 coustomer_orders,customers,orders HASH_JOIN
file:/tmp/temp684888790/tmp907443576, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Successfully read records from: "/home/cloudera/80344_pig/join/customers.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp907443576"
Job DAG:
job_local1229324401_0022
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)

Inner Join

grunt> coustomer_orders = JOIN customers BY id, orders BY customer_id;
grunt> dump coustomer_orders; Success!
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1435502397"
Job DAG:

job_local776963137_0024
2023-02-21 19:57:14,938 [main] INFO
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)

Self-join

grunt> customers1 = LOAD '/home/cloudera/8044_pig/join/customers.txt' USING PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int); grunt> customers2 = LOAD '/home/cloudera/8044_pig/join/customers.txt' USING PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int); grunt> customers3 = JOIN customers1 BY id, customers2 BY id; grunt> dump customers3;

HadoopVersion PigVersion UserId StartedAtFinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-02-21 20:00:49 2023-02-21 20:01:02 HASH_JOIN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local1320335944_0025 customers1,customers2,customers3 HASH_JOIN
file:/tmp/temp684888790/tmp1600481455, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1600481455"
Job DAG:
job_local1320335944_0025
(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)
(2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500)
(3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000)
(4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500)
(5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500)
(6,Komal,22,MP,4500,6,Komal,22,MP,4500)
(7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)

grunt> outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;
dump outer_left;

HadoopVersion PigVersion UserId StartedAtFinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-02-21 20:01:56 Success! 2023-02-21 20:02:08 HASH_JOIN
Job Stats (time in seconds):
JobId Alias Feature Outputs

job_local32100113_0026 customers,orders,outer_left HASH_JOIN file:/tmp/temp684888790/tmp-541550944,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-541550944"
Job DAG:
job_local32100113_0026
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)(,,,,,,,,)

grunt> outer_right = JOIN customers BY id RIGHT, orders BY customer_id;
grunt> dump outer_right;

```
HadoopVersion   PigVersion      UserId  StartedAtFinishedAt     Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera 2023-02-21 20:03:15   2023-02-21 20:03:28   HASH_JOIN
Success!
Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local1203591295_0027  customers,orders,outer_right HASH_JOIN   file:/tmp/temp684888790/tmp-796814027,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-796814027"
Job DAG:
job_local1203591295_0027
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)
(,,,,,,,)

grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;
grunt> dump outer_full; Success!
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp504703719"
Job DAG:
job_local748198099_0028
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)(,,,,,,,,)(,,,,,,,,)

grunt> cross_data = CROSS customers, orders;
grunt> dump cross_data;
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/join/customers.txt"
Successfully read records from: "/home/cloudera/8044_pig/join/orders.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1569998779"
Job DAG: job_local853946806_0029
(,,,,,,,)
(,,,,,103,2008-05-20,4,2060) (,,,,,101,2009-11-
20,2,1560)
(,,,,,100,2009-10-08,3,1500)
(,,,,,102,2009-10-08,3,3000)
(7,Muffy,24,Indore,10000,,,,)
(7,Muffy,24,Indore,10000,103,2008-05-20,4,2060)
(7,Muffy,24,Indore,10000,101,2009-11-20,2,1560)
(7,Muffy,24,Indore,10000,100,2009-10-08,3,1500)
(7,Muffy,24,Indore,10000,102,2009-10-08,3,3000)
(6,Komal,22,MP,4500,,,,)
(6,Komal,22,MP,4500,103,2008-05-20,4,2060) (6,Komal,22,MP,4500,101,2009-
11-20,2,1560)
(6,Komal,22,MP,4500,100,2009-10-08,3,1500)
```

(6,Komal,22,MP,4500,102,2009-10-08,3,3000)
(5,Hardik,27,Bhopal,8500,,,,)
(5,Hardik,27,Bhopal,8500,103,2008-05-20,4,2060)
(5,Hardik,27,Bhopal,8500,101,2009-11-20,2,1560)
(5,Hardik,27,Bhopal,8500,100,2009-10-08,3,1500)
(5,Hardik,27,Bhopal,8500,102,2009-10-08,3,3000)
(4,Chaitali,25,Mumbai,6500,,,,)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20,4,2060)
(4,Chaitali,25,Mumbai,6500,101,2009-11-20,2,1560)
(4,Chaitali,25,Mumbai,6500,100,2009-10-08,3,1500)
(4,Chaitali,25,Mumbai,6500,102,2009-10-08,3,3000)
(3,kaushik,23,Kota,2000,,,,)
(3,kaushik,23,Kota,2000,103,2008-05-20,4,2060)
(3,kaushik,23,Kota,2000,101,2009-11-20,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08,3,3000)
(2,Khilan,25,Delhi,1500,,,,)
(2,Khilan,25,Delhi,1500,103,2008-05-20,4,2060)
(2,Khilan,25,Delhi,1500,101,2009-11-20,2,1560)
(2,Khilan,25,Delhi,1500,100,2009-10-08,3,1500)
(2,Khilan,25,Delhi,1500,102,2009-10-08,3,3000)
(1,Ramesh,32,Ahmedabad,2000,,,,)
(1,Ramesh,32,Ahmedabad,2000,103,2008-05-20,4,2060)
(1,Ramesh,32,Ahmedabad,2000,101,2009-11-20,2,1560)
(1,Ramesh,32,Ahmedabad,2000,100,2009-10-08,3,1500)
(1,Ramesh,32,Ahmedabad,2000,102,2009-10-08,3,3000)

WEEK-6

```
[cloudera@quickstart ~]$ cd 8044_pig
[cloudera@quickstart 8044_pig]$ pig -x local
grunt> student1 = LOAD '/home/cloudera/8044_pig/student1.txt' USING PigStorage(',') as (id:int,
firstname:chararray, lastname:chararray, phone:chararray, city:chararray); grunt> student2 =
LOAD '/home/cloudera/8044_pig/student2.txt' USING PigStorage(',') as (id:int,
firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
grunt> student = UNION student1,student2;
grunt> Dump student;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 03:36:19        2023-02-21 03:36:36        UNION
Success!
Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local1826418349_0001        student,student1,student2 MAP_ONLY
file:/tmp/temp1430935385/tmp-66186234, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp1430935385/tmp-66186234"
Job DAG:
Job_local1826418349_0001
(1,Rajiv,Reddy,9848022337,Hyderabad )
(2,siddarth,Battacharya,9848022338,Kolkata )
(3,Rajesh,Khanna,9848022339,Delhi )
(4,Preethi,Agarwal,9848022330,Pune )
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar )
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,Trivendram)
(8,Bharathi,Nambiayar,9848022333,Chennai )
(,,,,)

grunt> SPLIT student into student1 if id<5, student2 if (id>=5 and id<=8);
grunt> dump student1;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features

2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 03:50:15        2023-02-21 03:50:29        UNION
Success!
Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local264266537_0002student,student1,student2 MAP_ONLY
file:/tmp/temp-1966704737/tmp-2102617194,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1966704737/tmp-2102617194"
Job DAG:
job_local264266537_0002
(1,Rajiv,Reddy,9848022337,Hyderabad )
(2,siddarth,Battacharya,9848022338,Kolkata )
```

(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
grunt> dump student2;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 03:50:43 2023-02-21 03:50:57 UNION
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local973261997_0003student,student1,student2 MAP_ONLY file:/tmp/temp-1966704737/tmp-30390278,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1966704737/tmp-30390278"
Job DAG:
Job_local973261997_0003
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,Trivendram)
(8,Bharathi,Nambiayar,9848022333,Chennai)

grunt> filter_data = FILTER student BY city == 'Chennai';
Dump filter_data;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 03:55:41 2023-02-21 03:55:56 FILTER,UNION Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local513044262_0001filter_data,student,student1,student2MAP_ONLY file:/tmp/temp-981425089/tmp-1873043599,

Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-981425089/tmp-1873043599"
Job DAG:
job_local513044262_0001
(6,Archana,Mishra,9848022335,Chennai)

grunt> distinct_data = DISTINCT student;
grunt>dump distinct_data;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:03:50 2023-02-21 04:04:07
DISTINCT,UNION
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local1166082710_0001 student,student1,student2 DISTINCT
file:/tmp/temp-1991038400/tmp1165624713, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):

```
Successfully stored records in: "file:/tmp/temp-1991038400/tmp1165624713"
Job DAG:
job_local1166082710_0001
(1,Rajiv,Reddy,9848022337,Hyderabad )
(2,siddarth,Battacharya,9848022338,Kolkata )
(3,Rajesh,Khanna,9848022339,Delhi )
(4,Preethi,Agarwal,9848022330,Pune )
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar )
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,Trivendram)
(8,Bharathi,Nambiayar,9848022333,Chennai )
(,,,,)

grunt> foreach_data = FOREACH student GENERATE id,city;
grunt>Dump foreach_data;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:06:39        2023-02-21 04:06:53        UNION
Success!
Job Stats (time in seconds):
JobId     Alias     Feature Outputs
job_local337126499_0002foreach_data,student,student1,student2
MAP_ONLY file:/tmp/temp-1991038400/tmp-1786253640, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp-1786253640"
Job DAG:
job_local337126499_0002
(1,Hyderabad )
(2,Kolkata )
(3,Delhi )
(4,Pune )
(5,Bhuwaneshwar )
(6,Chennai)
(7,Trivendram)
(8,Chennai )
(,)

grunt> order_by_data = ORDER student BY city DESC;
grunt> Dump order_by_data;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0     0.12.0-cdh5.13.0 cloudera2023-02-21 04:08:36        2023-02-21 04:09:15
ORDER_BY,UNION
Success!
Job Stats (time in seconds):
JobId     Alias     Feature Outputs
job_local2036620617_0005          order_by_data     ORDER_BY       file:/tmp/temp-1991038400/tmp-
238772032, job_local517778782_0003student,student1,student2 MAP_ONLY
job_local629277012_0004order_by_data      SAMPLER
Input(s):
```

```
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp-238772032"
Job DAG:

job_local517778782_0003->	job_local629277012_0004,
job_local629277012_0004->	job_local2036620617_0005,
job_local2036620617_0005
(7,Komal,Nayak,9848022334,Trivendram)
(4,Preethi,Agarwal,9848022330,Pune )
(2,siddarth,Battacharya,9848022338,Kolkata )
(1,Rajiv,Reddy,9848022337,Hyderabad )
(3,Rajesh,Khanna,9848022339,Delhi )
(8,Bharathi,Nambiayar,9848022333,Chennai )
(6,Archana,Mishra,9848022335,Chennai)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar )
(,,,,)

grunt> limit_data = LIMIT student 4;
grunt> dump limit_data;
HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt	Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:10:43 2023-02-21 04:12:03 LIMIT,UNION Success!

Job Stats (time in seconds):
JobId	Alias	Feature	Outputs
job_local1209999958_0009		student2
job_local1639624460_0007		student2
job_local2088962484_0008		student1
job_local339977671_0006student1
job_local906100408_0010limit_data,student
job_local995624576_0011	file:/tmp/temp-
1991038400/tmp-623417348, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp-623417348"
Job DAG:
job_local339977671_0006->	job_local2088962484_0008,
job_local2088962484_0008	->	job_local906100408_0010,
job_local1639624460_0007	->	job_local1209999958_0009,
job_local1209999958_0009	->	job_local906100408_0010,
job_local906100408_0010->	job_local995624576_0011,
job_local995624576_0011
(1,Rajiv,Reddy,9848022337,Hyderabad )
(2,siddarth,Battacharya,9848022338,Kolkata )
(3,Rajesh,Khanna,9848022339,Delhi )
(4,Preethi,Agarwal,9848022330,Pune )

grunt> student_group_all = Group student All;
grunt> dump student_group_all;
HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt	Features
```

2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:14:13 2023-02-21 04:14:26
GROUP_BY,UNION
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local1442882893_0012 student,student1,student2,student_group_all GROUP_BY file:/tmp/temp-
1991038400/tmp-1911199507,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp-
1911199507" Job DAG: job_local1442882893_0012

(all,{(,,,,),(8,Bharathi,Nambiayar,9848022333,Chennai
),(7,Komal,Nayak,9848022334,Trivendram),(6,Archana,Mishra,9848022335,Chennai),(5,Trupthi,Mohanthy,984802
2336,Bhuwaneshwar),(4,Preethi,Agarwal,9848022330,Pune),(3,Rajesh,Khanna,9848022339,Delhi
),(2,siddarth,Battacharya,9848022338,Kolkata),(1,Rajiv,Reddy,9848022337,Hyderabad)})

grunt> student_id_avg = foreach student_group_all Generate student, AVG(student.id); grunt>
student_id_avg = foreach student_group_all Generate (student.id,student.city), AVG(student.id);
grunt> student_id_avg = foreach student_group_all Generate AVG(student.id); grunt> Dump
student_id_avg;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:18:12 2023-02-21 04:18:25
GROUP_BY,UNION
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local1495144860_0013 student,student1,student2,student_group_all,student_id_avg
GROUP_BY,COMBINERfile:/tmp/temp-1991038400/tmp-1325427288,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp-1325427288"
Job DAG:
job_local1495144860_0013
(4.5)

grunt> student_id_max =l foreach student_group_alGenerate (student.id,student.city), MAX(student.id);
grunt> DUMP student_id_max;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 04:20:04 2023-02-21 04:20:17
GROUP_BY,UNION
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local849927799_0014student,student1,student2,student_group_all,student_id_max GROUP_BY
file:/tmp/temp-1991038400/tmp1853925741, Input(s):

```
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1991038400/tmp1853925741"
Job DAG:
job_local849927799_0014
(({(),(8),(7),(6),(5),(4),(3),(2),(1)},{(),(Chennai ),(Trivendram),(Chennai),(Bhuwaneshwar ),(Pune ),(Delhi
),(Kolkata ),(Hyderabad )}),8) grunt> student_id_min = foreach student_group_all Generate

(student.id,student.city), MIN(student.id); grunt> Dump student_id_min;

HadoopVersion    PigVersion          UserId   StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0    0.12.0-cdh5.13.0 cloudera2023-02-21 04:31:06         2023-02-21 04:31:24
GROUP_BY,UNION
Success!
Job Stats (time in seconds):
JobId     Alias     Feature Outputs
job_local1044436065_0001          student,student1,student2,student_group_all,student_id_min    GROUP_BY
file:/tmp/temp1783146403/tmp-2044909029, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp1783146403/tmp-2044909029"
Job DAG:
job_local1044436065_0001
(({(),(8),(7),(6),(5),(4),(3),(2),(1)},{(),(Chennai ),(Trivendram),(Chennai),(Bhuwaneshwar ),(Pune ),(Delhi
),(Kolkata ),(Hyderabad )}),1)

grunt> student_id_count = foreach student_group_all Generate COUNT(student.id);
HadoopVersion    PigVersion          UserId   StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0    0.12.0-cdh5.13.0 cloudera2023-02-21 04:31:06         2023-02-21 04:31:24
GROUP_BY,UNION
Success!
Job Stats (time in seconds):
JobId     Alias     Feature Outputs
job_local1044436065_0001          student,student1,student2,student_group_all,student_id_min    GROUP_BY
file:/tmp/temp1783146403/tmp-2044909029, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/student2.txt"
Successfully read records from: "/home/cloudera/8044_pig/student1.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp1783146403/tmp-2044909029"
Job DAG:
job_local1044436065_0001
8
```

Week - 7

Word count using pig

grunt> lines = LOAD '/home/cloudera/8044_pig/wc.txt' AS (line:chararray);
grunt> dump lines; Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local1212899870_0001 lines MAP_ONLY file:/tmp/temp-1022446998/tmp886033061,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/wc.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1022446998/tmp886033061"
Job DAG:
job_local1212899870_0001
(This is pig latin script, and I am performing word count.)

grunt> words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word; grunt>dump words; Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local177336160_0002lines,words MAP_ONLY file:/tmp/temp-1022446998/tmp569267224,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/wc.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1022446998/tmp569267224"
Job DAG: job_local177336160_0002
(This)
(is)
(pig)
(latin)
(script)
(and)
(I)
(am)
(performing)
(word)
(count.)

grunt> grouped = GROUP words BY word;
grunt> dump grouped;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:00:33 2023-02-21 19:00:46 GROUP_BY Success!
Job Stats (time in seconds): JobId
Alias Feature Outputs
job_local1470234805_0003
grouped,lines,words
GROUP_BY

file:/tmp/temp-1022446998/tmp848016435,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/wc.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1022446998/tmp848016435"
Job DAG: job_local1470234805_0003
(I,{(I)})
(am,{(am)})
(is,{(is)})
(and,{(and)})
(pig,{(pig)})
(This,{(This)})
(word,{(word)})
(latin,{(latin)})
(count.,{(count.)})
(script,{(script)})
(performing,{(performing)})

```
grunt> wordcount = FOREACH grouped GENERATE group, COUNT(words);
grunt> dump wordcount;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:01:40        2023-02-21 19:01:52        GROUP_BY
Success!
Job Stats (time in seconds):
JobId     Alias     Feature Outputs
job_local690062669_0004grouped,lines,wordcount,words
        GROUP_BY,COMBINER file:/tmp/temp-1022446998/tmp52293942, Input(s):
Successfully read records from: "/home/cloudera/8044_pig/wc.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1022446998/tmp52293942"
Job DAG:
job_local690062669_0004
(I,1)
(am,1)
(is,1)
(and,1)
(pig,1)
(This,1)
(word,1)
(latin,1)
(count.,1)
(script,1)
(performing,1)
```

STARTSWITH

```
grunt> s = load '/home/cloudera/8044_pig/emp.txt' using PigStorage(',') AS
(id:int,name:chararray,age:int,city:chararray);
grunt> startswith_data = FOREACH s GENERATE (id,name), STARTSWITH (name,'Ro');
grunt> dump startswith_data;
HadoopVersion   PigVersion         UserId  StartedAt           FinishedAt          Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:09:04        2023-02-21 19:09:19        UNKNOWN
Success!
Job Stats (time in seconds):
```

JobId Alias Feature Outputs job_local143130368_0001s,startswith_data MAP_ONLY
file:/tmp/temp684888790/tmp-127282921,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-127282921"
Job DAG:
job_local143130368_0001
((1,Robin),true)
((2,BOB),false)
((3,Maya),false)
((4,Sara),false)
((5,David),false)
((6,Maggy),false)
((7,Robert),true)
((8,Syam),false)
((9,Mary),false)
((10,Saran),false)
((11,Stacy),false)
((12,Kelly),false)
((,),)

ENDSWITH

```
grunt> endswith_data = FOREACH s GENERATE (name), ENDSWITH (name,'id');
grunt> dump endswith_data;
```

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:11:33 2023-02-21 19:11:46 UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local1513545110_0002 endswith_data,s MAP_ONLY
file:/tmp/temp684888790/tmp1294856412,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1294856412"
Job DAG: job_local1513545110_0002
(Robin,false)
(BOB,false)
(Maya,false)
(Sara,false)
(David,true)
(Maggy,false)
(Robert,false)
(Syam,false)
(Mary,false)
(Saran,false)
(Stacy,false)
(Kelly,false)
(,)

EqualsIgnoreCase - (did not execute)

INDEXOF ( )

```
grunt> indexof_data = FOREACH s GENERATE (id,name), INDEXOF(name, 'a');
grunt> dump indexof_data;
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:22:09    2023-02-21 19:22:16     UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local1309811012_0008 indexof_data,s MAP_ONLY
file:/tmp/temp684888790/tmp1133051610,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1133051610"
Job DAG: job_local1309811012_0008
((1,Robin),-1)
((2,BOB),-1)
((3,Maya),1)
((4,Sara),1)
((5,David),1)
((6,Maggy),1)
((7,Robert),-1)
((8,Syam),2)
((9,Mary),1)
((10,Saran),1)
((11,Stacy),2)
((12,Kelly),-1)
((,),)

grunt> indexof_data = FOREACH s GENERATE (id,name), LAST_INDEX_OF(name, 'a');
grunt> dump indexof_data;
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:23:41    2023-02-21 19:23:54     UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local2092085637_0009 indexof_data,s MAP_ONLY
file:/tmp/temp684888790/tmp317067729,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp317067729"
Job DAG: job_local2092085637_0009
((1,Robin),-1)
((2,BOB),-1)
((3,Maya),3)
((4,Sara),3)
((5,David),1)
((6,Maggy),1)
((7,Robert),-1)
((8,Syam),2)
((9,Mary),1)
```

((10,Saran),3)
((11,Stacy),2)
((12,Kelly),-1)
((,),)

LCFIRST() & UCFIRST()

```
grunt> Lcfirst_data = FOREACH s GENERATE (id,name), LCFIRST(name);
grunt> DUMP Lcfirst_data;
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:25:01    2023-02-21 19:25:14     UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local630276362_0010Lcfirst_data,s MAP_ONLY
file:/tmp/temp684888790/tmp-137178099,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-137178099"
Job DAG:
job_local630276362_0010
((1,Robin),robin)
((2,BOB),bOB)
((3,Maya),maya)
((4,Sara),sara)
((5,David),david)
((6,Maggy),maggy)
((7,Robert),robert)
((8,Syam),syam)
((9,Mary),mary)
((10,Saran),saran)
((11,Stacy),stacy)
((12,Kelly),kelly)
((,),)
grunt> Ucfirst_data = FOREACH s GENERATE (id,name), UCFIRST(name);
grunt> DUMP Ucfirst_data;
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:26:47    2023-02-21 19:27:00     UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local1282945209_0011 Ucfirst_data,s MAP_ONLY
file:/tmp/temp684888790/tmp-972927322,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-972927322"
Job DAG:
job_local1282945209_0011
((1,Robin),Robin)
((2,BOB),BOB)
((3,Maya),Maya)
((4,Sara),Sara)
```

((5,David),David)
((6,Maggy),Maggy)
((7,Robert),Robert)
((8,Syam),Syam)
((9,Mary),Mary)
((10,Saran),Saran)
((11,Stacy),Stacy)
((12,Kelly),Kelly)
((,),)

upper

grunt> upper_data = FOREACH s GENERATE (id,name), UPPER(name);
grunt> dump upper_data;
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
2.6.0-cdh5.13.0 0.12.0-cdh5.13.0 cloudera2023-02-21 19:28:10 2023-02-21 19:28:22 UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs job_local1889501186_0012 s,upper_data MAP_ONLY
file:/tmp/temp684888790/tmp-593317567,
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-593317567"
Job DAG:
job_local1889501186_0012
((1,Robin),ROBIN)
((2,BOB),BOB)
((3,Maya),MAYA)
((4,Sara),SARA)
((5,David),DAVID)
((6,Maggy),MAGGY)
((7,Robert),ROBERT)
((8,Syam),SYAM)
((9,Mary),MARY)
((10,Saran),SARAN)
((11,Stacy),STACY)
((12,Kelly),KELLY)
((,),)

LOWER

grunt> lower_data = FOREACH s GENERATE (id,name),
LOWER(name); grunt> dump lower_data; Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp-1976578602"
Job DAG:
job_local372141645_0013
((1,Robin),robin)

((2,BOB),bob)
((3,Maya),maya)
((4,Sara),sara)
((5,David),david)
((6,Maggy),maggy)
((7,Robert),robert)
((8,Syam),syam)
((9,Mary),mary)
((10,Saran),saran)
((11,Stacy),stacy)
((12,Kelly),kelly)((,),)

REPLACE ( )

grunt> replace_data = FOREACH s GENERATE
(id,city),REPLACE(city,'Bhuwaneshwar','Bhuw'); grunt> dump replace_data; Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp962592513"
Job DAG:
job_local1191915317_0014
((1,newyork),newyork)
((2,Kolkata),Kolkata)
((3,Tokyo),Tokyo)
((4,London),London)
((5,Bhuwaneshwar),Bhuw)
((6,Chennai),Chennai)
((7,newyork),newyork)
((8,Kolkata),Kolkata)
((9,Tokyo),Tokyo)
((10,London),London)
((11,Bhuwaneshwar),Bhuw)
((12,Chennai),Chennai)
((,),)

Trim()

grunt> trim_data = FOREACH s GENERATE (id,name), TRIM(name);
grunt> dump trim_data;
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
Input(s):
Successfully read records from: "/home/cloudera/8044_pig/emp.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp684888790/tmp1627564471"
Job DAG:
job_local1613557141_0015
((1,Robin),Robin)
((2,BOB),BOB)

((3,Maya),Maya)
((4,Sara),Sara)
((5,David),David)
((6,Maggy),Maggy)
((7,Robert),Robert)
((8,Syam),Syam)
((9,Mary),Mary)
((10,Saran),Saran)
((11,Stacy),Stacy)
((12,Kelly),Kelly)((,),)

Week - 8 & 9 (HIVE)

Databases, Tables, Views, Functions, and Indexes

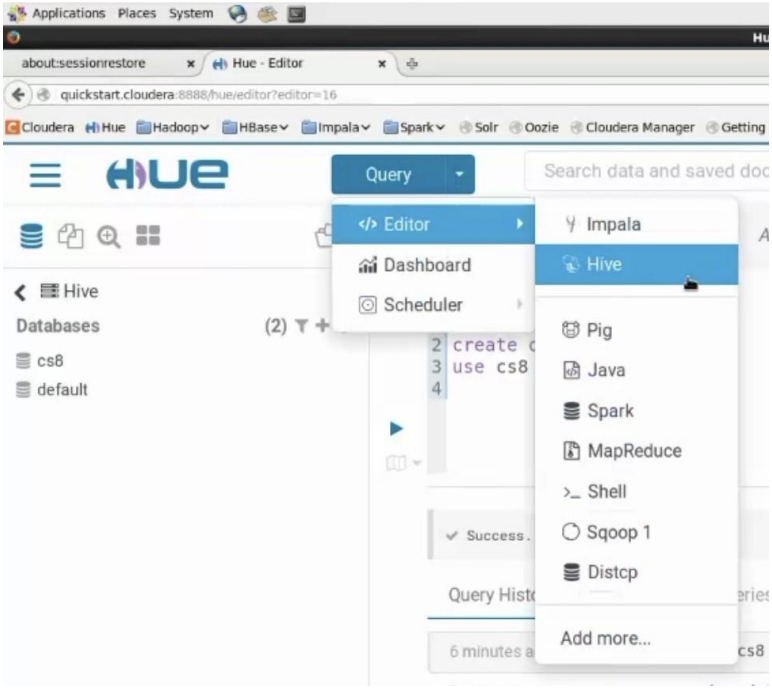
1. CREATE 2. SHOW 3. DESCRIBE 4. USE 5. DROP 6. ALTER 7. TRUNCATE

Table-1 Hive DDL commands

DDL Command	Use With
CREATE	Database, Table
SHOW	Databases, Tables, Table Properties, Partitions, Functions, Index
DESCRIBE	Database, Table, view
USE	Database
DROP	Database, Table
ALTER	Database, Table
TRUNCATE	Table

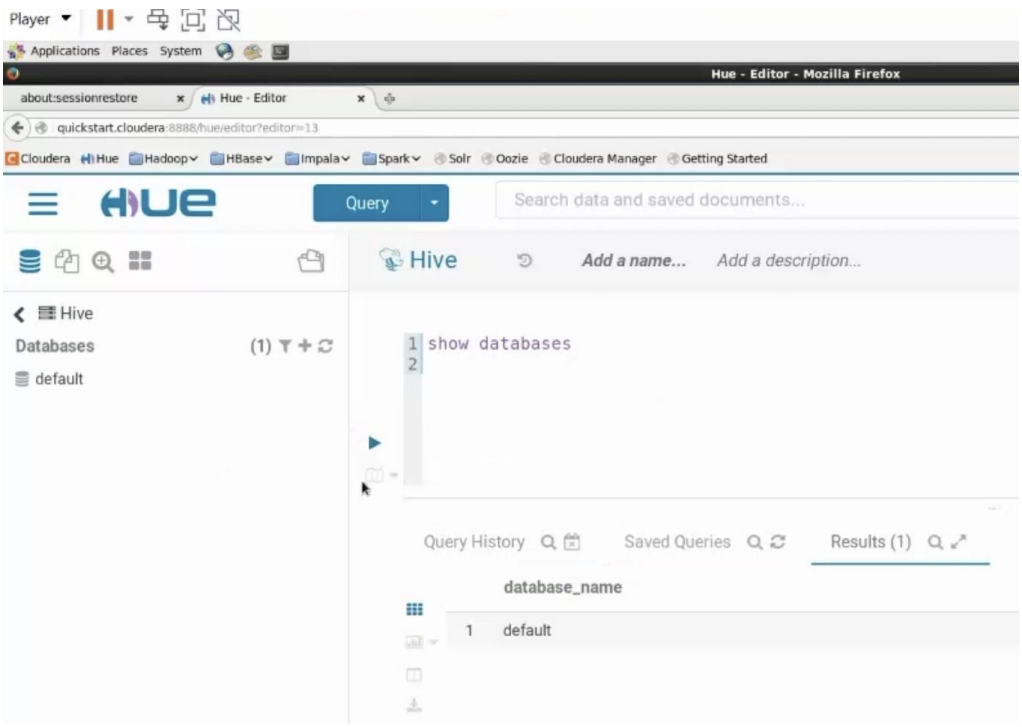
Before moving forward, note that the Hive commands are **case-insensitive**.
CREATE DATABASE is the same as create database.

1) Open Hue ->Under Query -> Choose Editor -> Click on Hive



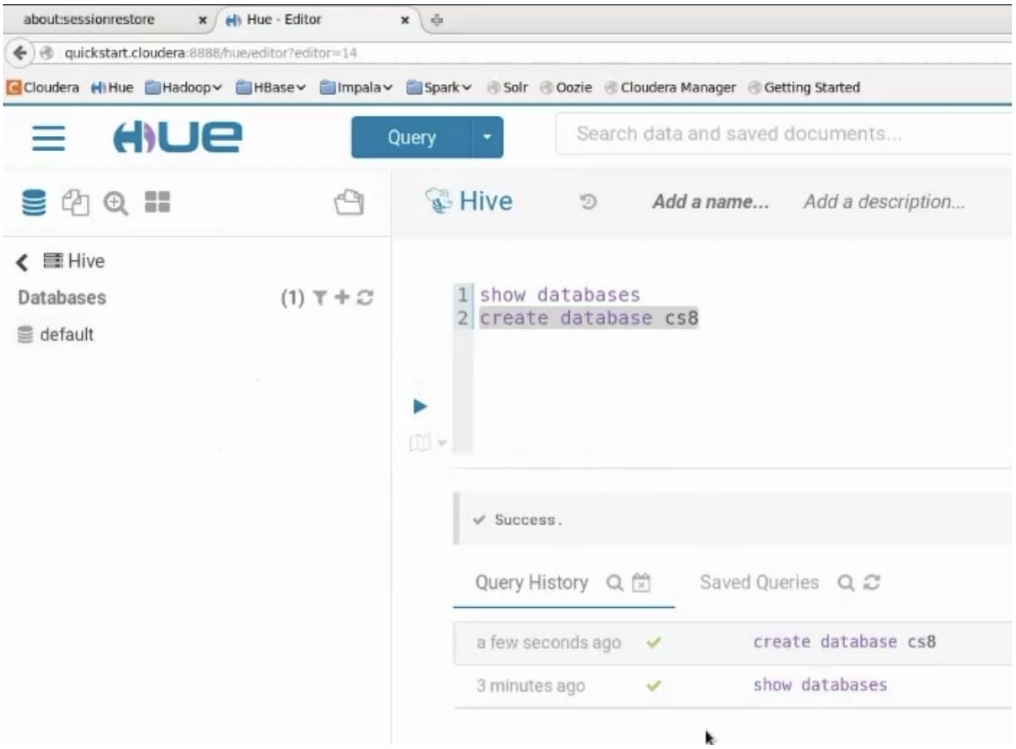
2) To Check Available Databases (Hive contains a default database)

SHOW DATABASES



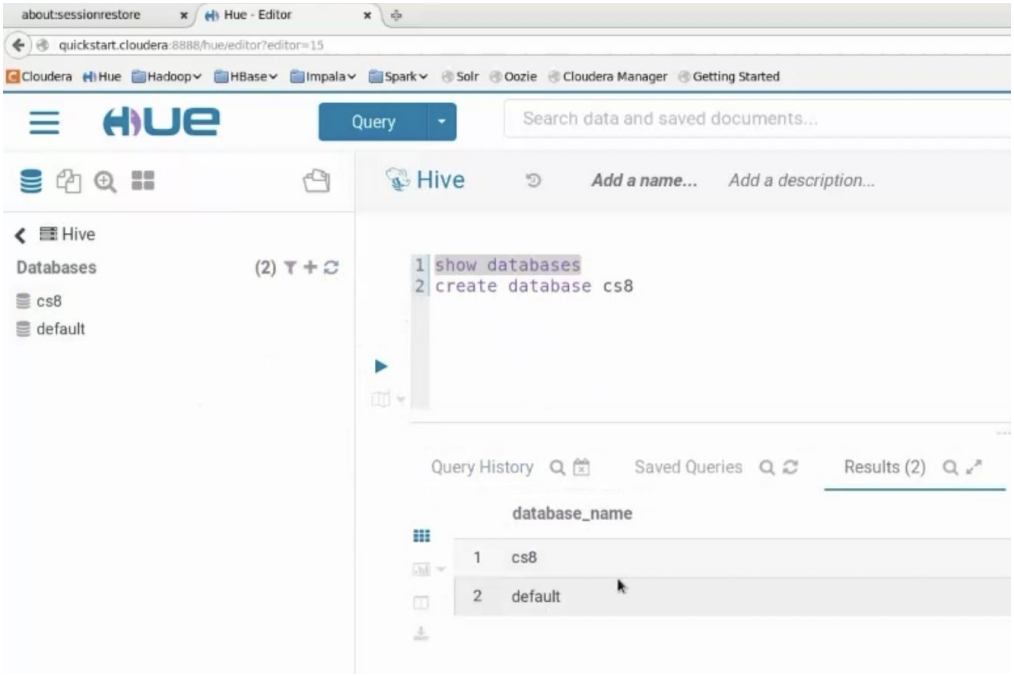
3) To Create a new Database

CREATE DATABASE CS8

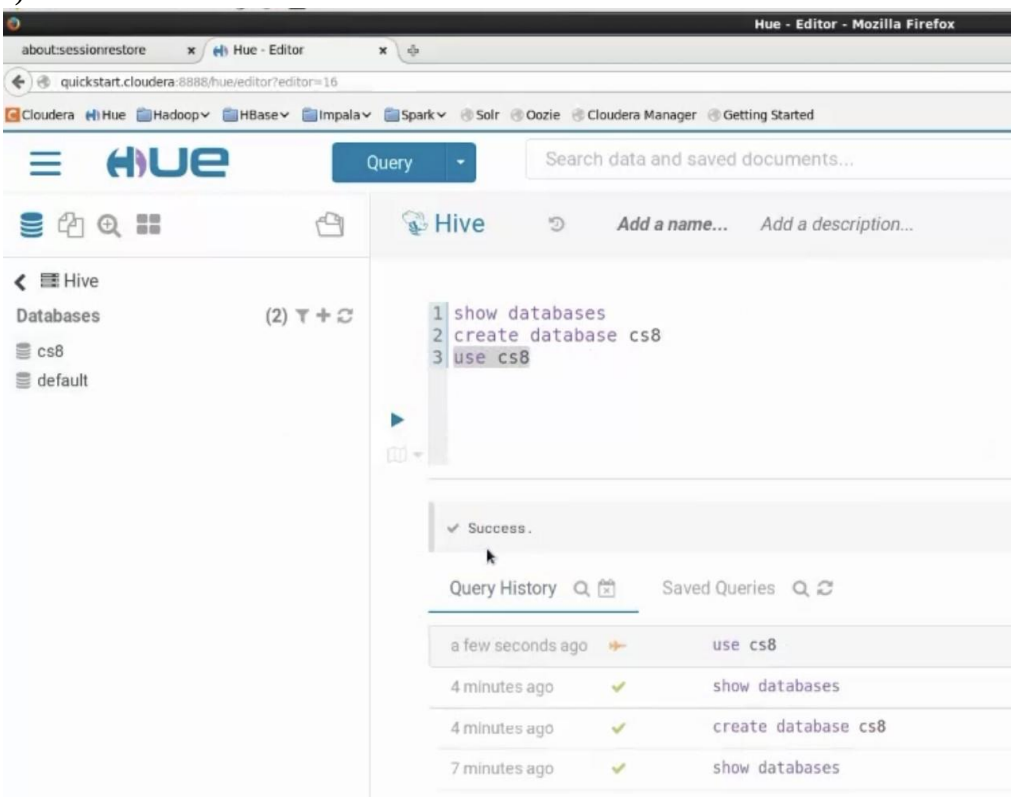


To verify the database available

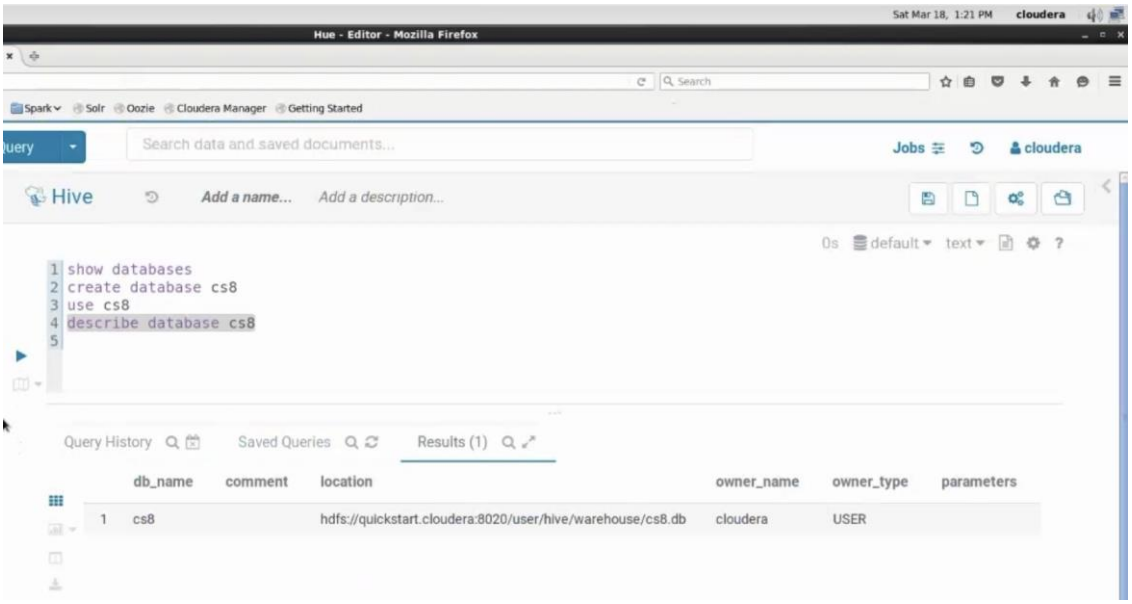
Run the first row again i.e., **SHOW DATABASES**



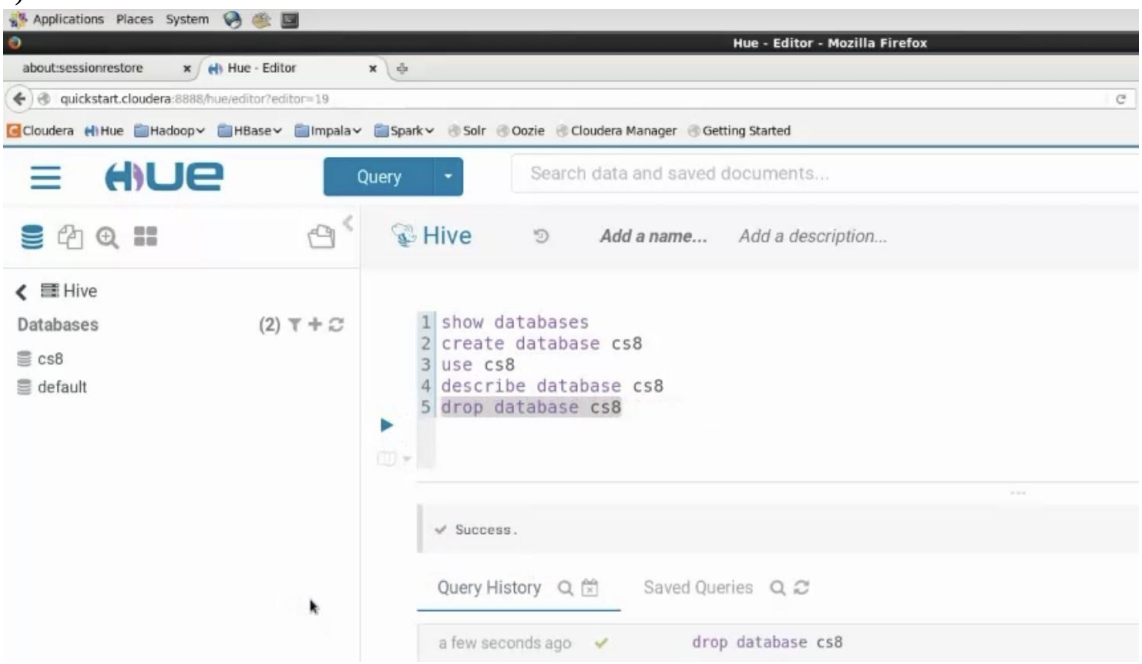
4) **USE CS8**



5) **DESCRIBE**



6) **DROP**

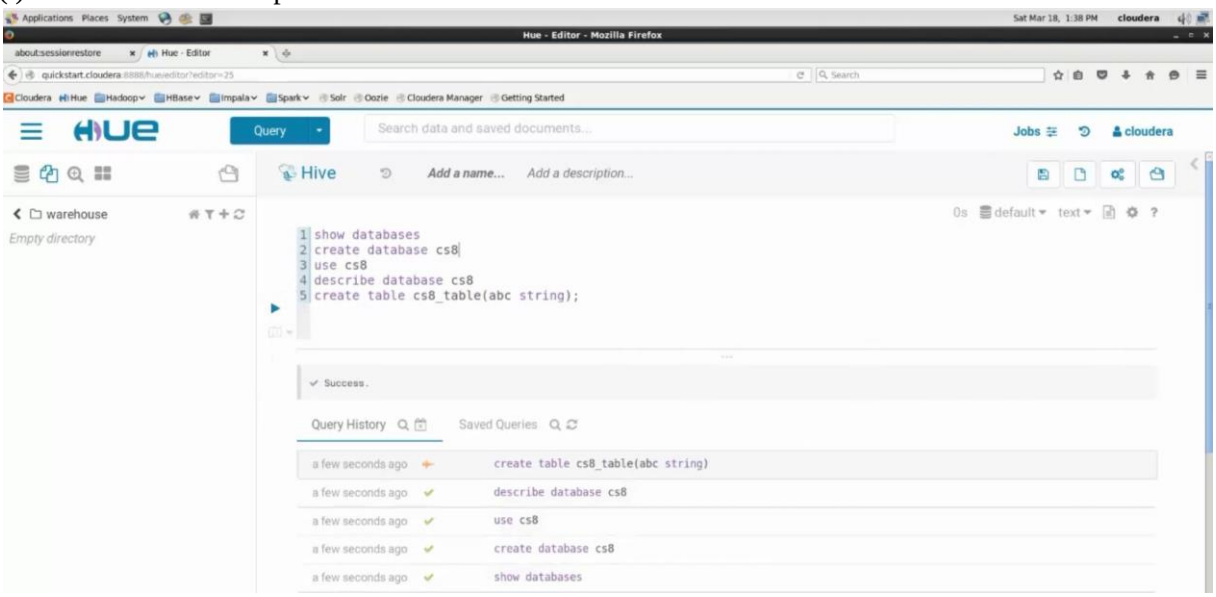


7) Before creation of a table, check the following path
/user/hive/warehouse/

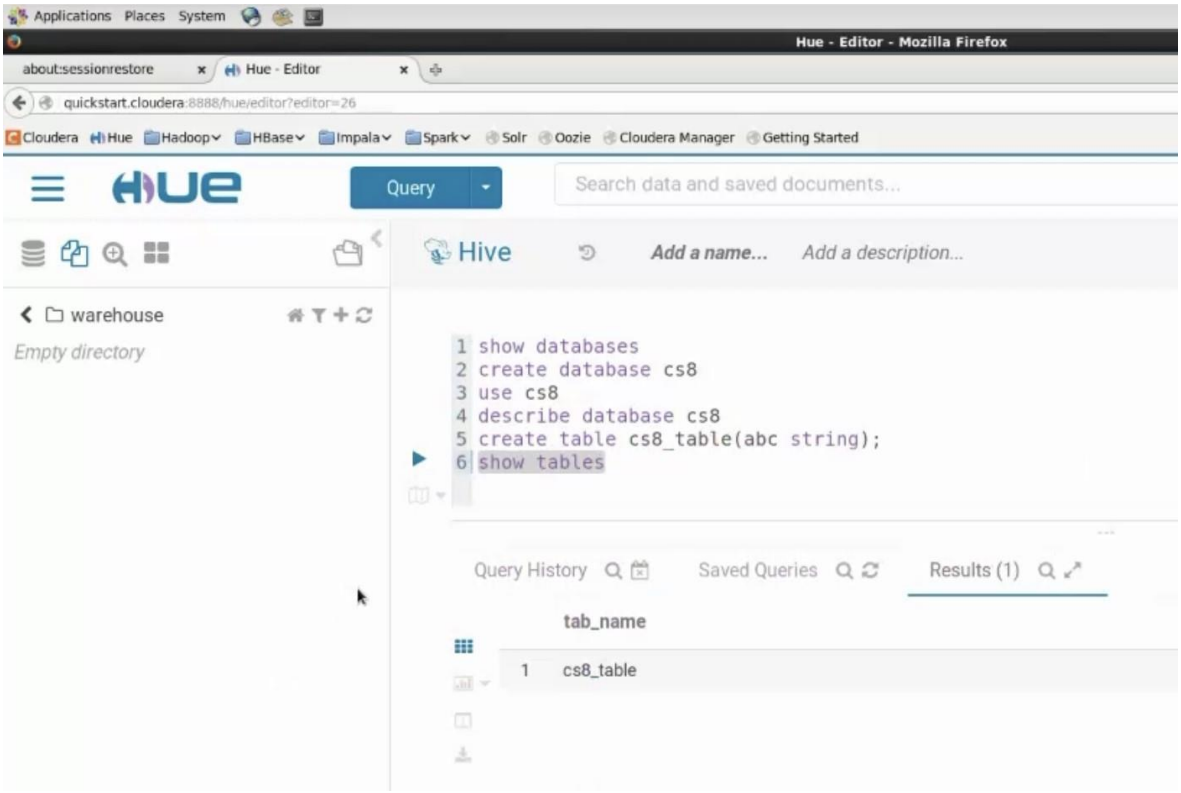
Tables can be created in two ways: Internal Table,also known as Managed Table and External Table.
a) Internal Table Syntax: **CREATE table table_name(schema);**
Example: **create table gitam_table(abc STRING);**
b) ExternalTableSyntax: **CREATE external table table_name(schema);**
Example: **create external table hyd_table(abc STRING);**

(since the database was previously dropped create new again in the /user/hive/warehouse/ path)

(i) Internal Table Example

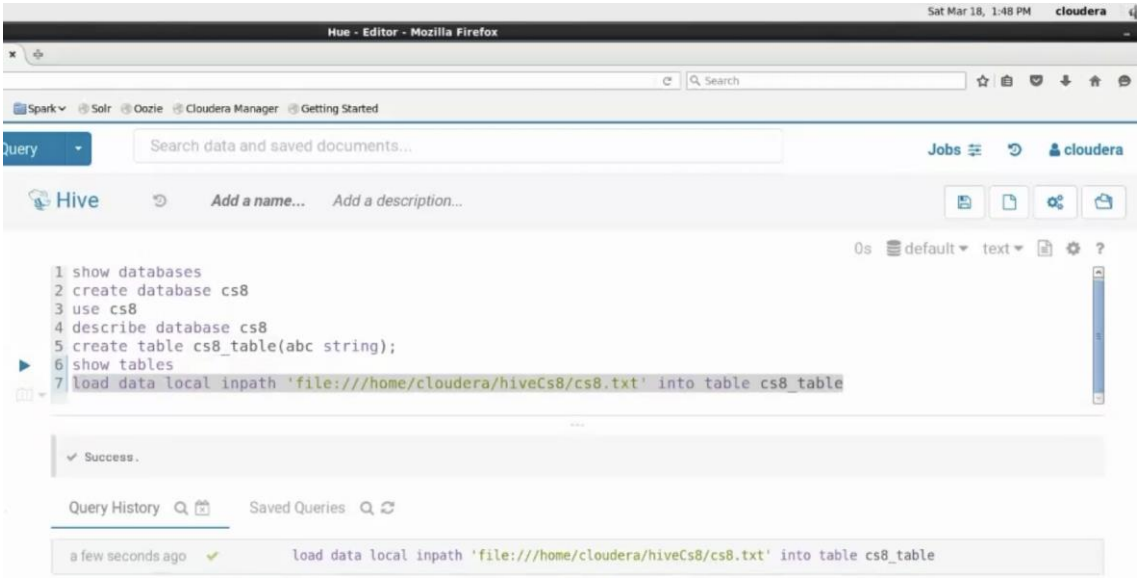


SHOW Tables



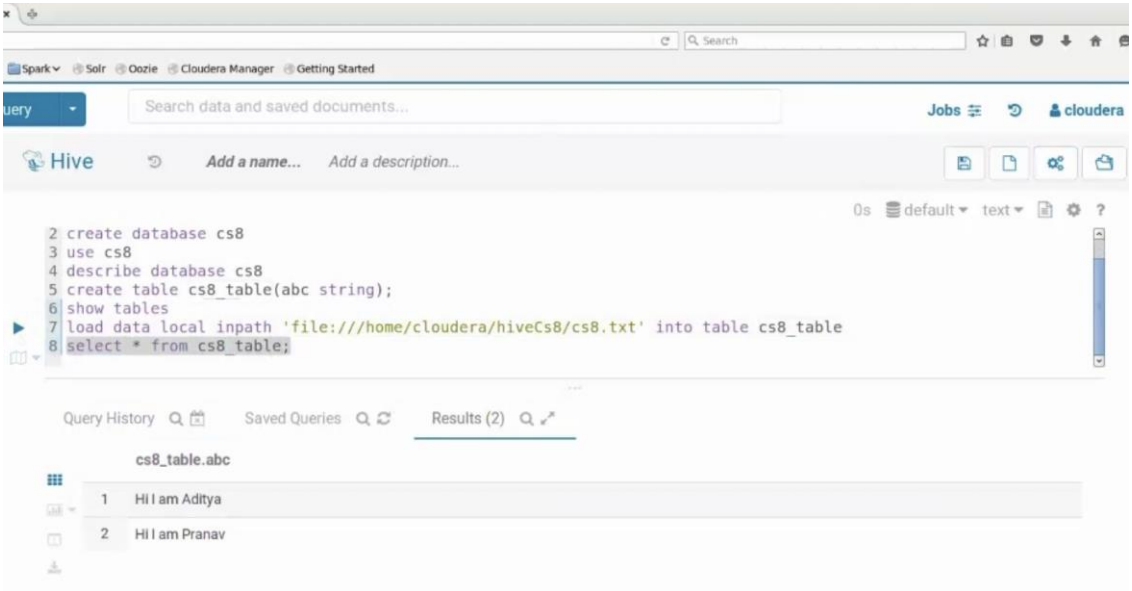
Create a text document in the folder in /home/cloudera/(folder name)/textfile name

Example: /home/cloudera/hive_practice/sample.txt (local file system) **load data local inpath 'file:///home/cloudera/hiveCs8/cs8.txt' into table cs8_table**



To check the content inside the table:

SELECT * FROM cs8_table



To check whether the table is created or not, check the following path
/user/hive/warehouse/...

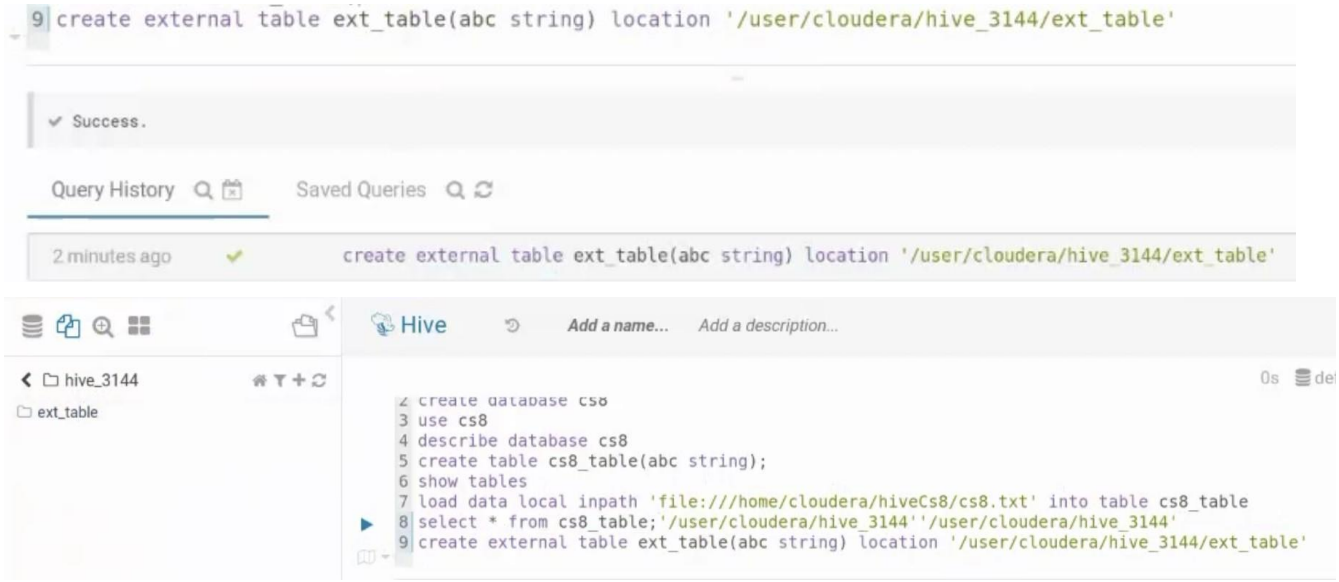
External table:

In an internal or managed table, the default path to store the table is /user/hive/warehouse/...

In the external table, we can store the table at our specified path.
To create a external table

create external table (table name)(abc STRING) location '/user/cloudera/(folder name)/(table name);

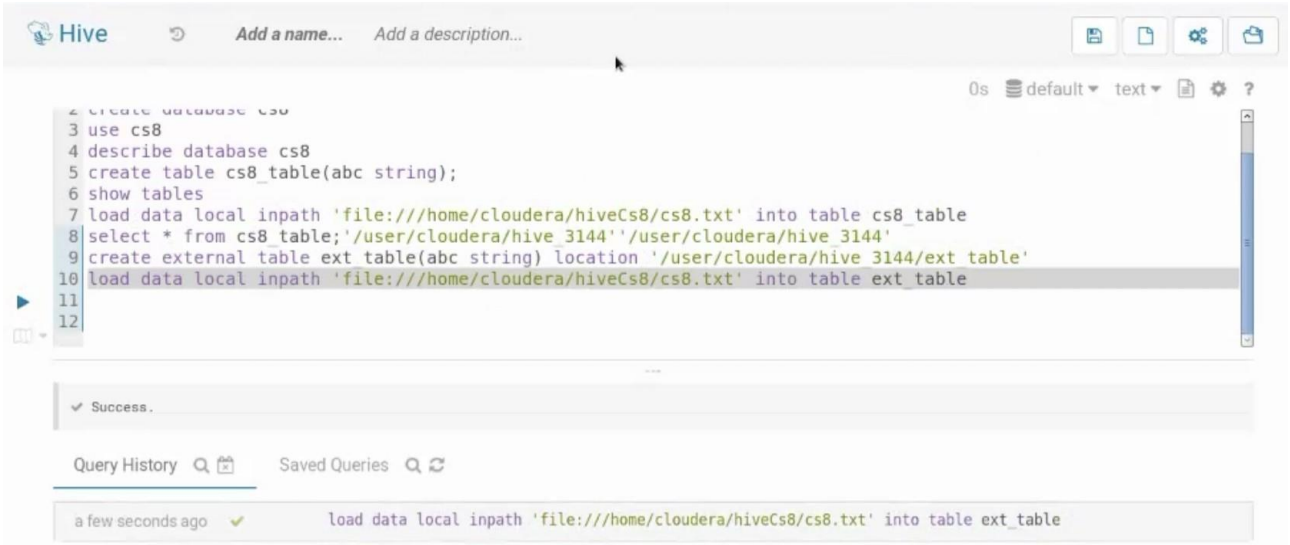
(It will automatically create a folder hive_3144, if it doesn't exist.) **create external table ext_table(abc string) location '/user/cloudera/hive_3144/ext_table'**



To load the data into the table

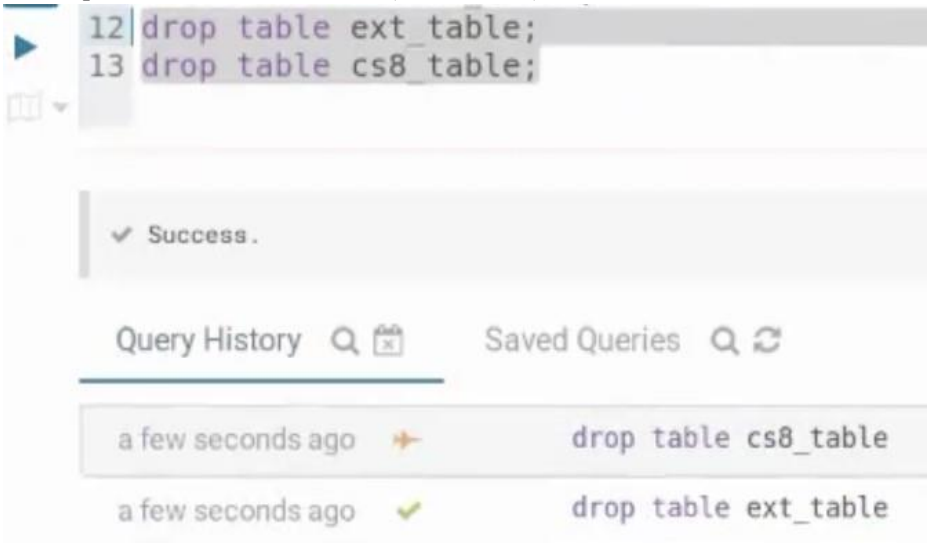
load data local inpath 'file:///home/cloudera/(folder name)/sample.txt' into table (table name);

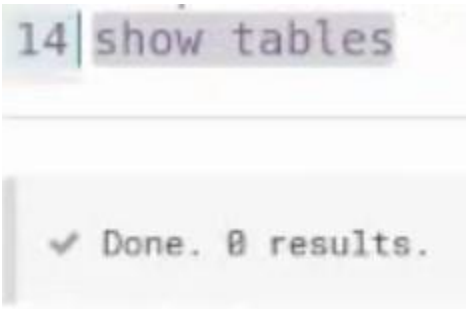
load data local inpath 'file:///home/cloudera/hiveCs8/cs8.txt' into table ext_table



To retrieve the records of the table **SELECT * FROM (table name)**

To Drop a table **DROP TABLE (table name)**





Problems :

1) Given the following table schema :

Employee_table {ID: INT, Name: Varchar (10), Age: INT, Salary: INT}

Loan_table {LoanID:INT, ID: INT, Loan_applied: Boolean, Loan_amt: INT)

a) Create a database and the following tables in Hive.

b) Insert records into the table.

c) Write an SQL to retrieve the employee details who have applied for a loan.

a. Create a database and the following tables in Hive. **create table if not exists employee_table(id int,name string,age int,salary int);**

```
1 create table if not exists employee_table(id int,name string,age int,salary int);
```

describe employee_table

```
1 create table if not exists employee_table(id int,name string,age int,salary int);
2 describe employee_table
```

Query History | Saved Queries | Results (4)

	col_name	data_type	comment
1	id	int	
2	name	string	
3	age	int	
4	salary	int	

create table if not exists loan_table(loan_id int,id int,loan_applied boolean,loan_amt int);
describe loan_table

```
1 create table if not exists employee_table(id int,name string,age int,salary int);
2 describe employee_table
3 create table if not exists loan_table(loan_id int,id int,loan_applied boolean,loan_amt int);
4 describe loan_table
```

Query History | Saved Queries | Results (4)

	col_name	data_type	comment
1	loan_id	int	
2	id	int	
3	loan_applied	boolean	
4	loan_amt	int	

b. Insert records into the table

(You can insert single or more values at once)

insert into employee_table values(1,'Aditya',22,2900000),(2,'Pranav',19,31000);
insert into employee_table values (3,'sid',5,230);

```
5 insert into employee_table values(1,'Aditya',22,2900000),(2,'Pranav',19,31000);
6 insert into employee_table values (3,'sid',5,230);
```

✔ Success.

Query History | Saved Queries

a minute ago	✔	insert into employee_table values (3,'sid',5,230)
3 minutes ago	✔	insert into employee_table values(1,'Aditya',22,2900000),(2,'Pranav',19,31000)

select * from employee_table;

```
7 select * from employee_table;
```

Query History | Saved Queries | Results (3)

	employee_table.id	employee_table.name	employee_table.age	employee_table.salary
1	1	Aditya	22	2900000
2	2	Pranav	19	31000
3	3	sid	5	230

insert into loan_table values(101,1,true,200),(102,2,false,3);
insert into loan_table values (103,3,true,2300); select * from loan_table;

```
 8 insert into loan_table values(101,1,true,200),(102,2,false,3);
 9 insert into loan_table values (103,3,true,2300);
10 select * from loan_table;
```

Query History | Saved Queries | Results (3)

	loan_table.loan_id	loan_table.id	loan_table.loan_applied	loan_table.loan_amt
1	101	1	true	200
2	102	2	false	3
3	103	3	true	2300

c. Write an SQL to retrieve the employee details who have applied for a loan. **select e.name,l.loan_id from employee_table e,loan_table l where l.id=e.id and l.loan_applied=true;**

```
11 select e.name,l.loan_id from employee_table e,loan_table l where l.id=e.id and l.loan_applied=true;
```

Query History | Saved Queries | Results (2)

	e.name	l.loan_id
1	Aditya	101
2	sid	103

select e.name from employee_table e,loan_table l where e.id=l.id and loan_applied=true;

```
12 select e.name from employee_table e,loan_table l where e.id=l.id and loan_applied=true;
```

Query History | Saved Queries | Results (2)

	e.name
1	Aditya
2	sid

2) Write a query to create a table which stores the employee records working in the same department together in the same sub-directory in HDFS. The schema for the table is given below:
Emp_table: {e_id, e_name, dept_id, yoj}
Dept_table: {dept_id, dept_name}

create table if not exists dept_hyd (dept_id int,dept_name varchar(20));
describe dept_hyd; insert into dept_hyd values
(901,'CSE'),(902,'MECH'),(903,'EEE'); select * from dept_hyd;

```
1 create table if not exists `dept_hyd` (`dept_id` int,`dept_name` varchar(20));
2 describe dept_hyd;
3 insert into dept_hyd values (901,'CSE'),(902,'MECH'),(903,'EEE');
4 select * from dept_hyd;
```

Query History | Saved Queries | Results (3)

	dept_hyd.dept_id	dept_hyd.dept_name
1	901	CSE
2	902	MECH
3	903	EEE

create table if not exists emp_hyd (id int,emp_name varchar(20),dept_id int,yoj int); insert into emp_hyd values (1, 'Pranav', 901, 2020),(2, 'Aditya', 901, 2020),(3, 'Sreekar', 902, 2021),(4, 'Bhavin', 903, 2021),(5, 'Vinieth', 903, 2021);
select * from emp_hyd;

```
6 create table if not exists emp_id (id int,emp_name varchar(20),dept_id int,yoj int);
7 insert into emp_id values (1, 'Pranav', 901, 2020),(2, 'Aditya', 901, 2020),(3, 'Sreekar', 902, 2021),(4, 'Bha
8 select * from emp_id;
```

Query History | Saved Queries | Results (5)

	emp_id.id	emp_id.emp_name	emp_id.dept_id	emp_id.yoj
1	1	Pranav	901	2020
2	2	Aditya	901	2020
3	3	Sreekar	902	2021
4	4	Bhavin	903	2021
5	5	Vinieth	903	2021

select e.emp_name,d.dept_name from emp_hyd e,dept_hyd d where d.dept_id=e.dept_id and d.dept_id=901;

```
12 select e.emp_name,d.dept_name from emp_hyd e,dept_hyd d where d.dept_id=e.dept_id and d.dept_id=901;
```

Query History | Saved Queries | Results (2)

	e.emp_name	d.dept_name
1	Pranav	CSE
2	Aditya	CSE

select e.emp_name,d.dept_name from emp_hyd e,dept_hyd d where d.dept_id=e.dept_id and d.dept_id=902;

```
12 select e.emp_name,d.dept_name from emp_hyd e,dept_hyd d where d.dept_id=e.dept_id and d.dept_id=902;
```

Query History | Saved Queries | Results (1)

	e.emp_name	d.dept_name
1	Sreekar	MECH

3) Given

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

+-----+---------------------+-------------+--------+

|OID | DATE | CUSTOMER_ID | AMOUNT |

Create the following table in hive and insert transaction records into it. write a SQL query to find the customer details who have made an order?

create table if not exists customer_hyd (customer_id int, name varchar(20), age int, address varchar(20), salary int); describe customer_hyd; insert into customer_hyd values (1, ‘Aditya’, 22, ‘Hyderabad’, 122323), (2, ‘Pranav’, 22,

'Hyderabad', 4454543),(3, 'Sreekar', 29, 'Beeramguda', 123233);
select * from customer_hyd;

```
create table if not exists `customer_hyd` (`customer_id` int,`name` varchar(20),`age` int,`address` varchar(20),`salary` int);
describe customer_hyd;
insert into customer_hyd values (1,'Aditya',22,'hyd',1234567),(2,'Pranav',22,'hyd',654321),(3,'Sheik',29,'Beeramguda',10021);
select * from customer_hyd;

create table if not exists `order_hyd1` (`oid` int,`date` varchar(20),`customer_id` int,`amount` int);
```

Query History | Saved Queries | Results (3)

	customer_hyd.customer_id	customer_hyd.name	customer_hyd.age	customer_hyd.address	customer_hyd.salary
1	1	Aditya	22	hyd	1234567
2	2	Pranav	22	hyd	654321
3	3	Sheik	29	Beeramguda	10021

create table if not exists order_hyd1 (oid int, date varchar(20), customer_id int, amount int); insert into order_hyd1 values (201, '10-11-2020', 1, 2000), (202, '15-4-2021', 2, 9000), (203, '27-9-2022', 3, 10000), (204, '11-11-2023', 1 , 20012);
select * from order_hyd1;

```
create table if not exists `order_hyd1` (`oid` int,`date` varchar(20),`customer_id` int,`amount` int
insert into order_hyd1 values (201,'10-11-2020',1,2000),(202,'15-4-2021',2,9000),(203,'27-9-2022',3,10000),(204,'11-11-2023',1,20012);
select * from order_hyd1;
```

Query History | Saved Queries | Results (4)

	order_hyd1.oid	order_hyd1.date	order_hyd1.customer_id	order_hyd1.amount
1	201	10-11-2020	1	2000
2	202	15-4-2021	2	9000
3	203	27-9-2022	3	10000
4	204	11-11-2023	1	20012

select c.name,o.oid from customer_hyd c,order_hyd1 o where c.customer_id=o.customer_id and o.oid=201;

```
select c.name,o.oid from customer_hyd c,order_hyd1 o where c.customer_id=o.customer_id and o.oid=201;
```

Query History | Saved Queries | Results (1)

	name	oid
1	Aditya	201

Week - 10 (Spark)

Understanding Spark

The Spark program is written in terms of operations on RDDs. RDD is partitioned and distributed to cluster nodes. RDD can be stored on disk or memory. RDDs are manipulated using a set of parallel transformations and actions. Spark keeps track of how RDDs are created so that it can be rebuilt in case of job failures or in case of slow workers.

What is RDD?

- RDD is a fundamental data structure of spark.
- Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.
- Spark context is used to create RDDs.
- RDDs are immutable i.e, once created can not be changed. Changes made to RDD can only be stored in another RDD.

RDDs can be constructed in 3 ways

1) Create from python list

#create python
listlst=range(5,20)
lst
out[]: [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
parallelize(list, #partitions)
#create RDD using python list.
list_rdd1=sc.parallelize(lst,3)
print type(list_rdd1)
<class 'pyspark.rdd.RDD'>

2) By transforming the existing RDD

#to copy one RDD to another RDD
rdd2=list_rdd1

3) By reading from a file #to read from a file r3=sc.textFile(‘/user/cloudera/m r/file1’)
sc.textFile(‘file:///home/cloudera/path/’)

Operations on RDDs:

We can perform two types of operations on RDDs namely transformations and actions.

- Apache Spark Transformation is a function that produces new RDD from the existing RDDs.
- In other words, transformations are functions that take a RDD as the input and produce one or many RDDs as the output.
- Transformations are lazy i.e. they are not computed immediately but executed only when action is run on it.

• Spark remembers the set of transformations applied on the base dataset, apply the optimizations and execute when action is applied. • This also helps in automatic recovery from failed or slow machines.

Actions : Execute the transformations and get the data from the workers to the driver.

The following table gives a list of Actions, which return values.

S.No	Action & Meaning
1	**first()**
2	**take(n)**
3	**collect()**
4	**count()**
5	**reduce(func)**
7	**takeOrdered(n, [ordering])**

[cloudera@quickstart 8044]$ pyspark

Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
/__ / .__/\_,_/_/ /_/\_\   version 1.6.0
   /_/
```

>>> lst=range(5,20)

>>> lst

[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>> list_rdd1=sc.parallelize(lst,3)

>>> print type(list_rdd1)

<class 'pyspark.rdd.RDD'>

>>> rdd2=list_rdd1

>>> print type(rdd2)

<class 'pyspark.rdd.RDD'>

>>> r3=sc.textFile('/use/cloudera/8044/m1.txt')

>>> print type(r3)

<class 'pyspark.rdd.RDD'>

#first() returns the first element of the RDD.
>>> list_rdd1.first()
5

#take will display the first N elements of the RDD.
>>> list_rdd1.take(3)
[5, 6, 7]

#collect action will get all the elements of the RDD to the driver
>>> rdd2.collect()
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

 # count action counts the number of elements in the RDD
>>> rdd2.count()
15

#reduce action reduces to a single value by applying a function
>>> x=rdd2.reduce(lambda i,j:i+j)
>>> print type(x),x
<type 'int'> 180

S.No	Transformations
1	distinct([numTasks])
2	filter(func)
3	map(func)
4	flatMap(func)
5	union(otherDataset)
6	intersection(otherDataset)
7	groupByKey([numTasks])
8	reduceByKey(func, [numTasks])
9	join(otherDataset, [numTasks])
10	cartesian(otherDataset)

 # create a list
>>>lst=[('modi','pm'),('babu','cm'),('naidu','vp'),('kcr','cm'),('narasimham','governor'),('narasimham','governor'),('babu','cm')]

>>> rdd_lst=sc.parallelize(lst)

 # convert list to rdd

>>> rdd_lst.collect()

[('modi', 'pm'), ('babu', 'cm'), ('naidu', 'vp'), ('kcr', 'cm'), ('narasimham', 'governor'), ('narasimham', 'governor'), ('babu', 'cm')]

#distinct transformation is used to select unique elements of RDD

>>> distint_rddlist=rdd_lst.distinct()

>>> distint_rddlist.collect()

[('kcr', 'cm'), ('modi', 'pm'), ('babu', 'cm'), ('narasimham', 'governor'), ('naidu', 'vp')]

#filter transformation is used to select the elements satisfying the #function

>>> filter_rddlist=rdd_lst.filter(lambda element : element[1]=='cm')

>>> filter_rddlist.collect()

[('babu', 'cm'), ('kcr', 'cm'), ('babu', 'cm')]

#map: Returns a new distributed dataset, formed by passing each element of the source through a function

>>> map_rddlist = rdd_lst.map(lambda (x,y):(y,x))

>>> map_rddlist.collect()

[('pm', 'modi'), ('cm', 'babu'), ('vp', 'naidu'), ('cm', 'kcr'), ('governor', 'narasimham'), ('governor', 'narasimham'), ('cm', 'babu')]

#SortByKey: When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the Boolean ascending argument. #sort the input RDD by the key value

>>> sortbykey_rdd = map_rddlist.sortByKey()

>>> sortbykey_rdd.collect()

[('cm', 'babu'), ('cm', 'kcr'), ('cm', 'babu'), ('governor', 'narasimham'), ('governor', 'narasimham'), ('pm', 'modi'), ('vp', 'naidu')]

#groupByKey(): When called on a dataset of (K, V) pairs, returns a dataset of #(K, Iterable<V>) pairs. #Group the values for each key in the RDD into a single sequence

>>> groupbykey_rdd= map_rddlist.groupByKey()

>>> groupbykey_rdd.collect()

[('vp', <pyspark.resultiterable.ResultIterable object at 0x1c03310>), ('cm',

<pyspark.resultiterable.ResultIterable object at 0x1c032d0>), ('governor',

<pyspark.resultiterable.ResultIterable object at 0x1bfdf10>), ('pm', <pyspark.resultiterable.ResultIterable object at 0x1bfdfd0>)]

#reduceByKey() : When called on a dataset of (K, V) pairs, returns a dataset of #(K, V) pairs where the values for each key are aggregated using the given reduce #function *func*, which must be of type

(V, V) ⇒V. #Merge the values for each key

>>> **reducebykey_rdd = map_rddlist.reduceByKey(lambda x,y:x+y)**

>>> **reducebykey_rdd.collect()**

[('vp', 'naidu'), ('cm', 'babukcrbabu'), ('governor', 'narasimhamnarasimham'), ('pm', 'modi')]

>>> **rdd1=sc.parallelize(range(5,10))**

>>> **rdd2=sc.parallelize(range(8,15))**

>>> **list1=[1,2,3,4,5]**

>>> **rdd3 = sc.parallelize(list1)**

>>> **rdd1.collect()**

[5, 6, 7, 8, 9]

>>> **rdd2.collect()**

[8, 9, 10, 11, 12, 13, 14]

>>> **rdd3.collect()**

[1, 2, 3, 4, 5]

#flatMap transformation is similar to map, but each input item can be mapped #to 0 or more output items

>>> **flatmap_rdd3=rdd3.flatMap(lambda i : (i,i+5,i*5))**

>>> **flatmap_rdd3.collect()**

[1, 6, 5, 2, 7, 10, 3, 8, 15, 4, 9, 20, 5, 10, 25]

#takeOrdered : Get the N elements from a RDD ordered in ascending order or as specified by #the optional key function.

>>> **flatmap_rdd3.takeOrdered(7)**

[1, 2, 3, 4, 5, 5, 6]

#to order in descending order

>>> **flatmap_rdd3.takeOrdered(flatmap_rdd3.count(),lambda x: -x)**

[25, 20, 15, 10, 10, 9, 8, 7, 6, 5, 5, 4, 3, 2, 1]

#Union and Intersection

 # union returns a new RDD that contains the union of the elements in the source dataset and the argument and it includes duplicate elements also

>>> **rddunion=rdd1.union(rdd2)**

>>> **rddunion.collect()**

[5, 6, 7, 8, 9, 8, 9, 10, 11, 12, 13, 14]

#intersection returns a new RDD that contains the intersection of elements in the source dataset and the argument.

>>> **rddintersection=rdd1.intersection(rdd2)**

>>> **rddintersection.collect()**

[8, 9]

 # Cartesian: When called on datasets of types T and U, returns a dataset of

 # (T, U) pairs (all pairs of elements)

>>> **rddcartesian=rdd1.cartesian(rdd2)**

>>> **rddcartesian.collect()**

[(5, 8), (5, 9), (5, 10), (5, 11), (5, 12), (5, 13), (5, 14), (6, 8), (6, 9), (6, 10), (6, 11), (6, 12), (6, 13), (6, 14), (7, 8), (7, 9), (7, 10), (7, 11), (7, 12), (7, 13), (7, 14), (8, 8), (8, 9), (8, 10), (8, 11), (8, 12), (8, 13), (8, 14), (9, 8), (9, 9), (9, 10), (9, 11), (9, 12), (9, 13), (9, 14)]

#JOIN: When called on datasets of type (K, V) and (K, W), returns a dataset of

 # (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.

>>> **rdd1=sc.parallelize([(1,8,5),(2,7,7),(3,9),(5,6,5)])**

>>> **rdd2=sc.parallelize([(1,'cse'),(3,'ece'),(4,'me'),(5,'ece')])**

>>> **rddjoin=rdd1.join(rdd2)**

>>> **rddjoin.collect()**

[(1, (8, 'cse')), (3, (9, 'ece')), (5, (6, 'ece'))]

#leftOuterJoin: performs a join starting with the first (left-most) RDD and #then any matching second (right-most) RDD elements.

>>> **rddleftouter=rdd1.leftOuterJoin(rdd2)**

>>> **rddleftouter.collect()**

[(2, (7, None)), (1, (8, 'cse')), (3, (9, 'ece')), (5, (6, 'ece'))]

#rightOuterJoin: performs a join starting with the second (right-most) RDD and then any matching first (left-most) RDD elements.

>>> **rddrightouter=rdd1.rightOuterJoin(rdd2)**

>>> **rddrightouter.collect()**

[(4, (None, 'me')), (1, (8, 'cse')), (3, (9, 'ece')), (5, (6, 'ece'))]

#fullOuterJoin: returns all matching elements from both RDDs whether the other RDD matches or not.

>>> **rddfullouter=rdd1.fullOuterJoin(rdd2)**

>>> **rddfullouter.collect()**

[(2, (7, None)), (4, (None, 'me')), (1, (8, 'cse')), (3, (9, 'ece')), (5, (6, 'ece'))]

Map Reduce in Spark

>>> **x = sc.textFile("/user/cloudera/spark_cs8/wc.txt")**

>>> **x.collect()**

[u'Hi this is Aditya', u'Hello I am Pranav', u'Hello there siddharth', u'Hi again']

>>> **a = x.flatMap(lambda line:line.split(" "))**

>>> **a.collect()**

[u'Hi', u'this', u'is', u'Aditya', u'Hello', u'I', u'am', u'Pranav', u'Hello', u'there', u'siddharth', u'Hi', u'again']

>>> **b= a.map(lambda word:(word,1))**

>>> **b.collect()**

[(u'Hi', 1), (u'this', 1), (u'is', 1), (u'Aditya', 1), (u'Hello', 1), (u'I', 1), (u'am', 1), (u'Pranav', 1), (u'Hello', 1), (u'there', 1), (u'siddharth', 1), (u'Hi', 1), (u'again', 1)]

>>> **c = b.reduceByKey(lambda a,b:a+b)**

>>> **c.collect()**

[(u'Pranav', 1), (u'again', 1), (u'this', 1), (u'Aditya', 1), (u'is', 1), (u'there', 1), (u'am', 1), (u'I', 1), (u'Hi', 2), (u'siddharth', 1), (u'Hello', 2)]

>>> c.saveAsTextFile("user/cloudera/spark_cs8") (Doesn't work)