

Aim: To Install Cloudera Quickstart VM on VirtualBox

Cloudera is a software that provides a platform for data analytics, data warehousing, and machine learning. Initially, Cloudera started as an open-source Apache Hadoop distribution project, commonly known as Cloudera Distribution for Hadoop or CDH. It contains Apache Hadoop and other related projects where all the components are 100% open-source under Apache License.

Cloudera provides virtual machine images of complete Apache Hadoop clusters, making it easy to get started with Cloudera CDH. The following topics will be covered in this assignment on Cloudera QuickStart VM Installation.

1. What is Cloudera QuickStart VM?
2. Cloudera QuickStart VM Installation - Prerequisites
3. Downloading the Cloudera QuickStart VM
4. Cloudera QuickStart VM Installation on windows

What Is Cloudera QuickStart VM?

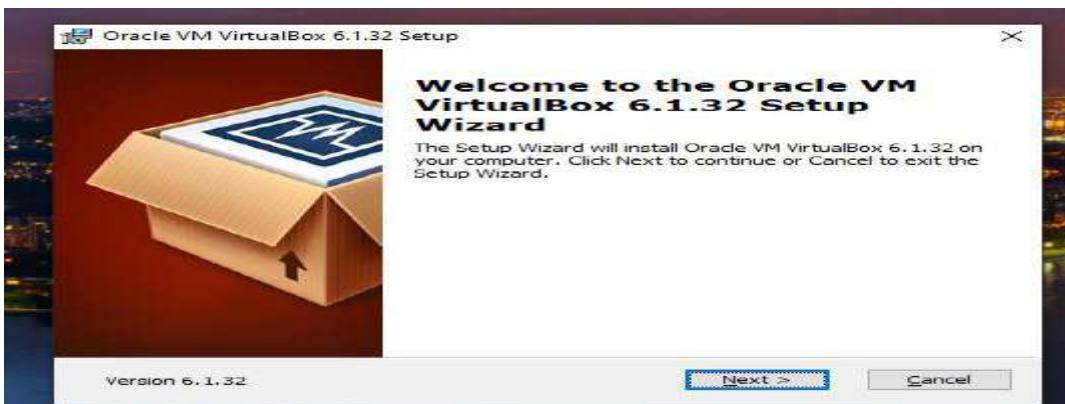
Cloudera QuickStart VM includes everything that you would need for using CDH, Impala, Cloudera Search, and Cloudera Manager. The Cloudera QuickStart VM uses a package-based install that allows you to work with or without the Cloudera Manager. It has a sample of Cloudera's platform for "Big Data."

Cloudera QuickStart VM Installation - Prerequisites

A virtual machine such as Oracle Virtual Box or VMWare RAM of 12+ GB. That is 4+ GB for the operating system and 8+ GB for Cloudera

80GB hard disk

Download Oracle Virtual Box from <https://www.virtualbox.org/wiki/Downloads> and install it in your system



Link: <https://www.cloudera.com/downloads.html>

A screenshot of the Cloudera Downloads page. At the top, a banner states "Effective Jan 31, 2021, all Cloudera software requires a subscription." Below the banner, there are three main download sections: "Cloudera CDH" (with "Free Trial" and "Download now" buttons), "Hortonworks Data Platform (HDP)" (with "Download now" buttons for enterprise, developer, and open source editions), and "Cloudera Workload XM" (with "Download now" button). At the bottom, there is a section for "Cloudera DataFlow (Ambari)" with a note that it is "formerly Hortonworks".

If already an account, then login else first register the account

CLOUDERA

Please Log In

Email Address*



Password*

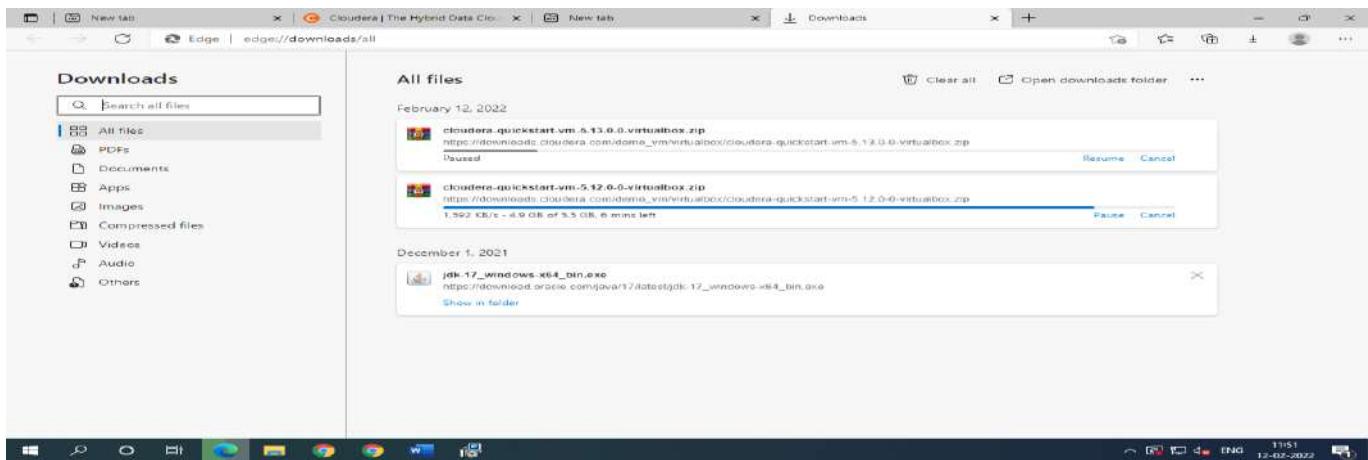


Log In

Cloudera uses cookies to provide and improve our site's services.
By using this site, you consent to use of cookies as outlined in Cloudera's Privacy and Data Policies.

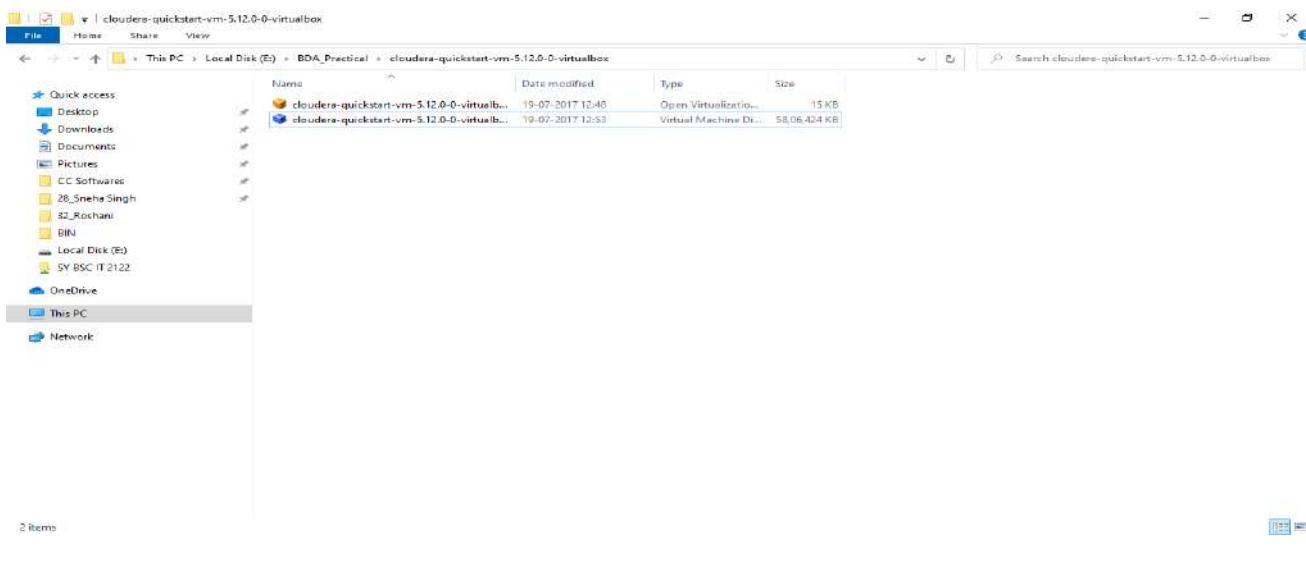
Accept Cookies

cloudera download link –

https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.13.0-0-virtualbox.zip


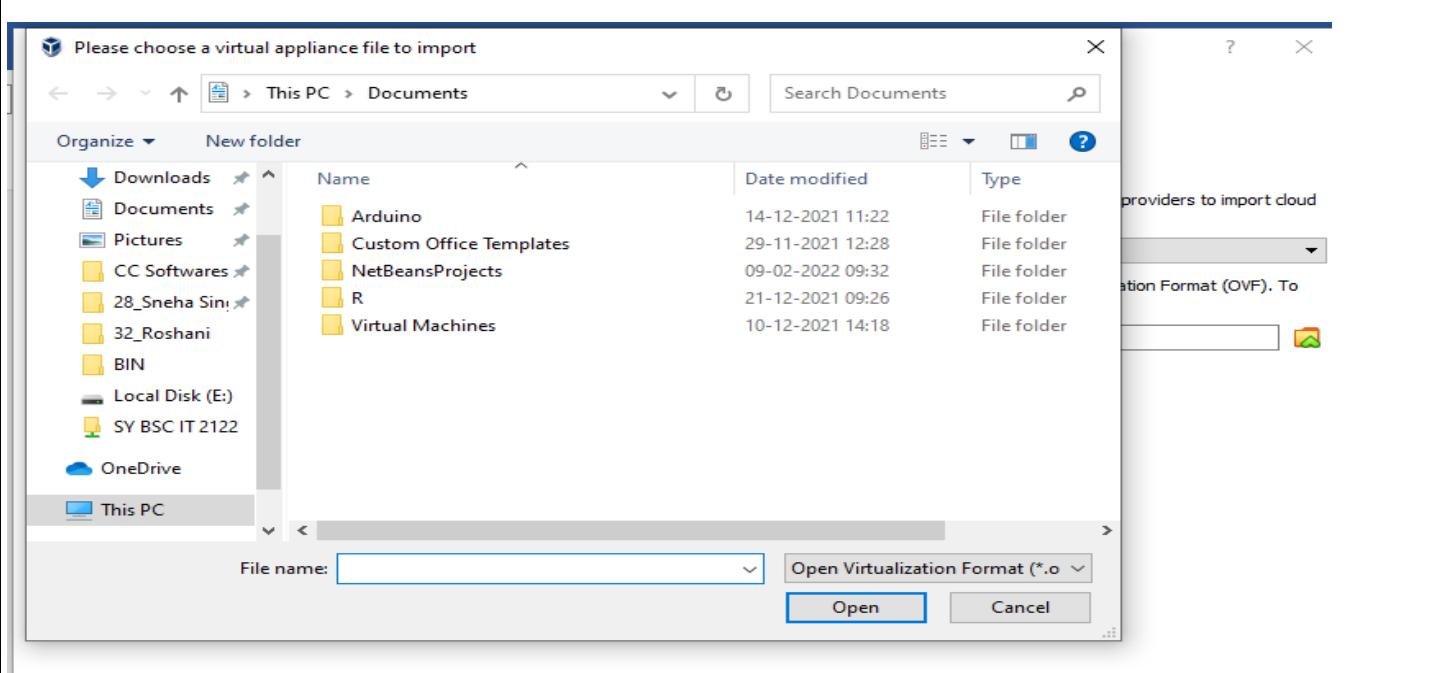
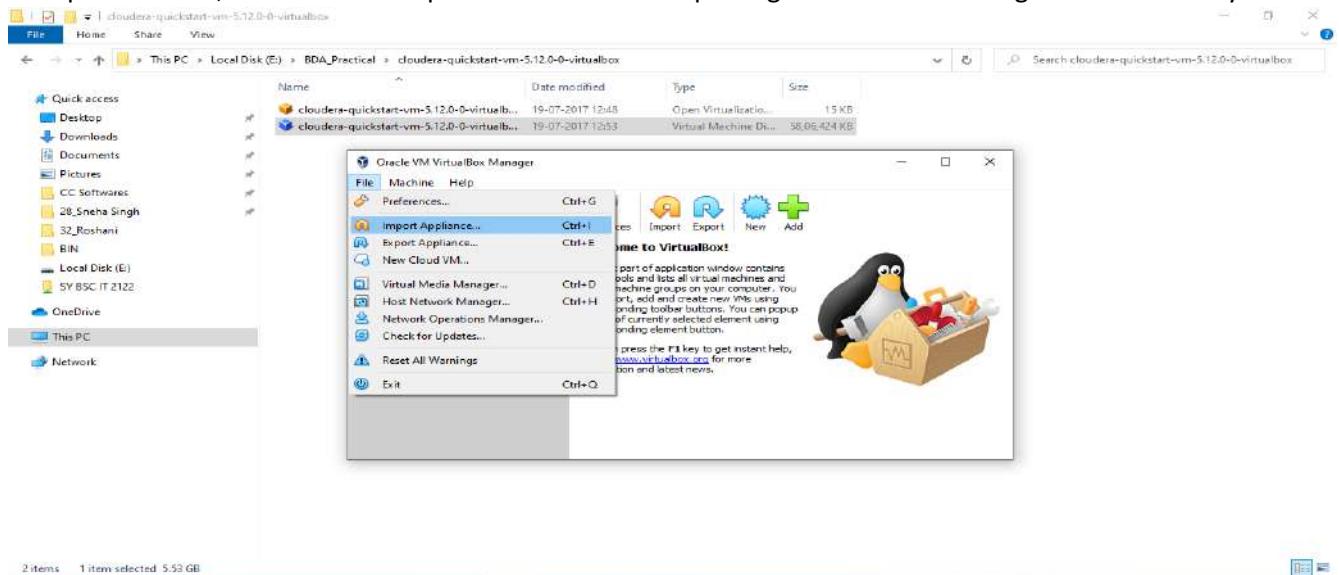
Downloading the Cloudera QuickStart VM

- The Cloudera QuickStart VMs are openly available as Zip archives in VirtualBox, VMware and KVM formats. To download the VM, search for <https://www.cloudera.com/downloads.html> and select the appropriate version of CDH that you require.
- Click on the 'GET IT NOW' button, and it will prompt you to fill in your details.
- Once the file is downloaded, go to the download folder and unzip these files. It can then be used to set up a single node Cloudera cluster.
- Shown below are the two virtual images of Cloudera QuickStart VM.
- Now that the downloading process is done with, let's move forward with this Cloudera QuickStart VM Installation guide and see the actual process.



Cloudera QuickStart VM Installation

- Before setting up the Cloudera Virtual Machine, you would need to have a virtual machine such as VMware or Oracle VirtualBox on your system.
- In this case, we are using Oracle VirtualBox to set up the Cloudera QuickStart VM.
- In order to download and install the Oracle VirtualBox on your operating system, click on the following link: Oracle VirtualBox(<https://www.virtualbox.org/wiki/Downloads>).
- To set up the Cloudera QuickStart VM in your Oracle VirtualBox Manager, click on 'File' and then select 'Import Appliance'.
- Choose the QuickStart VM image by looking into your downloads. Click on 'Open' and then 'Next'. Now you can see the specifications, then click on 'Import'. This will start importing the virtual disk image .vmdk file into your VM box.



- Click on "Open" and Wait for a while, as the importing finishes.
- The next step is to go ahead and set up a Cloudera QuickStart VM for practice. Once the importing is complete, you can see the Cloudera QuickStart VM on the left side panel.

Name: Pavan Yadav

Practical 1

Please choose a virtual appliance file to import

Organize New folder

Name Date modified Type

cloudera-quickstart-vm-5.12.0-0-virtualbox... 19-07-2017 12:48 Open Virtualizat...

Downloads Documents Pictures CC Softwares 28_Sneha Sini 32_Roshani BIN Local Disk (E:) SY BSC IT 2122 OneDrive This PC

File name: cloudera-quickstart-vm-5.12.0-0-virtualbox... Open Virtualization Format (*.ovf) Open Cancel

Import Virtual Appliance

Appliance to import

Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.

Source: Local File System

Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

File: E:\BDA_Practical\cloudera-quickstart-vm-5.12.0-0-virtualbox\cloudera-quickstart-vm-5.12.0-0-virtualbox.ovf

Import Virtual Appliance

Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1	
Name	cloudera-quickstart-vm-5.12.0-0-virtualbox
Guest OS Type	Red Hat (64-bit)
CPU	1
RAM	4096 MB
DVD	<input checked="" type="checkbox"/>
Network Adapter	<input checked="" type="checkbox"/> Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Virtual Disk Image	cloudera-quickstart-vm-5.12.0-0-virtualbox-disk1.vmdk
Base Folder	C:\Users\Admin\VirtualBox VMs
Primary Group	/

Machine Base Folder: C:\Users\Admin\VirtualBox VMs

MAC Address Policy: Include only NAT network adapter MAC addresses

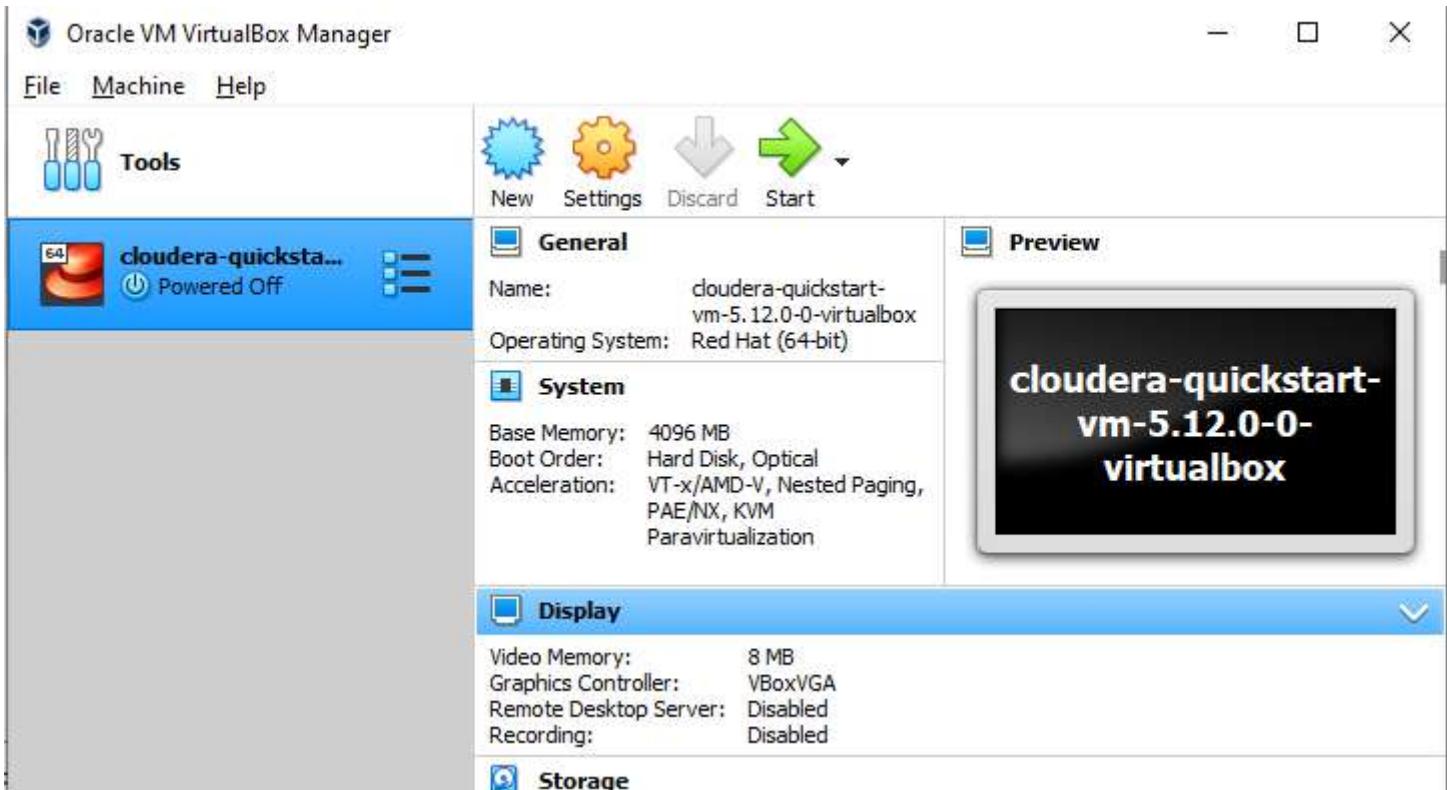
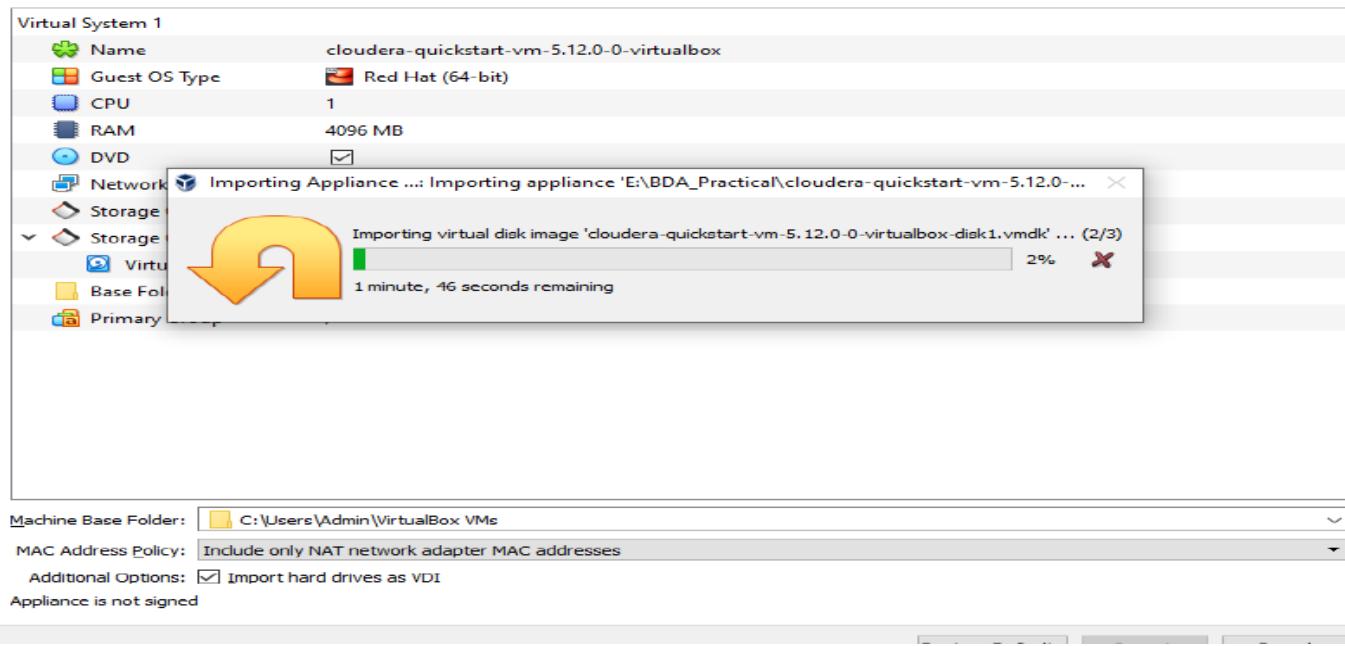
Additional Options: Import hard drives as VDI

Appliance is not signed

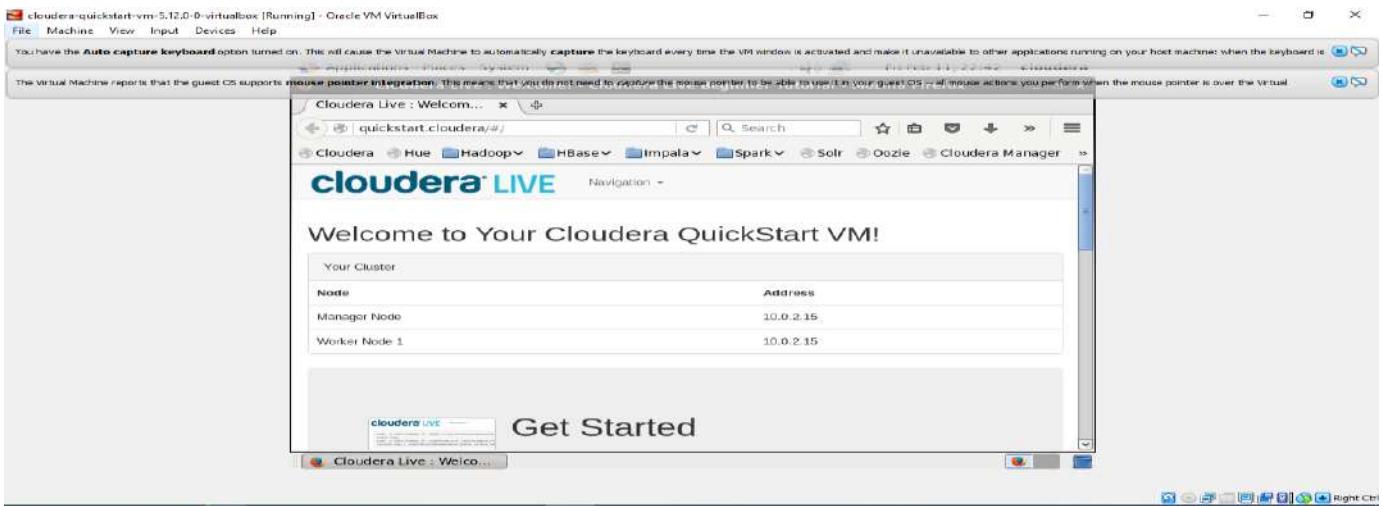
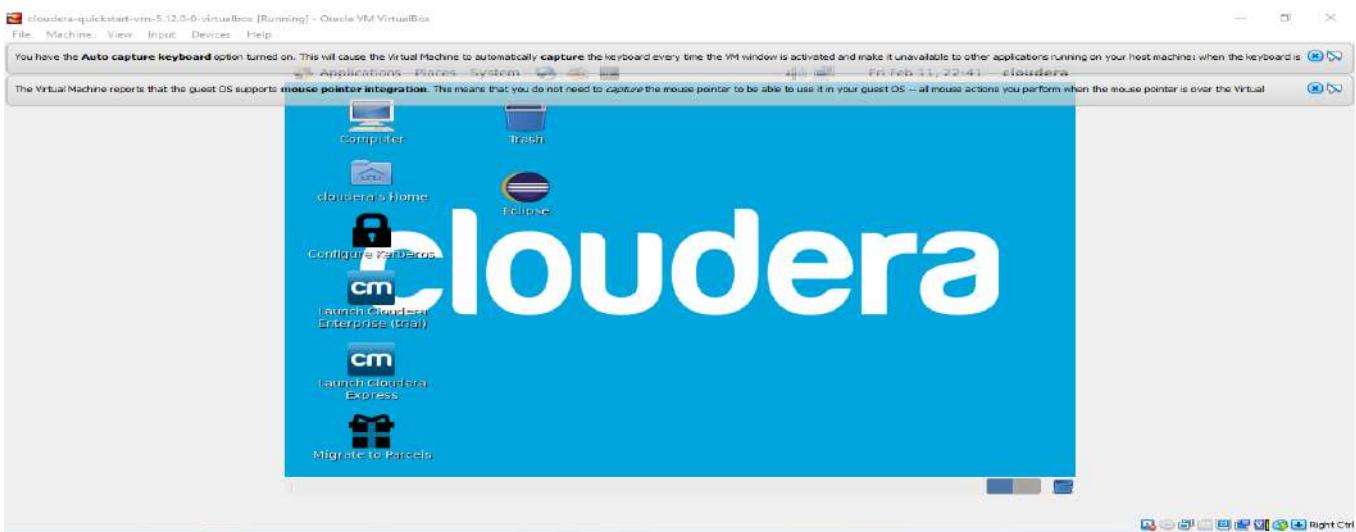
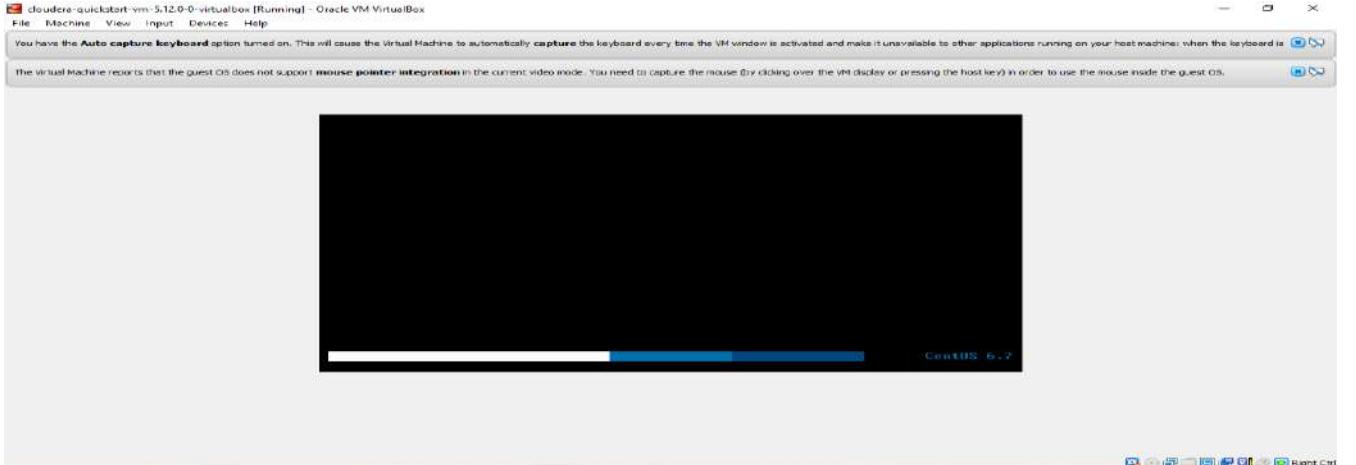
[← Import Virtual Appliance](#)

Appliance settings

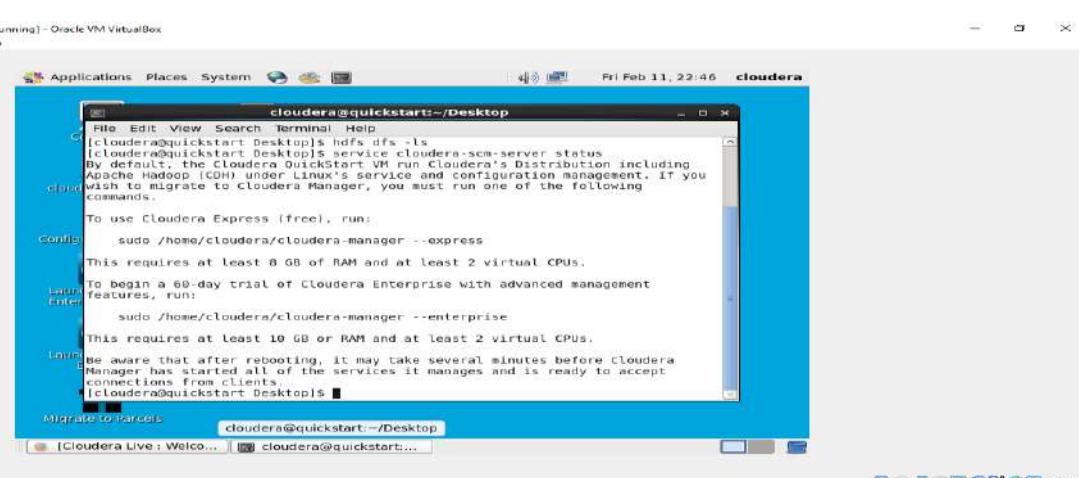
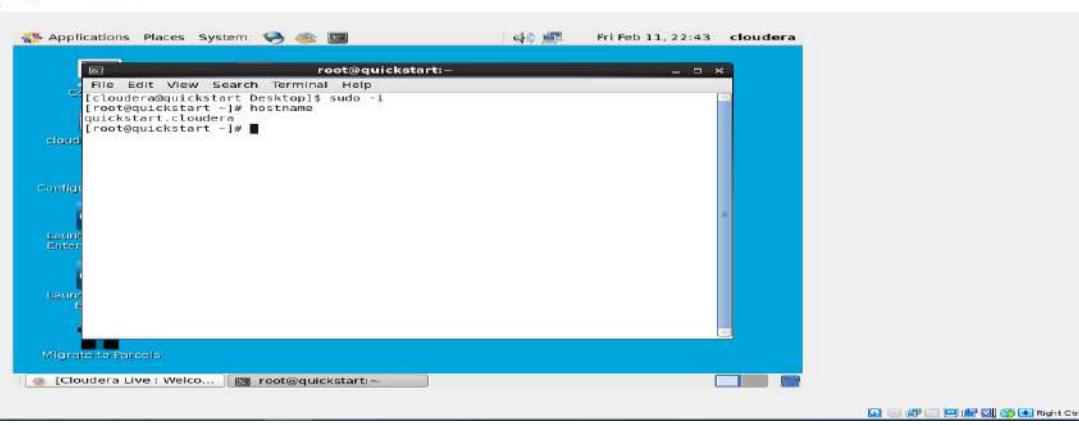
These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.



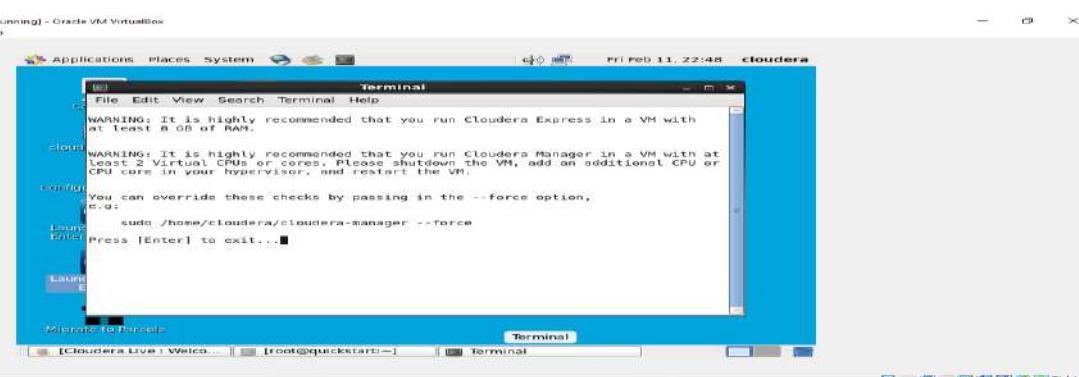
- The next step will be going ahead and starting the machine by clicking the 'Start' symbol on top.
- Once your machine comes on, it will look like this:



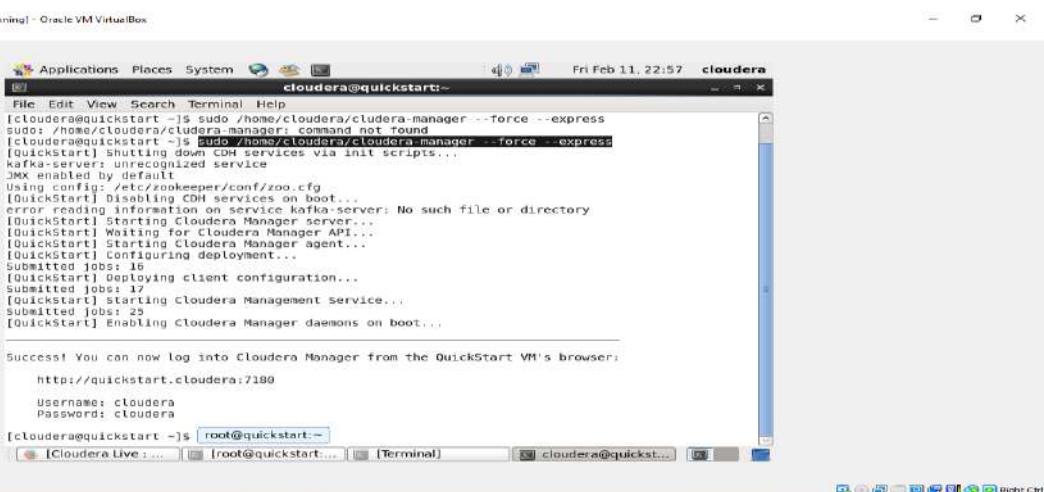
- Next, we have to follow a few steps to gain admin console access. You need to click on the terminal present on top of the desktop screen, and type in the following:
 1. **hostname** # This shows the hostname which will be quickstart.cloudera
 2. **hdfs dfs -ls /** # Checks if you have access and if your cluster is working. It displays what exists on your HDFS location by default
 3. **service cloudera-scm-server status** # Tells what command you have to type to use cloudera express free
 4. **su** - #Login as root
 5. **service cloudera-scm-server status** # The password for root is cloudera
- Once you see that your HDFS access is working fine, you can close the terminal. Then, you have to click on the following icon that says 'Launch Cloudera Express'.



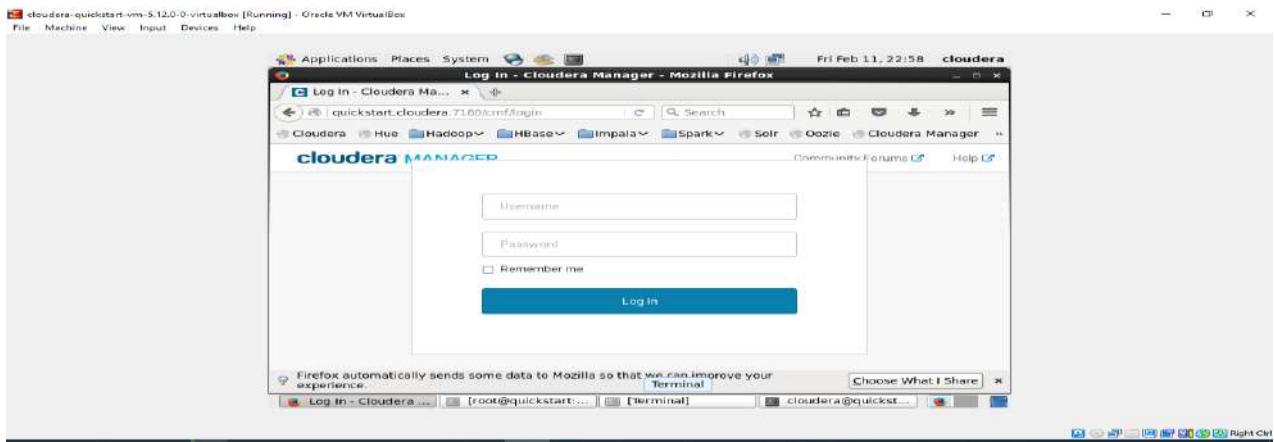
- You are required to copy the command, and run it on a separate terminal. Hence, open a new terminal, and use the below command to close the Cloudera based services. It will restart the services, after which you can access your admin console.



sudo /home/cloudera/cloudera-manager --force --express



- Now that our deployment has been configured, client configurations have also been deployed. Additionally, it has restarted the Cloudera Management Service, which gives access to the Cloudera QuickStart admin console with the help of a username and password.
- Go on and open up the browser and change the port number to 7180.
- You can log in to the Cloudera Manager by providing your username and password.



- You can go ahead and restart the services now. It will ensure that the cluster becomes accessible either by Hue as a web interface or Cloudera QuickStart Terminal, where you can write your commands.

Aim: To implement Various Hadoop HDFS Commands

What is Hadoop?

Apache Hadoop is an open source software framework used to develop data processing applications which are executed in a distributed computing environment.

Applications built using HADOOP are run on large data sets distributed across clusters of commodity computers. Commodity computers are cheap and widely available. These are mainly useful for achieving greater computational power at low cost.

Similar to data residing in a local file system of a personal computer system, in Hadoop, data resides in a distributed file system which is called as a Hadoop Distributed File system. The processing model is based on 'Data Locality' concept wherein computational logic is sent to cluster nodes (server) containing data. This computational logic is nothing, but a compiled version of a program written in a high-level language such as Java. Such a program, processes data stored in Hadoop HDFS.

Hadoop Distributed File System (HDFS) - Data Storage and Management

This is the most important component of the Hadoop ecosystem. HDFS is Hadoop's primary storage system. Hadoop Distributed File System (HDFS) is a Java-based file system that provides reliable, fault tolerance and accessible data storage for the big data. HDFS is a distributed file system that runs on conventional hardware. HDFS is already configured with the default settings for many installations. Typically, a large cluster configuration is required. Hadoop interacts directly with HDFS using commands. When comes to HDFS, there are also two components can be identified, which are known as Name Node and Data Node.

Name Node

It is also known as Master node. Here, it does not store actual data or datasets. Name Node stores the Meta data, for an example, the number of calls transform from a tower, their position, where the end users are getting the call, the Data node data and other details. Basically, this contains files and directories. The tasks of Name node can be recognized as follows.

- Managing file system namespace
- Controlling the access of clients to files
- Executing file system through naming, opening, closing files and directories

Data Node

Data node is called as Slave. Data node is responsible for the effective storage of data in HDFS. The data node completes read and write operations on customer request. They also send signals, known as heartbeats, to the name node. These heartbeats show the status of the data node. Replica Block of Data node consists of two files in the file system. The first file is for data and the second for registry metadata. HDFS metadata contains a data control. At startup, each Data node is connected to the appropriate Name node and grasp. The ID of the Data Node namespace and the software version are controlled by the handshake. If a discrepancy is detected, Data Node is automatically disabled. When comes to tasks of Data node, those can be detailed as follows.

- This is consisting of operations like block replica creation, deletion, and replication according to the instruction of Name node
- Managing data storage of the system

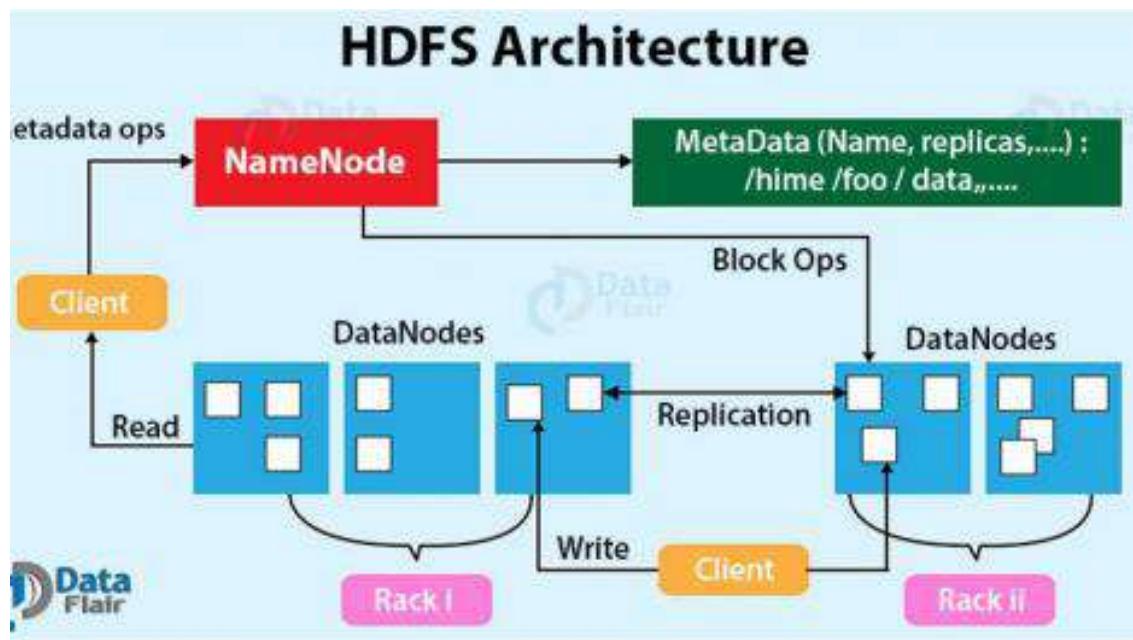
Processing and Computation – Hadoop MapReduce

When comes to Hadoop MapReduce, that is the main component of the Hadoop, that provides data processing. MapReduce is can be identified as an easy-to-write application framework that processes the large amount of structured and unstructured data stored in the Hadoop distributed file system.

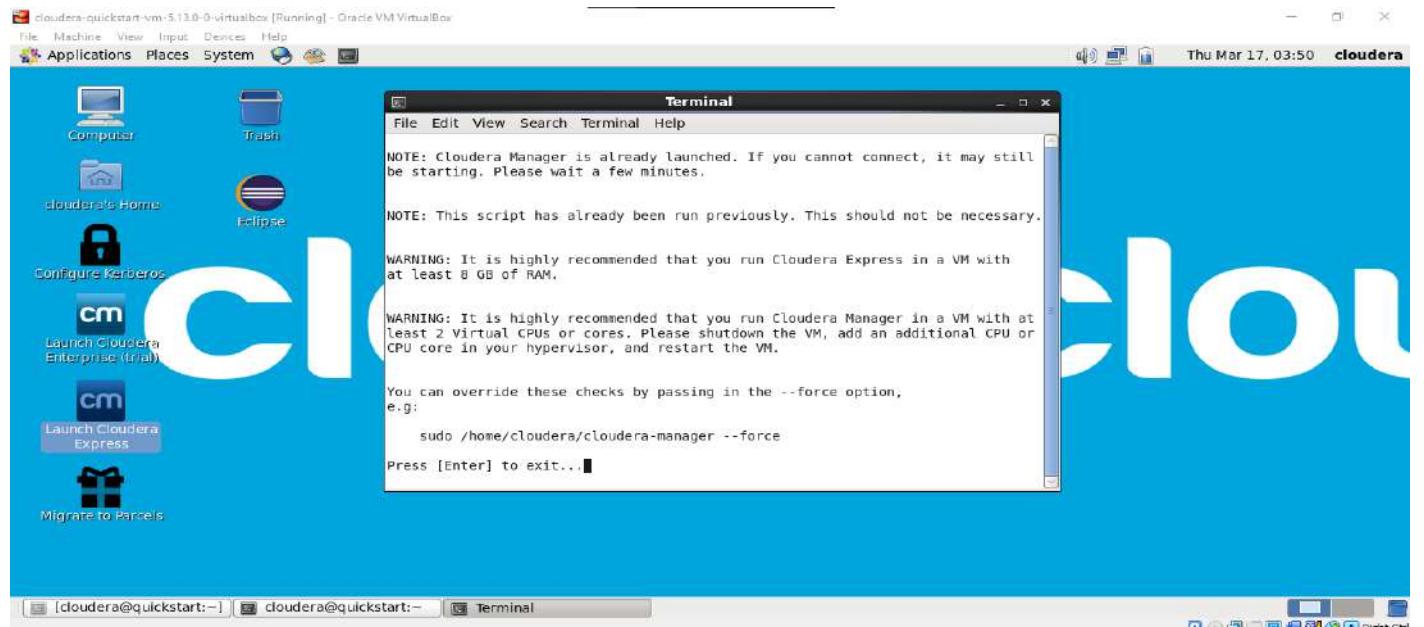
MapReduce programs are parallel, so they are very useful for large-scale data analysis using multiple clusters. Therefore, this parallelism increases the speed and reliability of the cluster. In MapReduce, there are two functions, Map function and Reduce function.

Two functions can be identified, map function and reduce function.

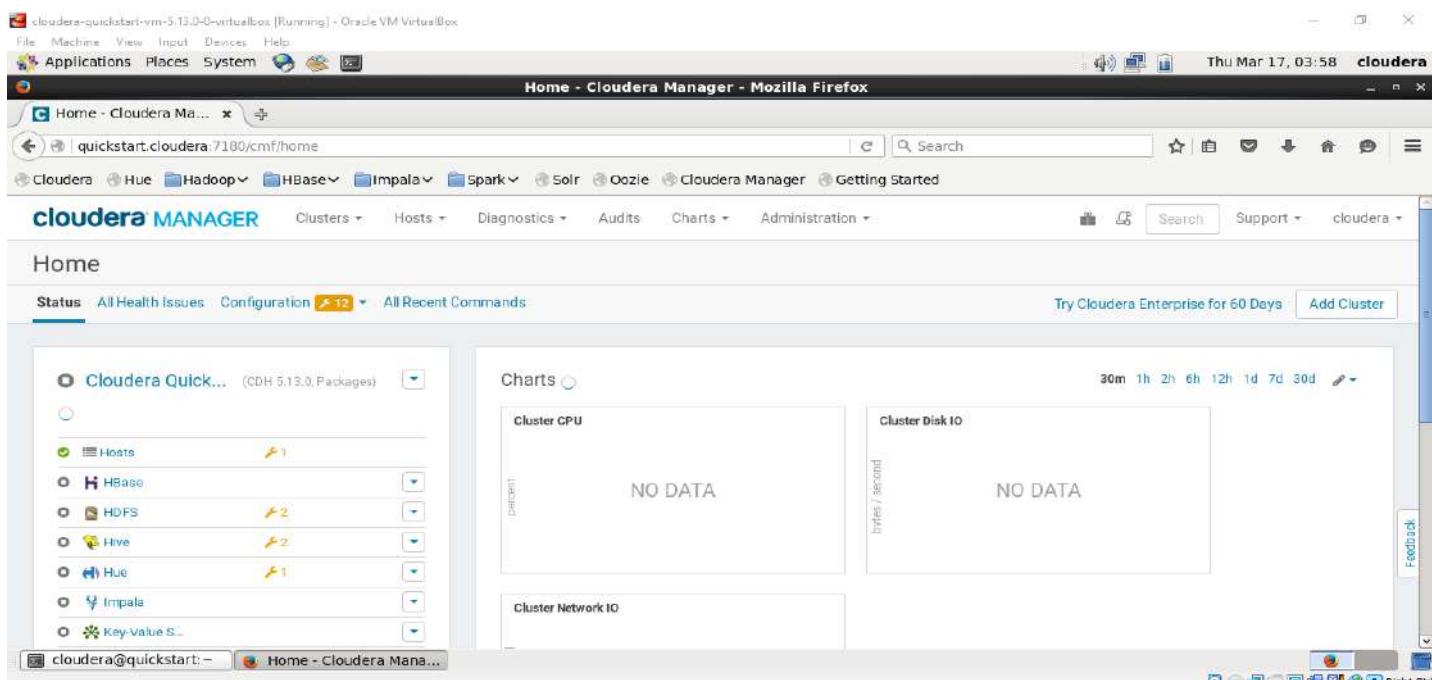
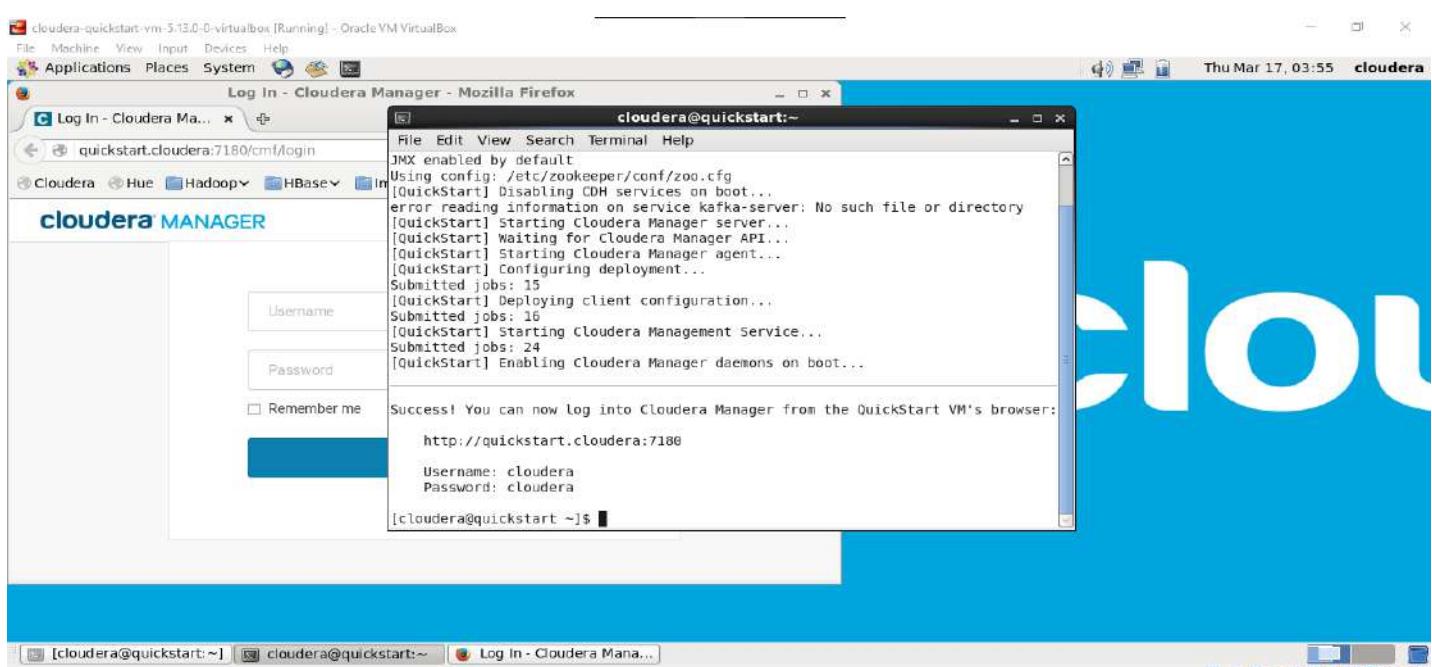
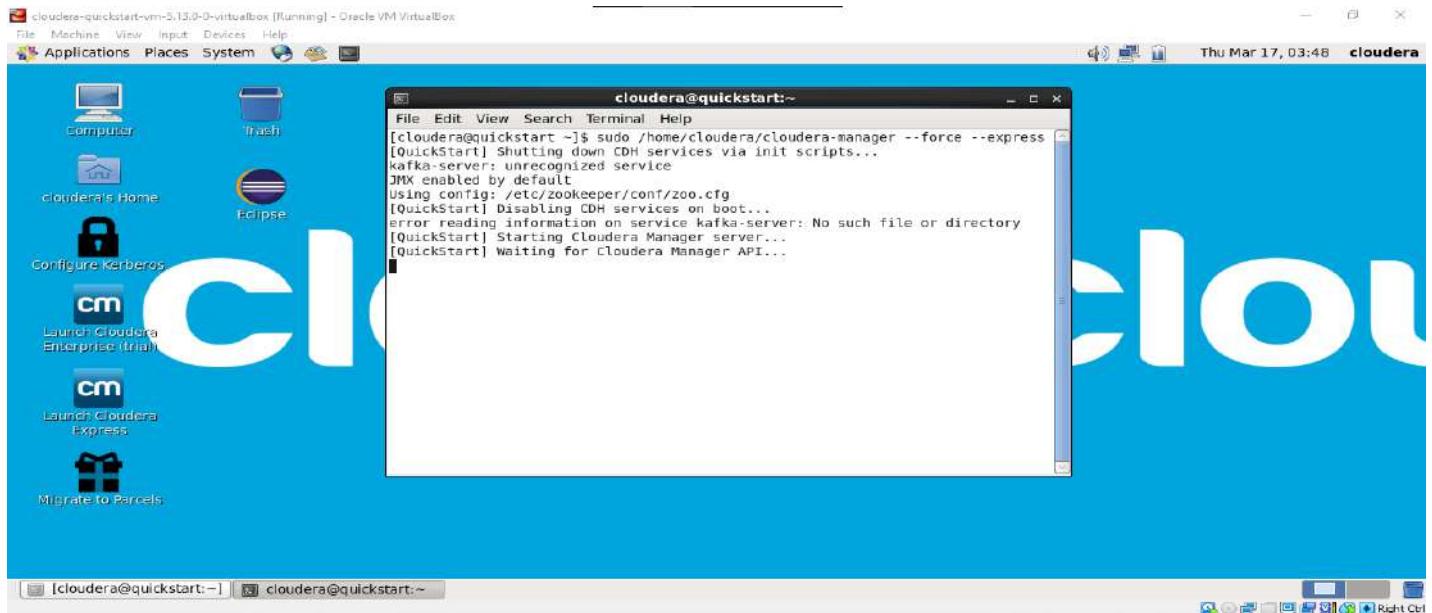
- The map function retrieves a data set and converts it to another data set. Each element is divided into processing (key / value pairs).
- The Reduce function accepts the Map output as an input and integrates these data nodes based on the key and changes the key value accordingly.



Copy the command from the Launch Cloudera Express



Type **sudo /home/cloudera/cloudera-manager --force --express** in terminal



Click on down arrow beside HDFS and select start

The screenshot shows the Cloudera Manager Home page. On the left, there is a sidebar with various service icons: Hosts, HBase, HDFS, Hive, Hue, Impala, Key-Value S., Cozie, Solr, Spark, Sqoop 1 Cli..., Sqoop 2, YARN (MR2...), and Zookeeper. A dropdown menu titled 'Hive Actions' is open, showing options: Start, Stop, Restart, Instances, Configuration, and Add Role Instances. The main area contains three charts: 'Cluster CPU' (bytes/second), 'Cluster Disk IO' (bytes/second), and 'Cluster Network IO' (bytes/second). All three charts show 'NO DATA'. At the top right, there is a date and time indicator: 'Thu Mar 17, 03:58' and a 'cloudera' user icon. The bottom of the screen shows a terminal window with the command 'cloudera@quickstart:~\$'.

The screenshot shows a 'Start Command' dialog box from Cloudera Manager. The title bar says 'Start Command'. Inside, it shows a status message: 'Status: Finished' (green circle), 'Context: Hive' (blue square), 'Mar 17, 3:59:07 AM', and '37.45s'. Below this, it says 'Successfully started service.' and 'Completed 1 of 1 step(s.)'. There are three radio button options: 'Show All Steps' (selected), 'Show Only Failed Steps', and 'Show Only Running Steps'. Under 'Show All Steps', there is a log entry: 'Starting 2 roles on service' with a green checkmark, followed by 'Successfully started 2 roles on service.' At the bottom right of the dialog is a 'Close' button. The background shows the same Cloudera Manager interface as the previous screenshot, with the terminal window at the bottom showing 'cloudera@quickstart:~\$'.

Hadoop HDFS Commands

First go to the **cd Desktop/**

1. `hadoop version` is used to check version of Hadoop system

```
[cloudera@quickstart Desktop]$ hadoop version
Hadoop 2.6.0-cdh5.13.0
subversion http://github.com/cloudera/hadoop -r 42e8800b182e55321bd5f3605264da4a
cloudera@quickstart:~$
```

2. `hdfs dfs -ls /` lists all the files and folders present in HDFS location. It lists the contents of the directory specified by path, names, permissions, owner, size and modification date for each entry. `hdfs dfs` is the command specific to HDFS.

```
[cloudera@quickstart Desktop]$ cd Desktop/
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 6 items
drwxrwxrwx  - hdfs  supergroup          0 2017-18-23 09:15 /benchmarks
drwxr-xr-x  - hbase supergroup          0 2022-03-17 03:00 /hbase
drwxr-xr-x  - solr   supergroup          0 2017-18-23 09:18 /solr
drwxrwxrwt - hdfs  supergroup          0 2022-03-17 06:49 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:29 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-18-23 09:17 /var
```

3. If getting any error on permission access use `export HADOOP_USER_NAME=hdfs` after this command run previous command again and use `hdfs dfs -ls /` to check if new directory "rjlocal" is created.

```
[cloudera@quickstart ~]$ export HADOOP_USER_NAME=hdfs
[cloudera@quickstart ~]$ hdfs dfs -ls /
ls: failed to connect to quickstart.cloudera:8020 - Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
```

Restart then execute the `hdfs dfs -ls /`

4. If getting error on safe mode then run **hadoop dfadmin -safemode leave**

```
[cloudera@quickstart:~]$ hadoop dfadmin -safemode leave
[cloudera@quickstart ~]$
```

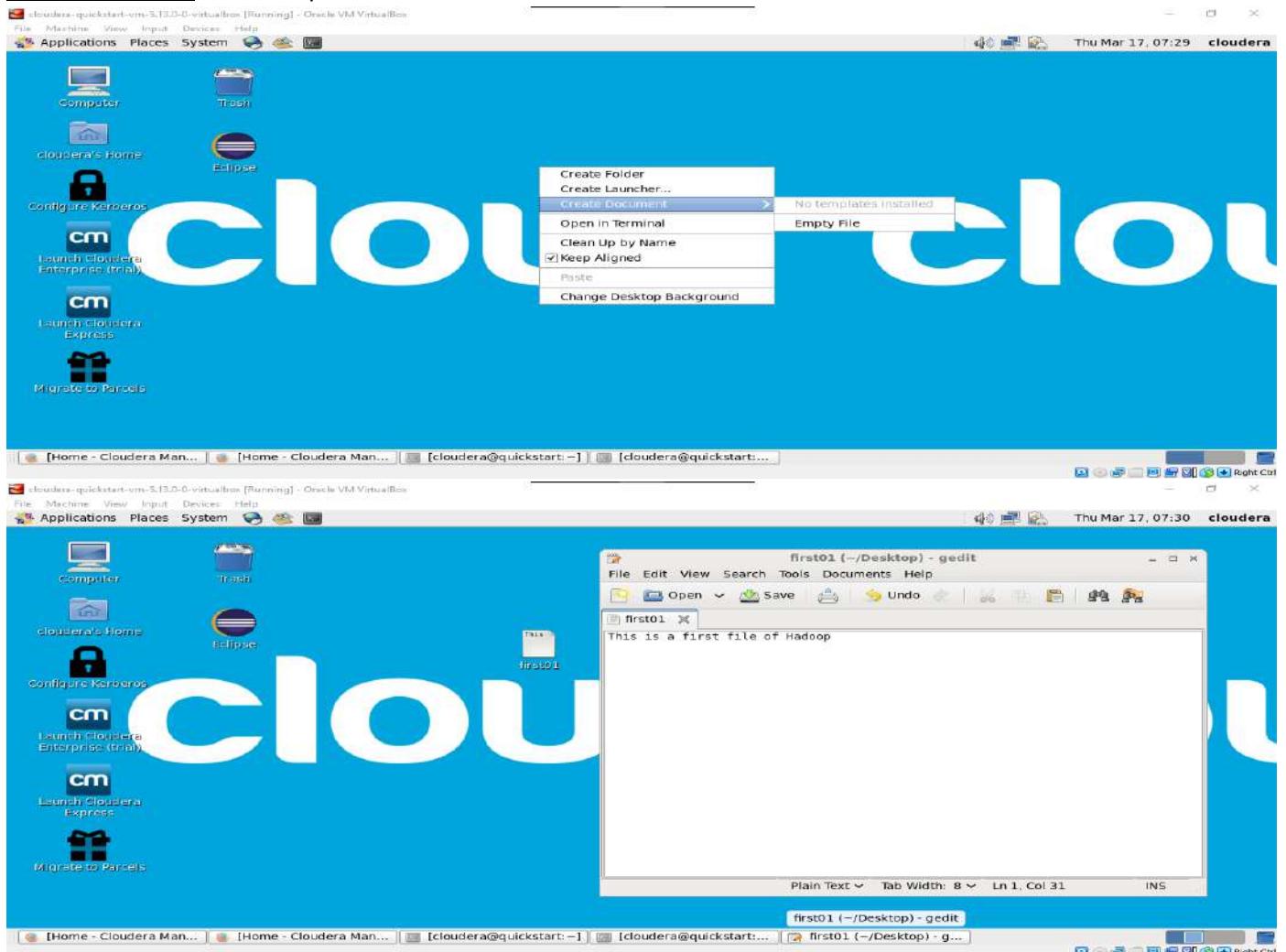
5. **hadoop fs -ls** / displays a list of content of a directory specified in the path provided by the user.

```
[cloudera@quickstart ~]$ cd Desktop/
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 6 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase  supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x  - solr   supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-17 06:48 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:28 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hadoop fs -ls /
Found 6 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase  supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x  - solr   supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-17 06:48 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:28 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$
```

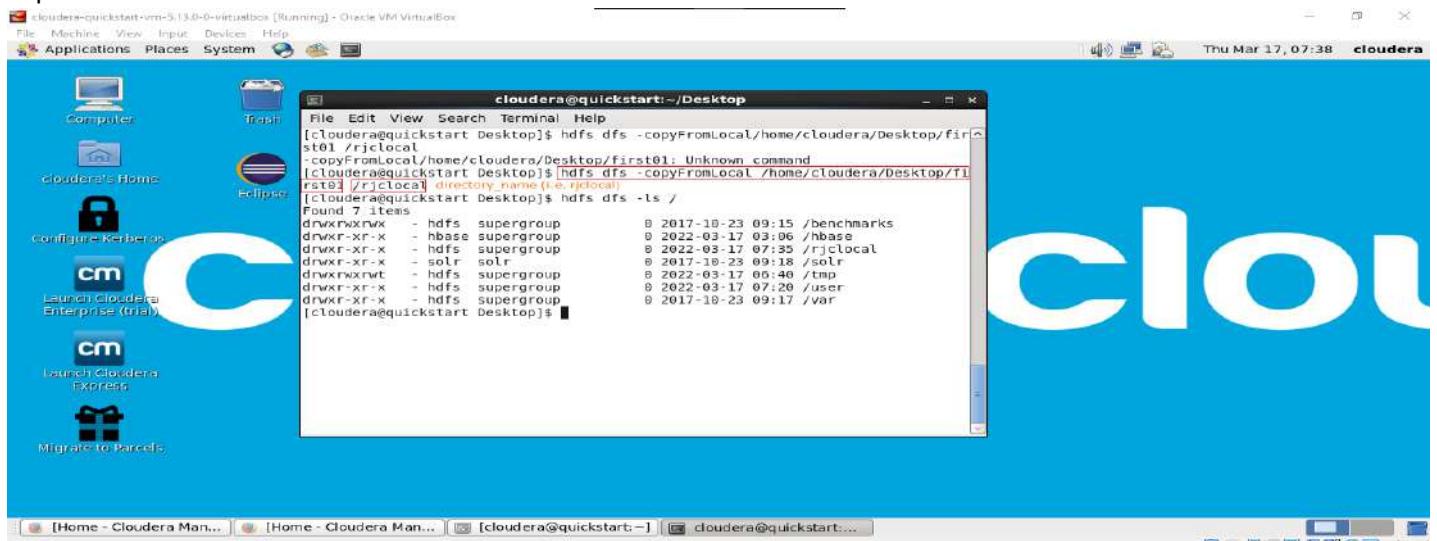
6. **hdfs dfs -mkdir /rjlocal**

```
[cloudera@quickstart ~]$ hadoop fs -ls /
Found 6 items
drwxrwxrwx  - hbase  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase  supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x  - solr   supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-17 06:48 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:28 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hadoop fs -mkdir /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase  supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:22 /rjlocal
drwxr-xr-x  - solr   supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-17 06:48 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:28 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$
```

7. Create a new file on desktop



8. Use `hdfs dfs -copyFromLocal /home/cloudera/Desktop/<file_name> /<directory_name>`- this command copies <file_name> file to HDFS <directory_name> directory and use `hdfs dfs -ls /rjclocal` to check is file is successfully copied



```

cloudera@quickstart:~$ hdfs dfs -copyFromLocal /home/cloudera/Desktop/first01 /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal /home/cloudera/Desktop/first01; Unknown command
[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal /home/cloudera/Desktop/first01 /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:35 /rjlocal
drwxr-xr-x - solr supergroup 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /rjlocal/first01
[cloudera@quickstart Desktop]$

```

9. `hdfs dfs -cat </directory_name>/<file_name>` reads content of file

```

On Hadoop file system file content copy from the local
cloudera@quickstart:~$ hdfs dfs -cat /rjlocal/first01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$

```

- 10. First we create a new directory in HDFS “`newdir`” `hdfs dfs -mkdir </dir_name>` and then use `hdfs dfs -cp </dir1_name>/<dir_file_name> </dir2_name>` then check if file is copied successfully by using `hdfs dfs -ls </dir2_name>` `cp` command is used to copy file from one directory to another within HDFS**

Make directory

```

cloudera@quickstart:~$ hdfs dfs -cat /rjlocal/first01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /newdir
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:53 /newdir
drwxr-xr-x - solr supergroup 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$

```

Copy the file from one directory to other directory

```
cloudera@quickstart:~$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx - hbase supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup           0 2022-03-17 07:53 /newdir
drwxr-xr-x - hdfs supergroup           0 2022-03-17 07:35 /rjlocal
drwxr-xr-x - solr  supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup           0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup           0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup           0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -cp /rjlocal/first01 /newdir
[cloudera@quickstart Desktop]$
```

Check the newdir file is copy or not using "hdfs dfs -ls </dir_name>"

```
cloudera@quickstart:~$ hdfs dfs -ls /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup          31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$
```

11. hdfs dfs -mv /<dir1_name>/<file1_name> /output_abc moves file <file1_name> from source to destination within HDFS

```
cloudera@quickstart:~$ hdfs dfs -ls /newdir
Found 2 items
-rw-r--r-- 1 hdfs supergroup          31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$ hdfs dfs -mv /rjlocal/first01 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 9 items
drwxrwxrwx - hdfs supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup          0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup           0 2022-03-17 07:53 /newdir
drwxr-xr-x - solr  supergroup          0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup           0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup           0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup           0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 2 items
-rw-r--r-- 1 hdfs supergroup          31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$
```

After moving file source folder to destination folder

```

cloudera@quickstart:~/Desktop$ hdfs dfs -cp /rjlocal/first01 /newdir
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$ hdfs dfs -mv /rjlocal/first01 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 9 items
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:28 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -cp /rjlocal/first01 /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$ hdfs dfs -mv /rjlocal/first01 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 9 items
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
drwxr-xr-x - hdfs supergroup 0 2022-03-17 12:41 /rjlocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:28 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /output_01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
[cloudera@quickstart Desktop]$ 

```

Use **hdfs dfs -cat </file1_name>** command to read moves file

```

cloudera@quickstart:~/Desktop$ hdfs dfs -cp /rjlocal/first01 /newdir
[cloudera@quickstart Desktop]$ hdfs dfs -mv /rjlocal/first01 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$ hdfs dfs -mv /rjlocal/first01 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 9 items
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
drwxr-xr-x - hdfs supergroup 0 2022-03-17 12:41 /rjlocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:28 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /output_01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -cat /output_01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$ 

```

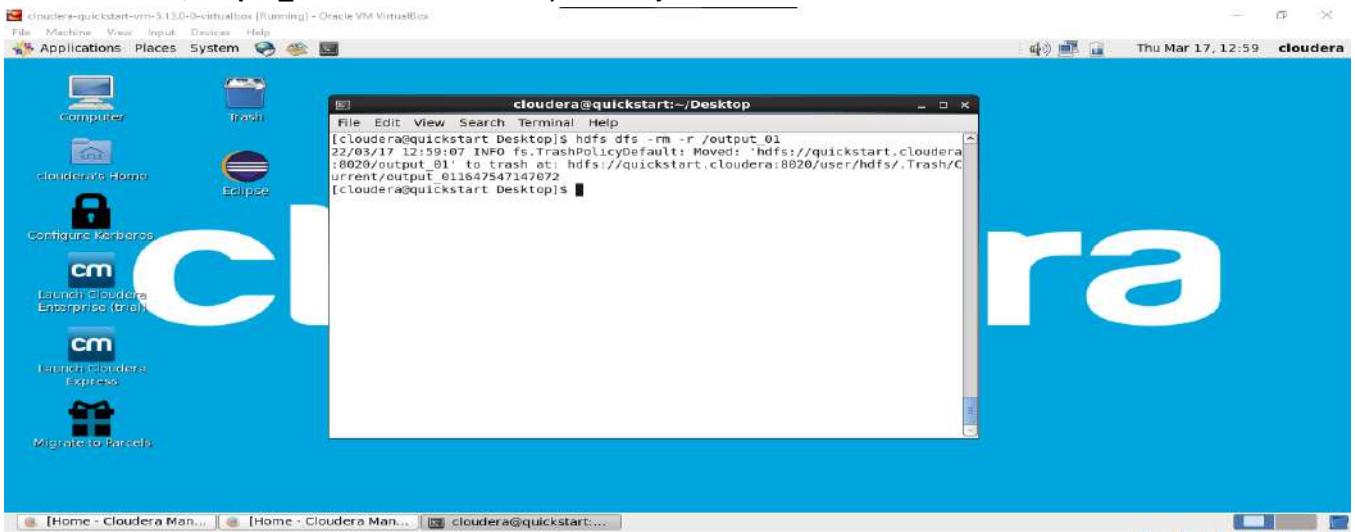
12. **hdfs dfs -rm </dir_name>/<file_name>** deletes objects and directories full of objects

```

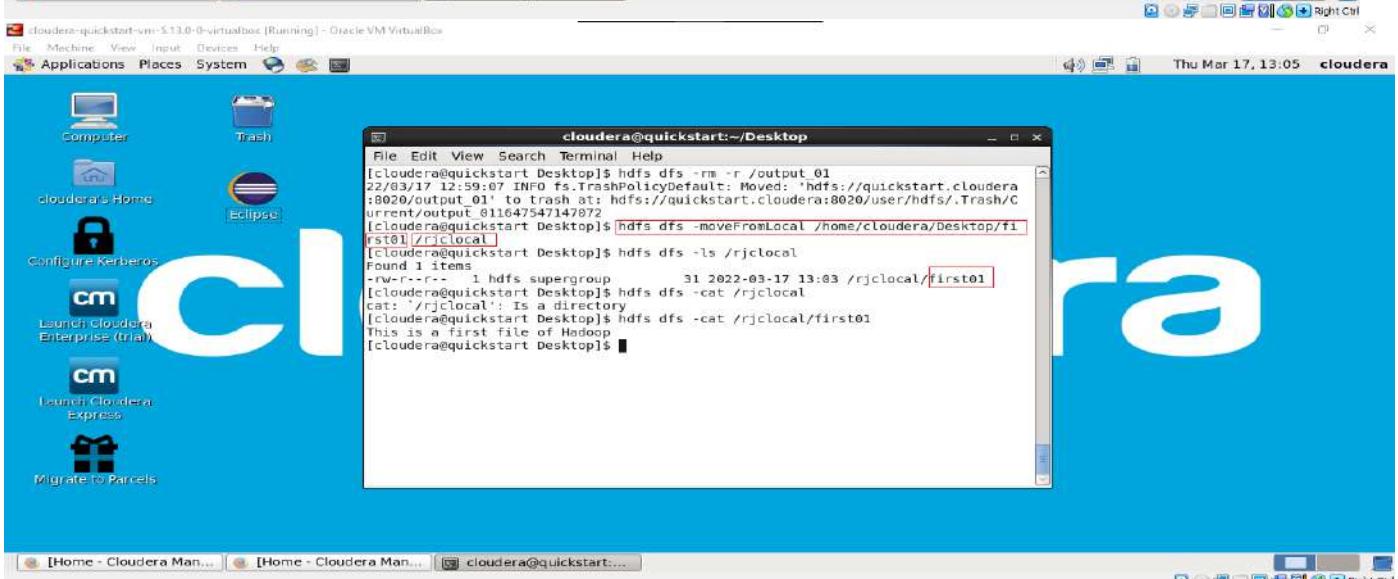
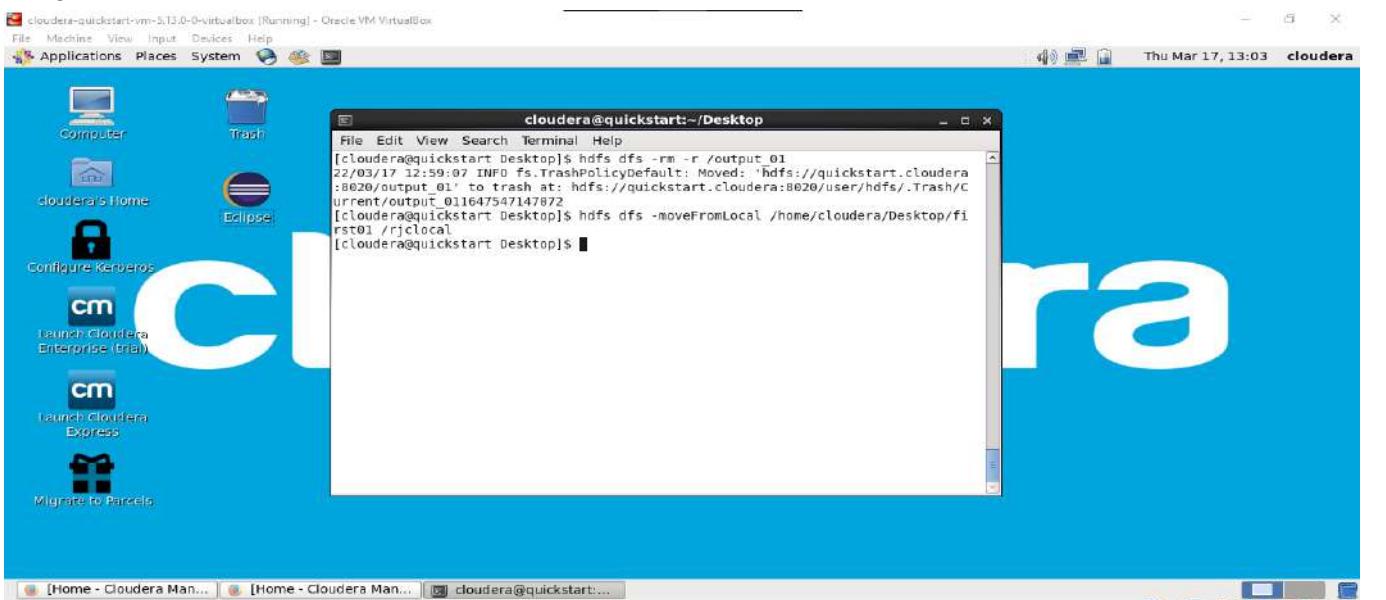
cloudera@quickstart:~/Desktop$ hdfs dfs -cp /rjlocal/first01 /newdir
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 9 items
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
drwxr-xr-x - hdfs supergroup 0 2022-03-17 12:41 /rjlocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:28 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /output_01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:35 /output_01
[cloudera@quickstart Desktop]$ hdfs dfs -cat /output_01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$ hdfs dfs -rm /rjlocal/first01
rm: '/rjlocal/first01': No such file or directory
[cloudera@quickstart Desktop]$ 

```

13. `hdfs dfs -rm -r /output_abc` removes directory with objects



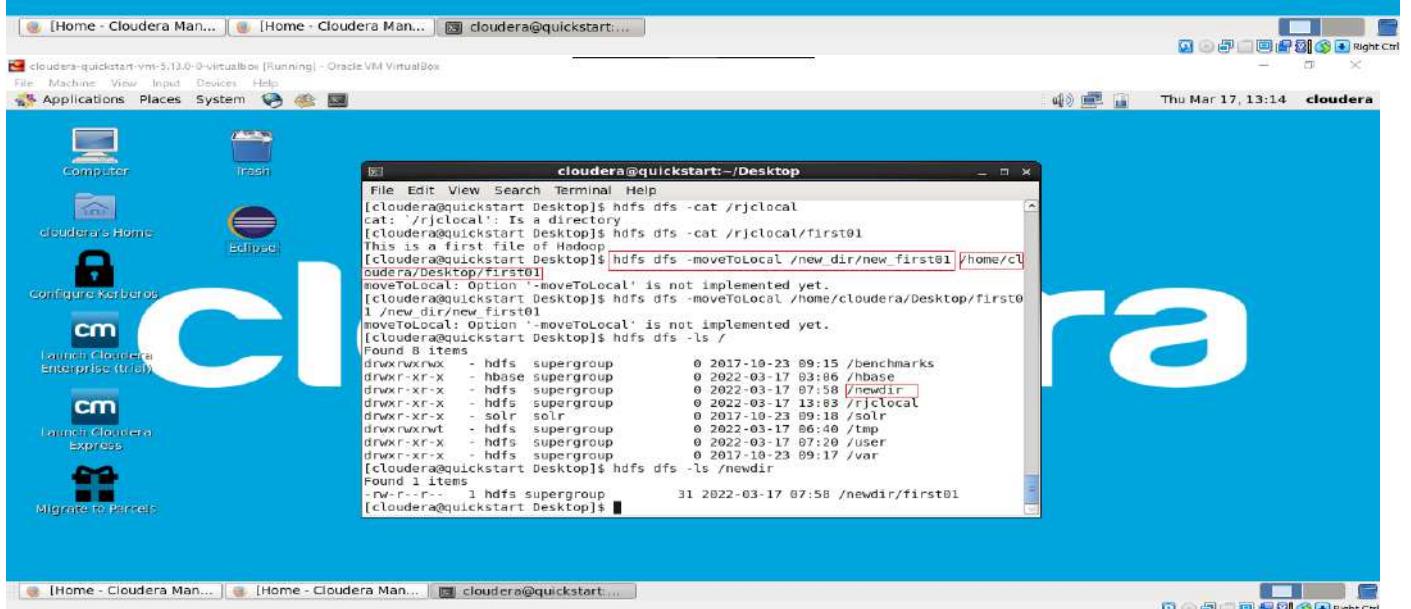
14. `hdfs dfs -moveFromLocal /home/cloudera/Desktop/<file_name> /<dir_name>` moves files from local file system to HDFS



15. `hdfs dfs -moveToLocal <new_dir_name>/<file_name>` /home/cloudera/Desktop/<file_name> moves files.

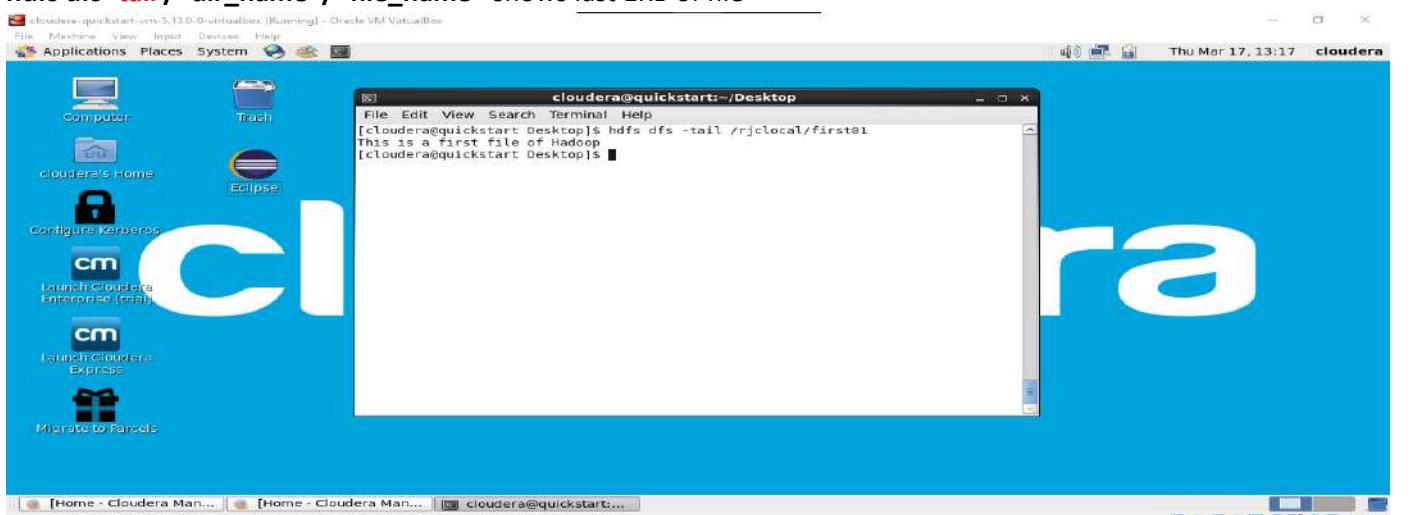


```
[cloudera@quickstart Desktop]$ hdfs dfs -rm -r /output_01
22/03/17 12:59:07 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera:8020/output_01' to trash at: hdfs://quickstart.cloudera:8020/user/hdfs/.Trash/current/output_011647547147872
[cloudera@quickstart Desktop]$ hdfs dfs -moveFromLocal /home/cloudera/Desktop/first01 /rjlocal
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart Desktop]$ hdfs dfs -cat /rjlocal
cat: /rjlocal/: Is a directory
[cloudera@quickstart Desktop]$ hdfs dfs -moveToLocal /new_dir/new_first01 /home/cloudera/Desktop/first01
moveToLocal: Option '-moveToLocal' is not implemented yet.
[cloudera@quickstart Desktop]$ hdfs dfs -moveToLocal /home/cloudera/Desktop/first01 /new_dir/new_first01
moveToLocal: Option '-moveToLocal' is not implemented yet.
[cloudera@quickstart Desktop]$
```



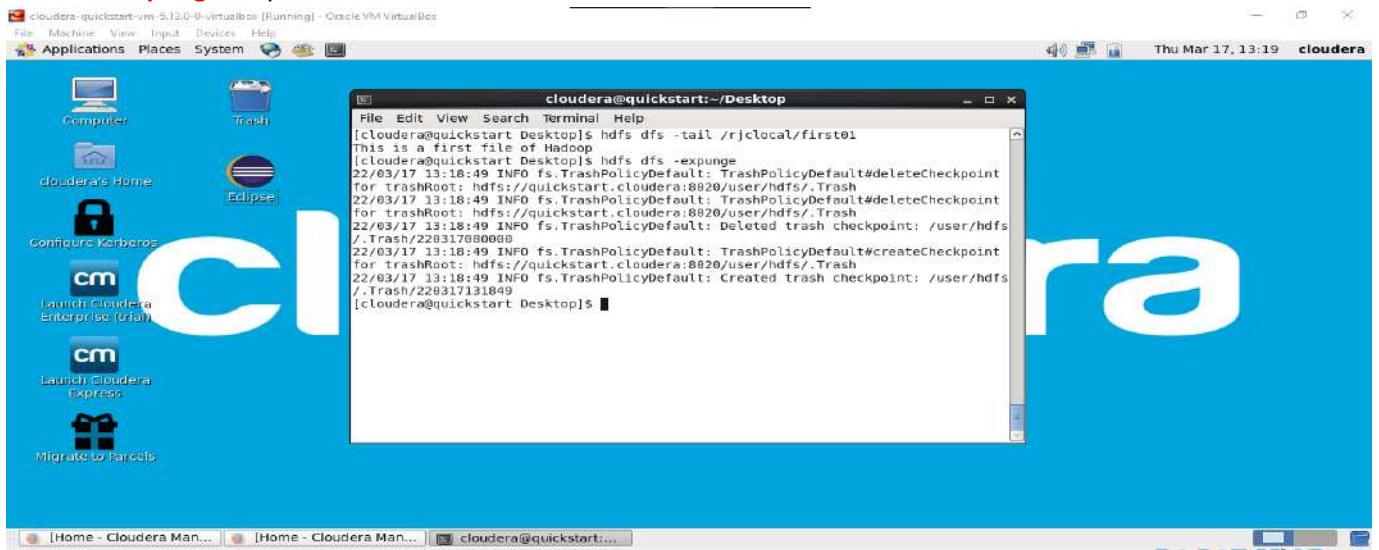
```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /rjlocal
cat: '/rjlocal': Is a directory
[cloudera@quickstart Desktop]$ hdfs dfs -cat /rjlocal/first01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$ hdfs dfs -moveToLocal /new_dir/new_first01 /home/cloudera/Desktop/first01
moveToLocal: Option '-moveToLocal' is not implemented yet.
[cloudera@quickstart Desktop]$ hdfs dfs -moveToLocal /home/cloudera/Desktop/first01 /new_dir/new_first01
moveToLocal: Option '-moveToLocal' is not implemented yet.
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-17 03:06 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
drwxr-xr-x - hdfs supergroup 0 2022-03-17 13:03 /rjlocal
drwxr-xr-x - solr supergroup 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-17 06:40 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$ hdfs dfs -ls /newdir
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 07:58 /newdir/first01
[cloudera@quickstart Desktop]$
```

16. `hdfs dfs -tail /<dir_name>/<file_name>` shows last 1KB of file



```
[cloudera@quickstart Desktop]$ hdfs dfs -tail /rjlocal/first01
This is a first file of Hadoop
[cloudera@quickstart Desktop]$
```

17. `hdfs dfs -expunge` empties the trash available in HDFS



18. First go to local host at `localhost:50070` click on Utilities and select browse the file system

The screenshot shows the 'Namenode Information' page in the Cloudera Manager interface. The 'Utilities' tab is selected. The 'Overview' section displays the following information:

Started:	Thu Mar 17 06:42:47 -0700 2022
Version:	2.6.0-cdh5.13.0, r42e8800b182e55321bd5f5605264da4adc8882be
Compiled:	Wed Oct 04 11:08:00 -0700 2017 by jenkins from Unknown
Cluster ID:	CID-a24185f9-a545-40fc-9553-84c3fdca489f
Block Pool ID:	BP-1067413441-127.0.0.1-1508775264580

The 'Summary' section is also visible.

Type `/` in test area and click go to see current directories in file system

The screenshot shows the 'Snapshot Summary' page in the Cloudera Manager interface. The 'Utilities' tab is selected. A context menu is open over the 'Utilities' tab, with the option 'Browse the file system' highlighted.

The 'Snapshot Summary' section displays the following information:

Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group
Snapshottable directories: 0						

The 'Snapshot Directory' table shows:

Snapshot ID	Snapshot Directory	Modification Time
Snapshotted directories: 0		

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	hdfs	supergroup	0 B	Mon Oct 23 09:15:43 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Thu Mar 17 03:06:27 -0700 2022	0	0 B	hbase
drwxr-xr-x	hdfs	supergroup	0 B	Thu Mar 17 07:58:45 -0700 2022	0	0 B	newdir
drwxr-xr-x	hdfs	supergroup	0 B	Thu Mar 17 13:03:05 -0700 2022	0	0 B	rjlocal
drwxr-xr-x	solr	solr	0 B	Mon Oct 23 09:18:01 -0700 2017	0	0 B	solr

hdfs dfs -setrep 4 <dir_name> sets replication of files in specifies directory 4 times

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ hdfs dfs -setrep 4 /newdir
Replication 4 set: /newdir/first01
[cloudera@quickstart Desktop]$
```

Check replication of file within specified directory

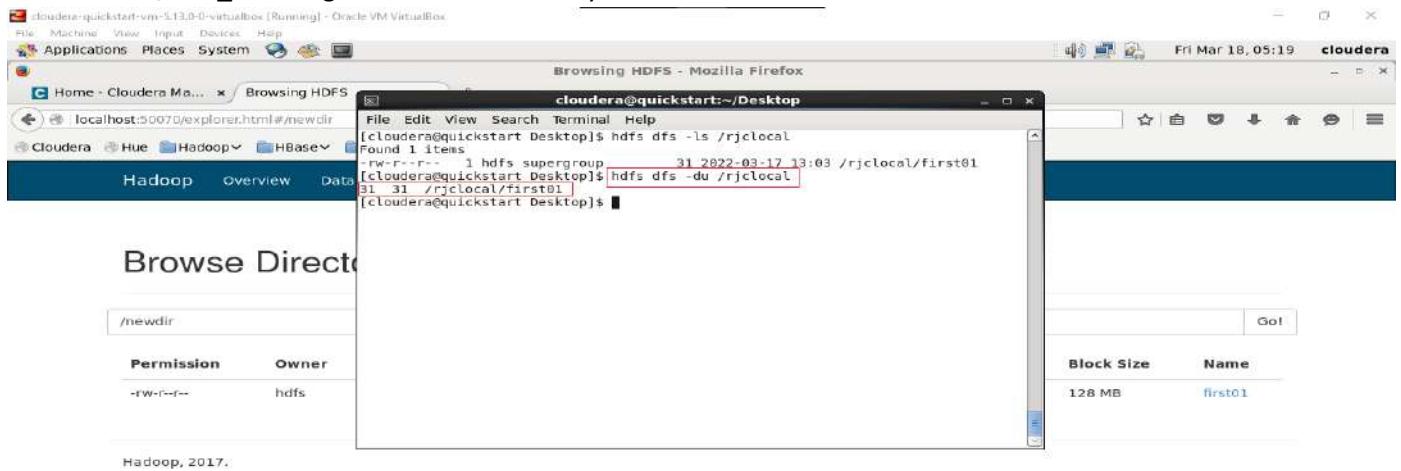
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hdfs	supergroup	31 B	Thu Mar 17 07:58:45 -0700 2022	4	128 MB	first01

Browse Directory

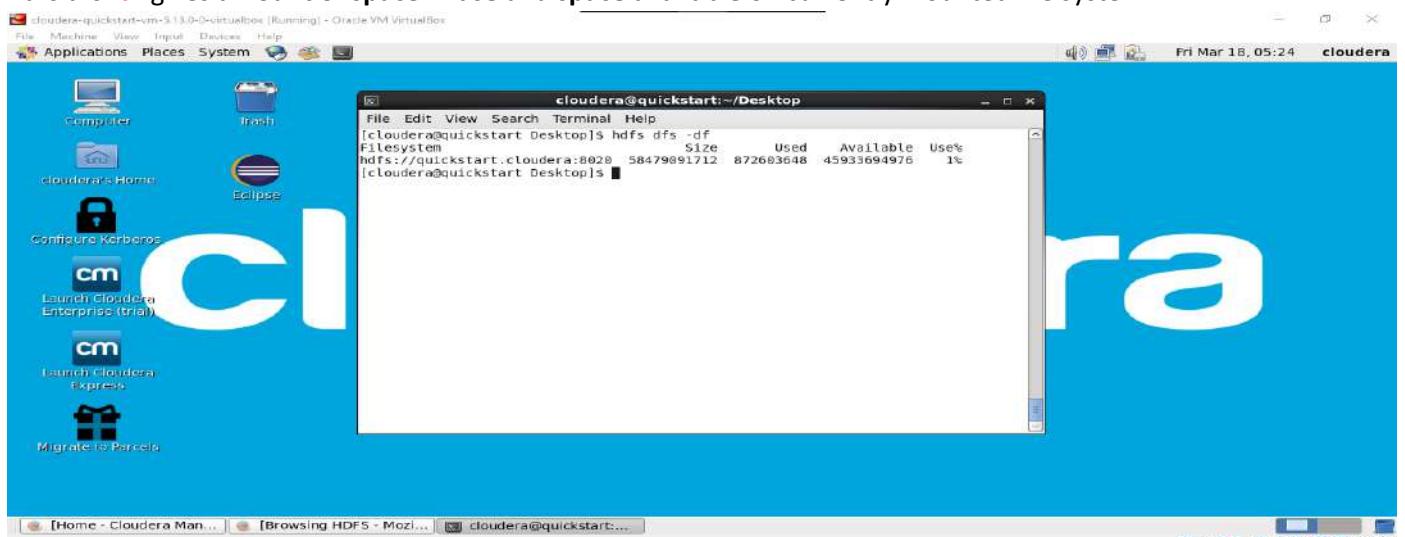
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hdfs	supergroup	31 B	Thu Mar 17 07:58:45 -0700 2022	4	128 MB	first01

Hadoop, 2017.

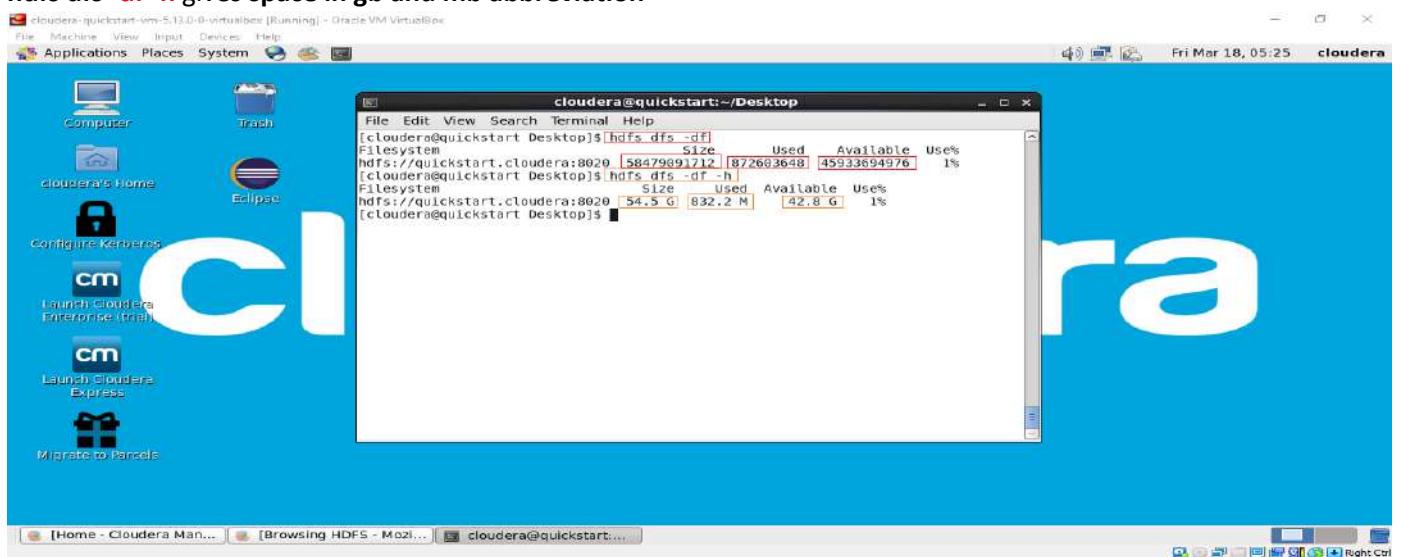
19. `hdfs dfs -du /<dir_name>` gives size of a directory in HDFS



20. `hdfs dfs -df` gives amount of space in use and space available on currently mounted file system



`hdfs dfs -df -h` gives space in gb and mb abbreviation



- 21. hdfs fsck <dir_name> checks health of HDFS and moves corrupted files to lost+found directory or It delete a corrupted file present in HDFS**

```
[cloudera@quickstart Desktop]$ hdfs fsck /rjlocal
Connecting to namenode via http://quickstart.cloudera:50070/fsck?ugi=hdfs&path=%2Frjlocal
FSCK started by hdfs (auth:SIMPLE) from /10.0.2.15 for path /rjlocal at Fri Mar 18 05:29:41 PDT 2022
.Status: HEALTHY
Total size: 31 B
Total dirs: 1
Total files: 1
Total symlinks: 0
Total blocks (validated): 1 (avg. block size 31 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Fri Mar 18 05:29:41 PDT 2022 in 14 milliseconds

The filesystem under path '/rjlocal' is HEALTHY
[cloudera@quickstart Desktop]$
```

hdfs fsck /rjlocal -files prints out the files being checked

```
[cloudera@quickstart Desktop]$ hdfs fsck /rjlocal -files
Connecting to namenode via http://quickstart.cloudera:50070/fsck?ugi=hdfs&path=%2Frjlocal
FSCK started by hdfs (auth:SIMPLE) from /10.0.2.15 for path /rjlocal at Fri Mar 18 05:32:45 PDT 2022
/rjlocal <dir>
/rjlocal/first01 31 bytes, 1 block(s): OK
.Status: HEALTHY
Total size: 31 B
Total dirs: 1
Total files: 1
Total symlinks: 0
Total blocks (validated): 1 (avg. block size 31 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Fri Mar 18 05:32:45 PDT 2022 in 2 milliseconds

The filesystem under path '/rjlocal' is HEALTHY
[cloudera@quickstart Desktop]$
```

- 22. hdfs dfs -touchz <dir_name>/<empty_file_name> creates an empty file**

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 1 items
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart Desktop]$ hdfs dfs -touchz /rjlocal/empFirst01
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-18 05:36 /rjlocal/empFirst01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart Desktop]$
```

23. **hdfs dfs -stat </dir_name>** prints statistics about the file or directory it shows recent date of modification

```
[cloudera@quickstart:~] hdfs dfs -stat /rjlocal
2022-03-18 12:36:11
[cloudera@quickstart:~]
```

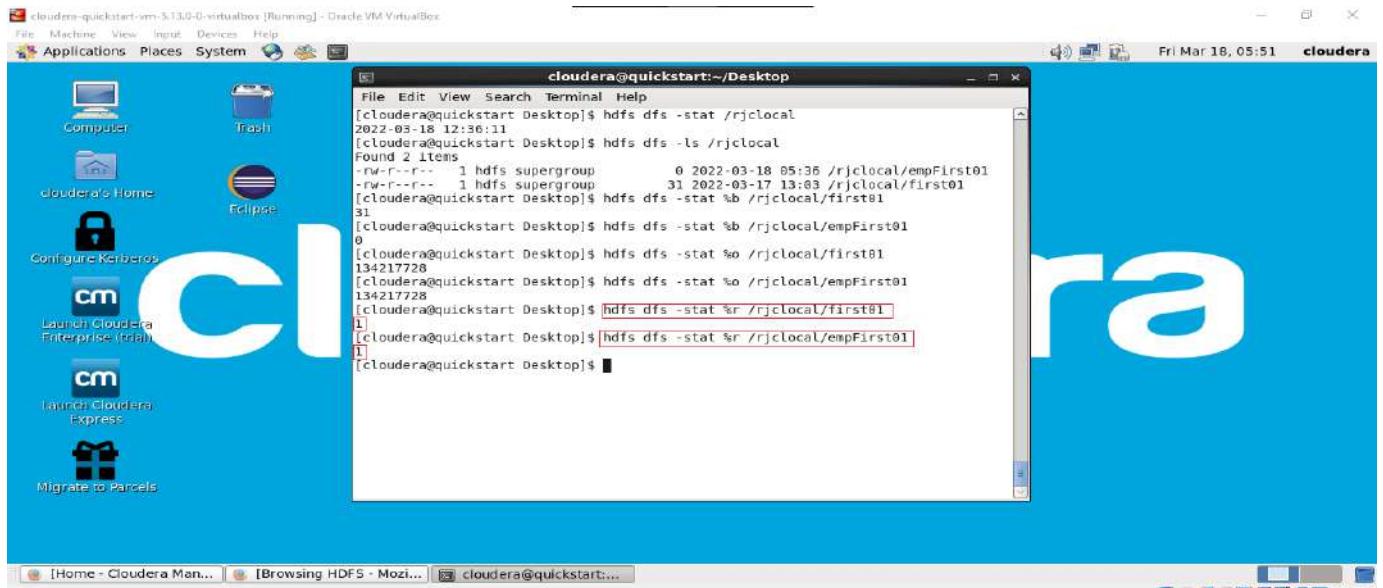
hdfs -stat %b </dir_name>/<file_name> gives file size in bytes

```
[cloudera@quickstart:~] hdfs dfs -stat /rjlocal
2022-03-18 12:36:11
[cloudera@quickstart:~] hdfs dfs -ls /rjlocal
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-18 05:36 /rjlocal/empFirst01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart:~] hdfs dfs -stat %b /rjlocal/first01
31
[cloudera@quickstart:~] hdfs dfs -stat %b /rjlocal/empFirst01
0
[cloudera@quickstart:~]
```

hdfs dfs -stat %o /<dir_name>/<file_name> gives block size of file

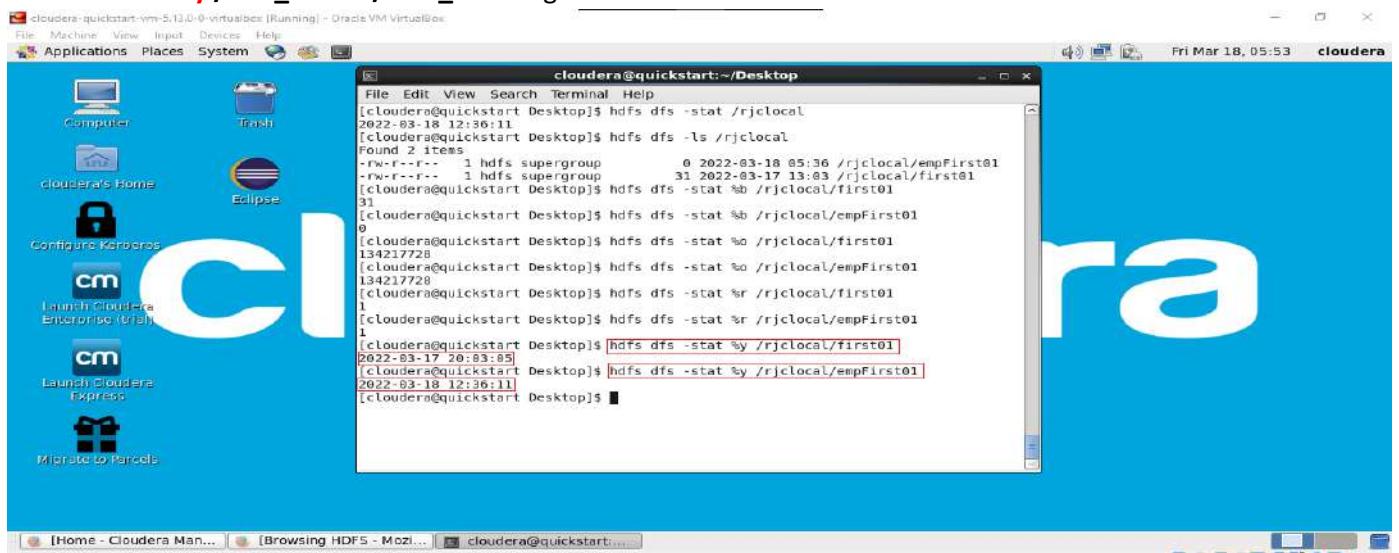
```
[cloudera@quickstart:~] hdfs dfs -stat /rjlocal
2022-03-18 12:36:11
[cloudera@quickstart:~] hdfs dfs -ls /rjlocal
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-18 05:36 /rjlocal/empFirst01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart:~] hdfs dfs -stat %o /rjlocal/first01
31
[cloudera@quickstart:~] hdfs dfs -stat %b /rjlocal/empFirst01
0
[cloudera@quickstart:~] hdfs dfs -stat %o /rjlocal/first01
134217728
[cloudera@quickstart:~] hdfs dfs -stat %o /rjlocal/empFirst01
134217728
[cloudera@quickstart:~]
```

hdfs dfs -stat %r /<dir_name>/<file_name> gives no. of replication of file



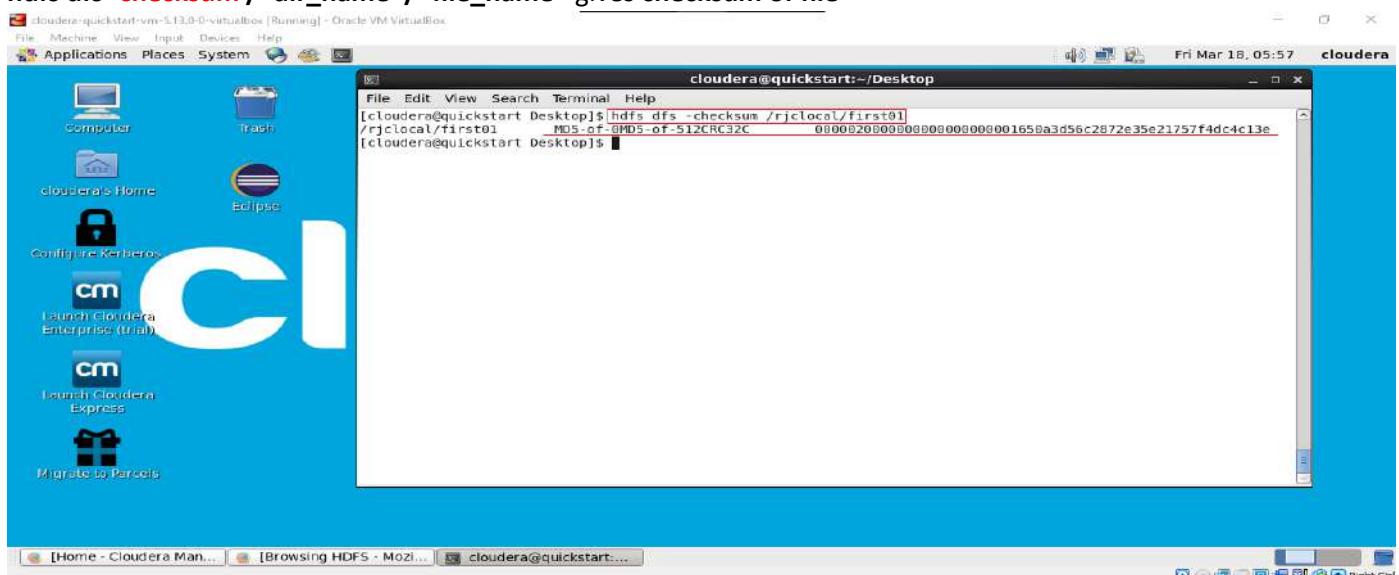
```
[cloudera@quickstart Desktop]$ hdfs dfs -stat /rjlocal
2022-03-18 12:36:11
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-18 05:36 /rjlocal/empFirst01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart Desktop]$ hdfs dfs -stat %b /rjlocal/first01
31
[cloudera@quickstart Desktop]$ hdfs dfs -stat %b /rjlocal/empFirst01
0
[cloudera@quickstart Desktop]$ hdfs dfs -stat %o /rjlocal/first01
134217728
[cloudera@quickstart Desktop]$ hdfs dfs -stat %o /rjlocal/empFirst01
134217728
[cloudera@quickstart Desktop]$ hdfs dfs -stat %r /rjlocal/first01
1
[cloudera@quickstart Desktop]$ hdfs dfs -stat %r /rjlocal/empFirst01
1
[cloudera@quickstart Desktop]$
```

hdfs dfs -stat %y </dir_name>/<file_name> gives last modification date



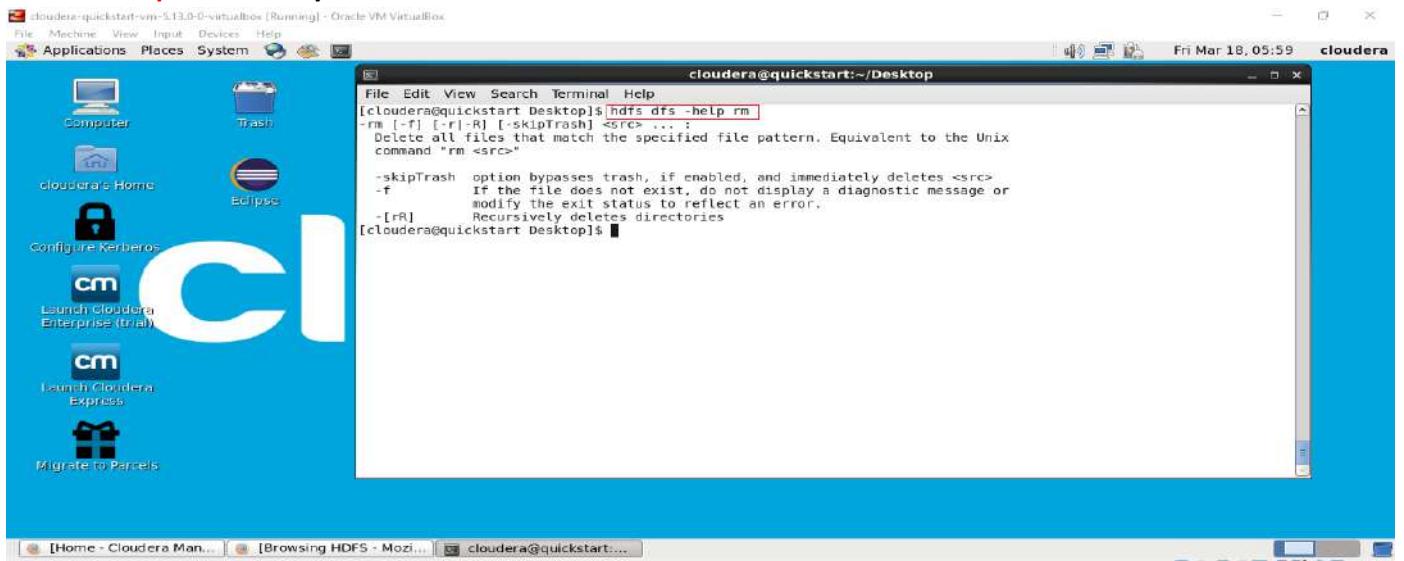
```
[cloudera@quickstart Desktop]$ hdfs dfs -stat /rjlocal
2022-03-18 12:36:11
[cloudera@quickstart Desktop]$ hdfs dfs -ls /rjlocal
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-18 05:36 /rjlocal/empFirst01
-rw-r--r-- 1 hdfs supergroup 31 2022-03-17 13:03 /rjlocal/first01
[cloudera@quickstart Desktop]$ hdfs dfs -stat %b /rjlocal/first01
31
[cloudera@quickstart Desktop]$ hdfs dfs -stat %b /rjlocal/empFirst01
0
[cloudera@quickstart Desktop]$ hdfs dfs -stat %o /rjlocal/first01
134217728
[cloudera@quickstart Desktop]$ hdfs dfs -stat %o /rjlocal/empFirst01
134217728
[cloudera@quickstart Desktop]$ hdfs dfs -stat %r /rjlocal/first01
1
[cloudera@quickstart Desktop]$ hdfs dfs -stat %r /rjlocal/empFirst01
1
[cloudera@quickstart Desktop]$ hdfs dfs -stat %y /rjlocal/first01
2022-03-17 20:03:03
[cloudera@quickstart Desktop]$ hdfs dfs -stat %y /rjlocal/empFirst01
2022-03-18 12:36:11
[cloudera@quickstart Desktop]$
```

24. hdfs dfs -checksum </dir_name>/<file_name> gives checksum of file

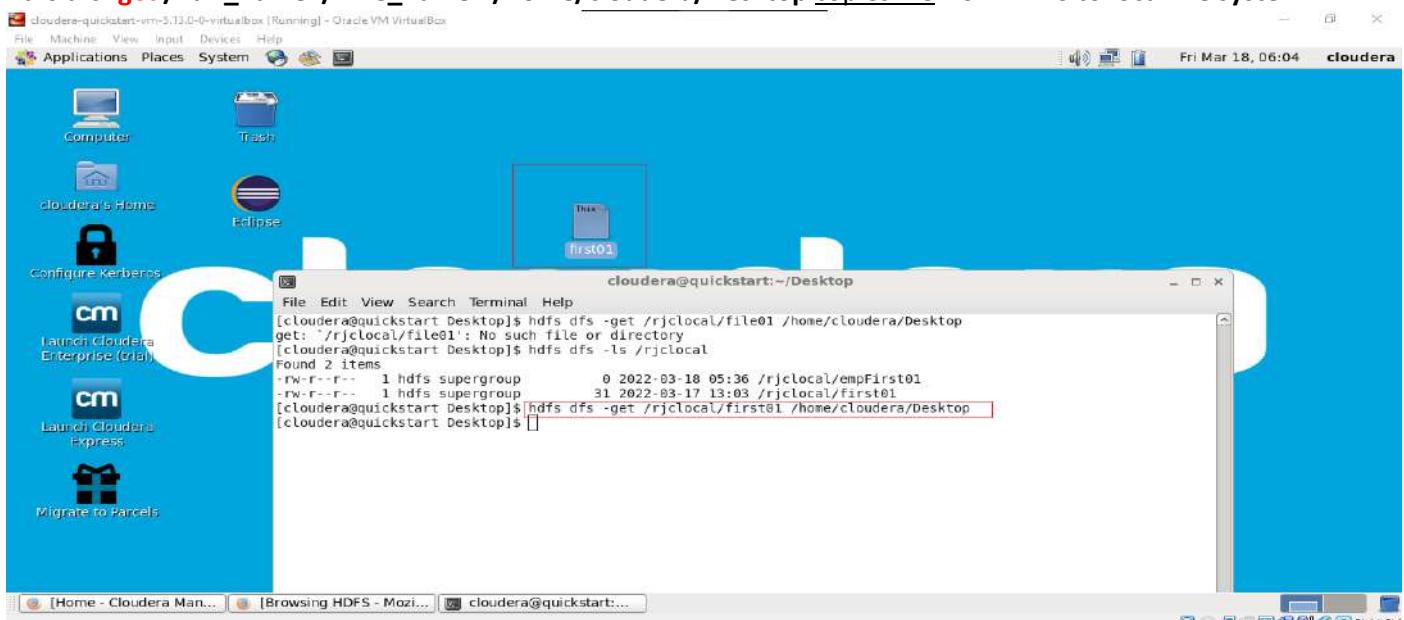


```
[cloudera@quickstart Desktop]$ hdfs dfs -checksum /rjlocal/first01
/rjlocal/first01 MD5-of-BMD5-OF-S12CRC32C 0000D20800000000000000001650a3d56c2872e35e21757f4dc4c13e
[cloudera@quickstart Desktop]$
```

25. `hdfs dfs -help rm` shows syntax of whereas commands



26. `hdfs dfs -get /<dir_name>/<file_name> /home/cloudera/Desktop` copies file from HDFS to local file system



Aim: To Implement WordCount problem using Hadoop MapReduce in Eclipse: (Without Combiner & With Combiner)

What is MapReduce?

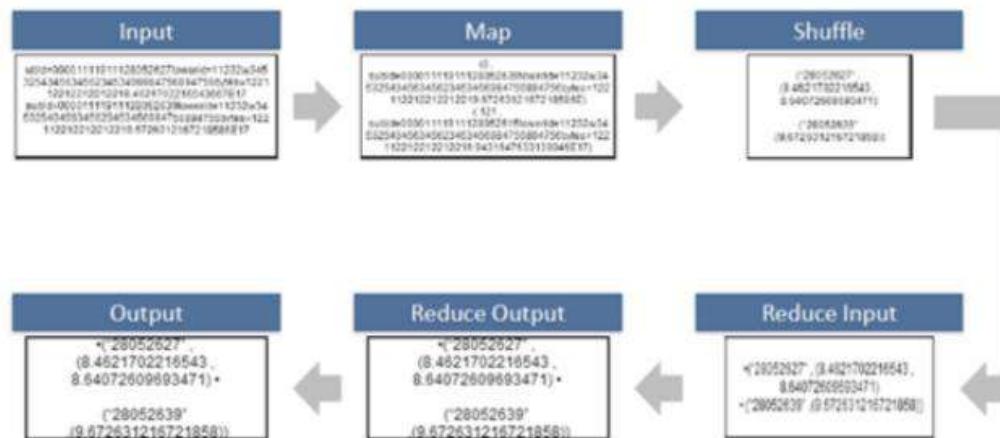
A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce:

"Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

Steps in Map Reduce

- The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

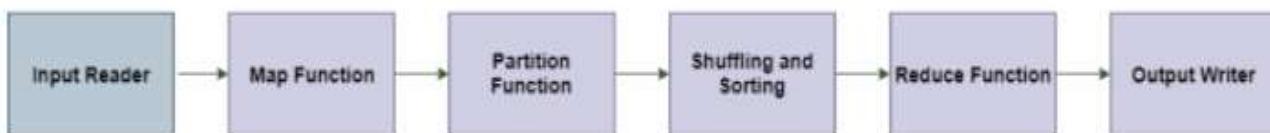


Sort and Shuffle

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

Data Flow In MapReduce

MapReduce is used to compute the huge amount of data. To handle the upcoming data in a parallel and distributed form, the data has to flow from various phases.



Phases of MapReduce data flow

Input reader

The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function.

Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS.

Map function

The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.

Partition function

The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers.

Shuffling and Sorting

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

Reduce function

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

Output writer

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.

To Implement WordCount problem using Hadoop MapReduce in Eclipse:

Hadoop WordCount operation occurs in 3 stages –

- Mapper Phase
- Shuffle Phase
- Reducer Phase

Hadoop WordCount - Mapper Phase Execution

- The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is ‘1’.
- For instance if you consider the sentence “An elephant is an animal”.
- The mapper phase in the WordCount example will split the string into individual tokens i.e. words. In this case, the entire sentence will be split into 5 tokens (one for each word) with a value 1 as shown below –

Key-Value pairs from Hadoop Map Phase Execution-

(an,1)
(elephant,1)
(is,1)
(an,1)
(animal,1)

Hadoop WordCount Example- Shuffle Phase Execution

- After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order.

- After the shuffle phase is executed from the WordCount example code, the output will look like this –

(an,1)
(an,1)
(animal,1)
(elephant,1)
(is,1)

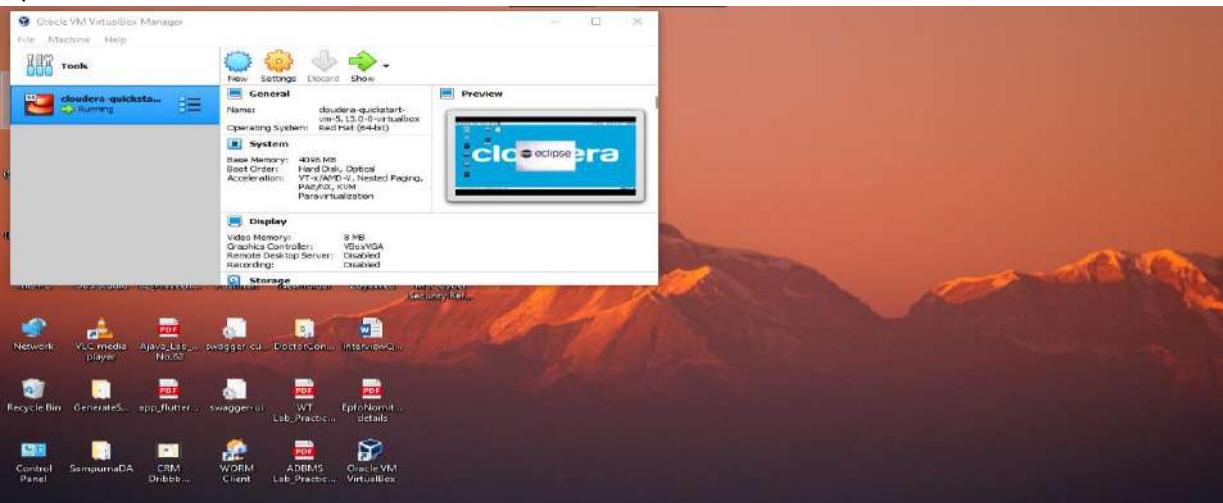
Hadoop WordCount Example- Reducer Phase Execution

- In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word.

- It is like an aggregation phase for the keys generated by the map phase. The reducer phase takes the output of shuffle phase as input and then reduces the key-value pairs to unique keys with values added up.
- In our example “An elephant is an animal.” is the only word that appears twice in the sentence.
- After the execution of the reduce phase of MapReduce WordCount example program, appears as a key only once but with a count of 2 as shown below –
(an,2)
(animal,1)
(elephant,1)
(is,1)
- This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file.
- An important point to note during the execution of the WordCount example is that the mapper class in the WordCount program will execute completely on the entire input file and not just a single sentence.
- Suppose if the input file has 15 lines then the mapper class will split the words of all the 15 lines and form initial key value pairs for the entire dataset.
- The reducer execution will begin only after the mapper phase is executed successfully.

Steps for Word Count in Cloudera: (Without Combiner)

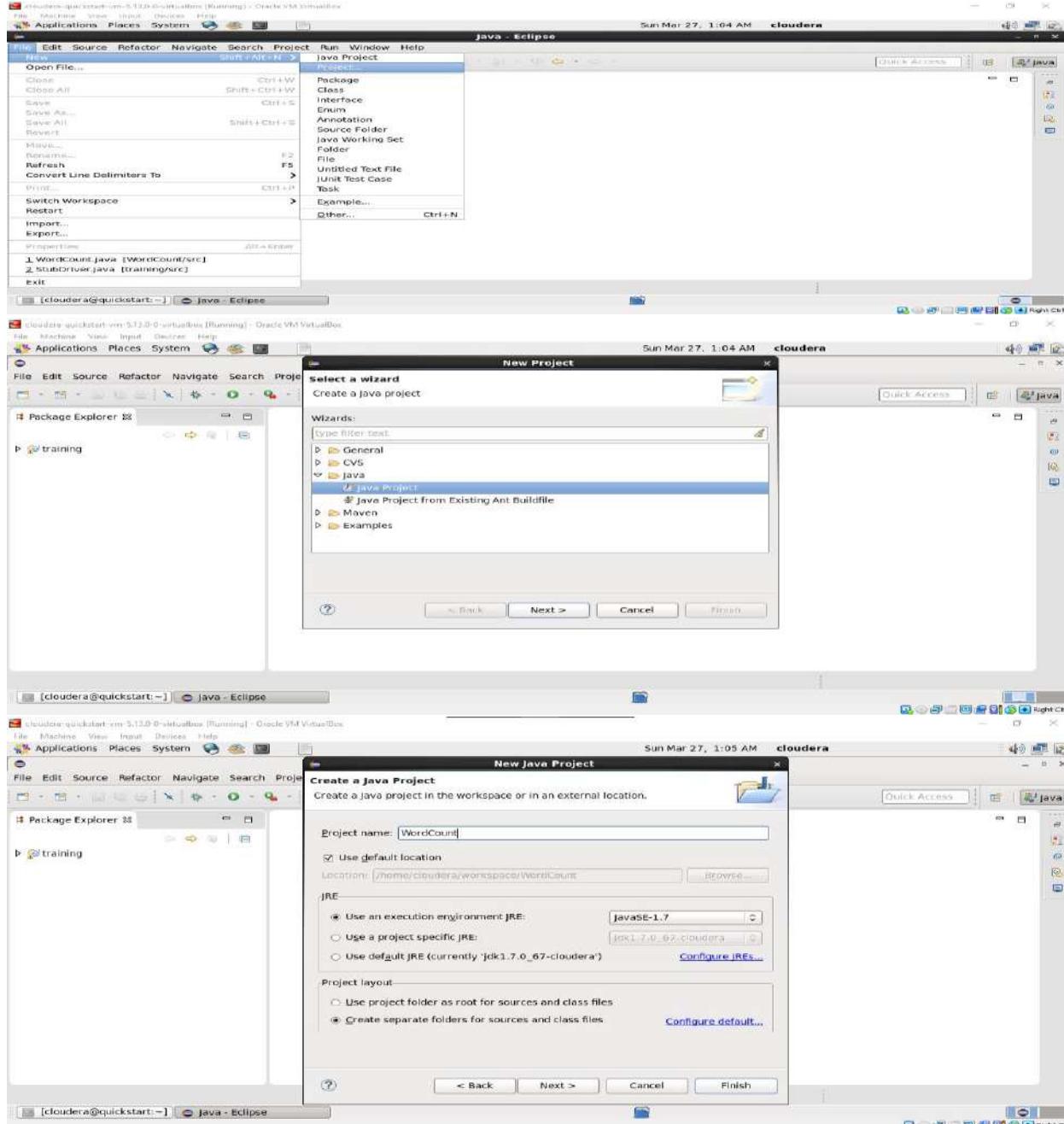
1. Open VirtualBox and then start Cloudera VM



2. Open Eclipse

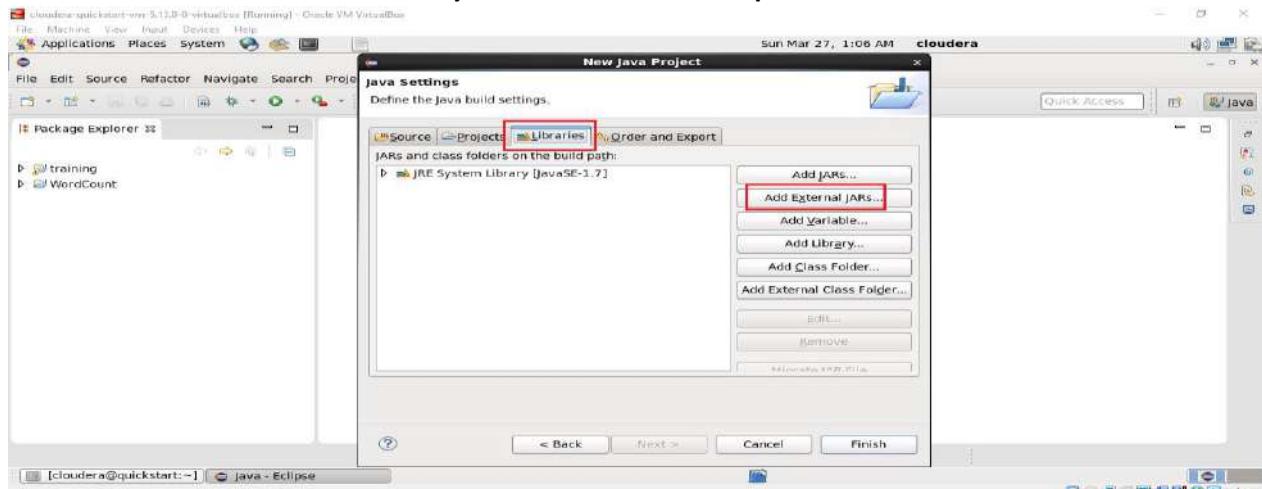


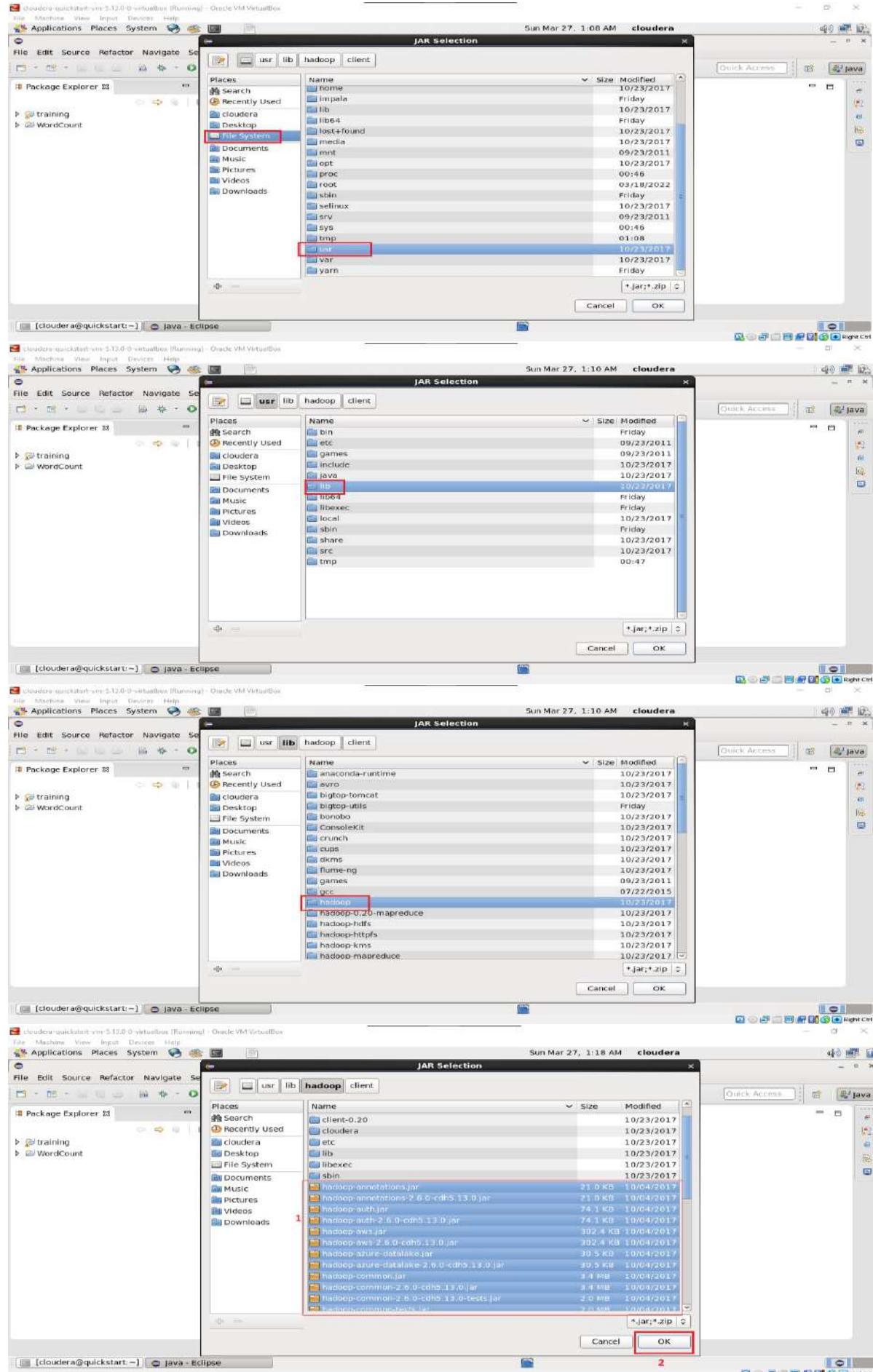
3. Create a New Java Project: File > New > Project > Java Project > Next Set project name as "WordCount".

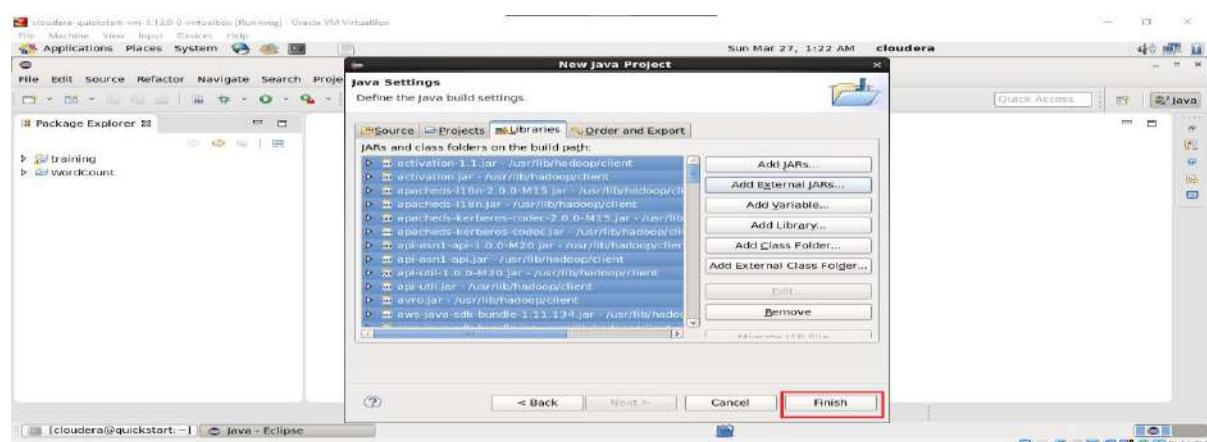
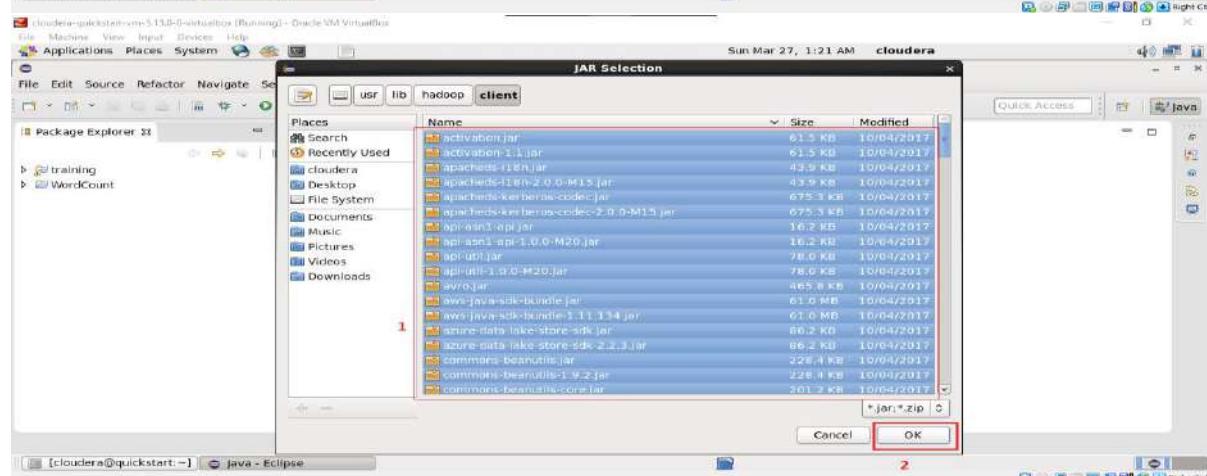
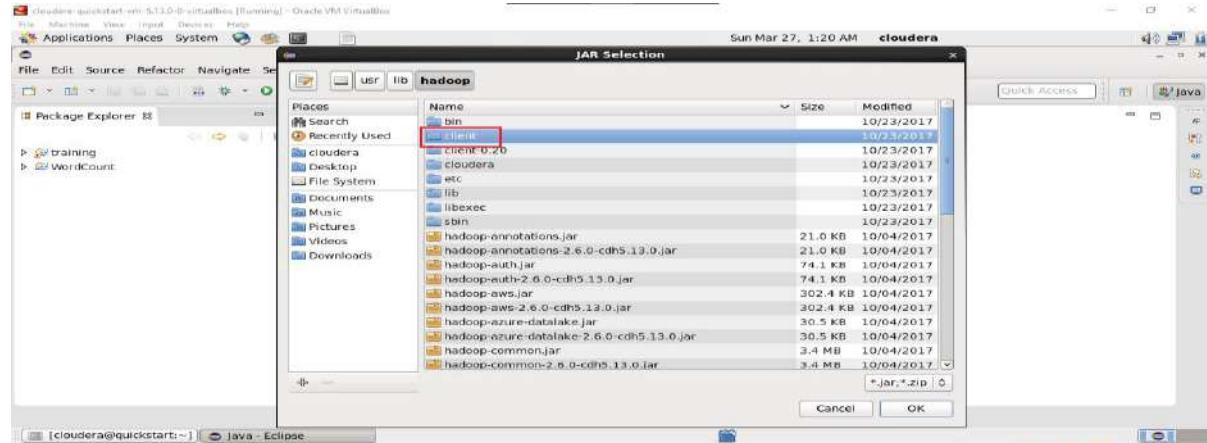
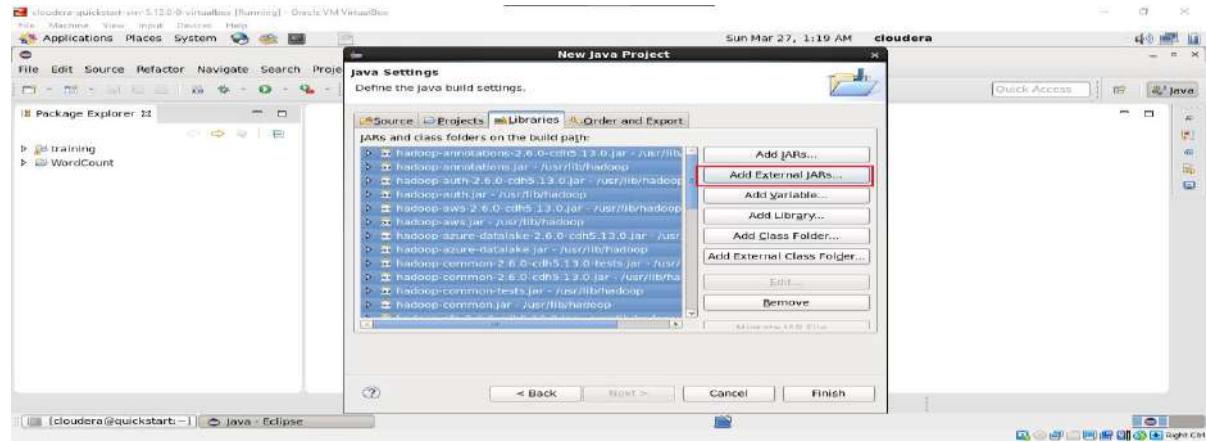


4. Adding the hadoop libraries to the project click on:

Libraries > Add External JARs > File System > usr > lib > hadoop > client > select all JAR files > OK > Finish.

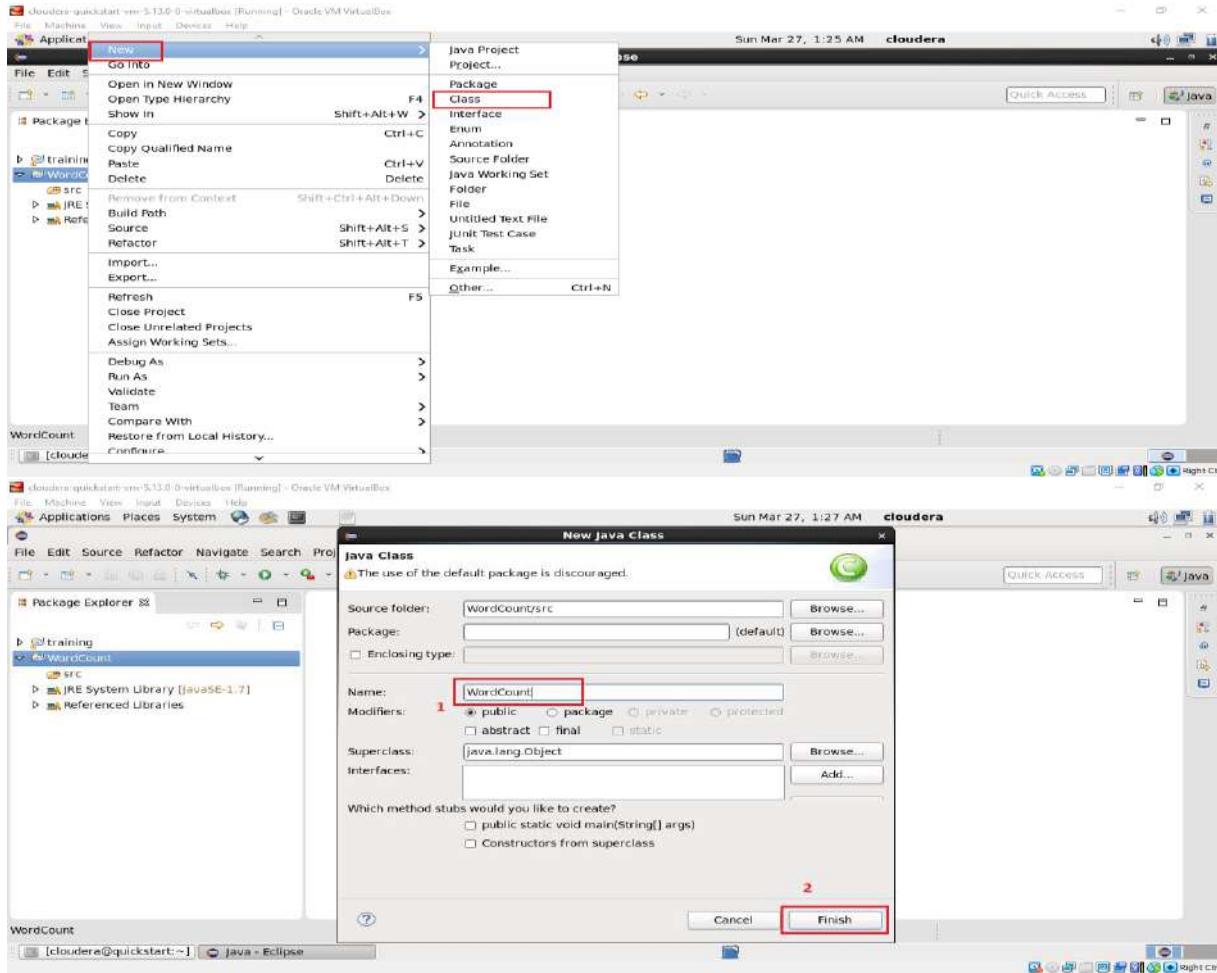






5. Right click on the name of project

"WordCount" > New > Class (Don't write anything for package) > class name "WordCount" > Finish
Then WordCount.java will open if not then open WordCount.java file manually



6. Source Code:

```

1 //Packages
2+ import java.io.IOException;
16
17 public class WordCount {
18     //Map Logic
19+     public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
31
32     //Reducer
33+     public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
46
47     //Main Function
48+     public static void main(String[] args) throws Exception {
62 }
63

```

```

//Packages

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

```

```
//Map Logic
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}

//Reducer
public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
    InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

//Main Function
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    //job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

```

1 //Packages
2 import java.io.IOException;
3 import java.util.StringTokenizer;
4
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.Text;
9
10 import org.apache.hadoop.mapreduce.Job;
11 import org.apache.hadoop.mapreduce.Mapper;
12 import org.apache.hadoop.mapreduce.Reducer;
13
14 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
15 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
16

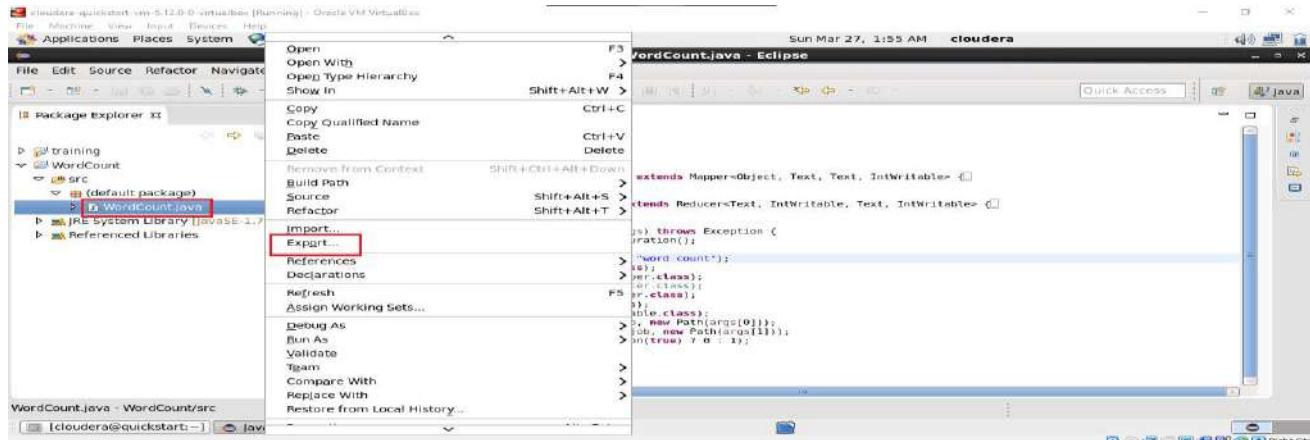
18 //Map Logic
19 public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
20     private final static IntWritable one = new IntWritable(1);
21     private Text word = new Text();
22
23     public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
24         StringTokenizer itr = new StringTokenizer(value.toString());
25         while (itr.hasMoreTokens()) {
26             word.set(itr.nextToken());
27             context.write(word, one);
28         }
29     }
30 }
31
32 //Reducer
33 public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
34     private IntWritable result = new IntWritable();
35
36     public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
37         int sum = 0;
38         for (IntWritable val : values) {
39             sum += val.get();
40         }
41         result.set(sum);
42         context.write(key, result);
43     }
44 }
45
46
47 //Main Function
48 public static void main(String[] args) throws Exception {
49     Configuration conf = new Configuration();
50
51     Job job = Job.getInstance(conf, "word count");
52     job.setJarByClass(WordCount.class);
53     job.setMapperClass(TokenizerMapper.class);
54     //job.setCombinerClass(IntSumReducer.class);
55     job.setReducerClass(IntSumReducer.class);
56     job.setOutputKeyClass(Text.class);
57     job.setOutputValueClass(IntWritable.class);
58     FileInputFormat.addInputPath(job, new Path(args[0]));
59     FileOutputFormat.setOutputPath(job, new Path(args[1]));
60     System.exit(job.waitForCompletion(true) ? 0 : 1);
61 }

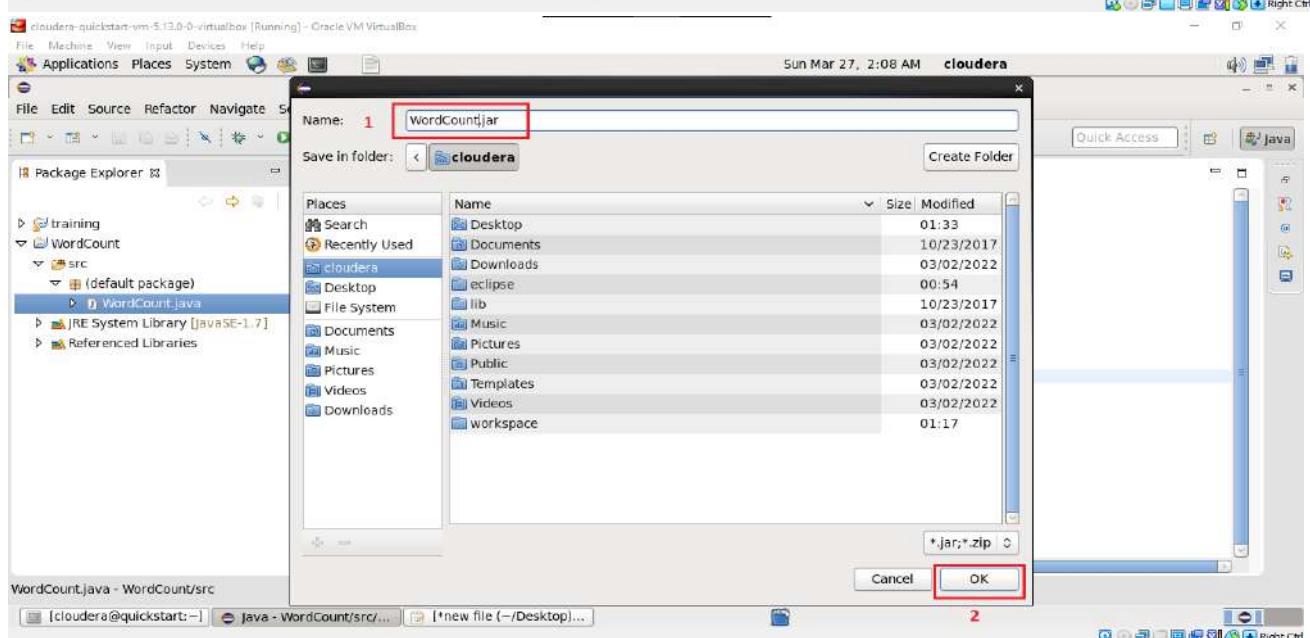
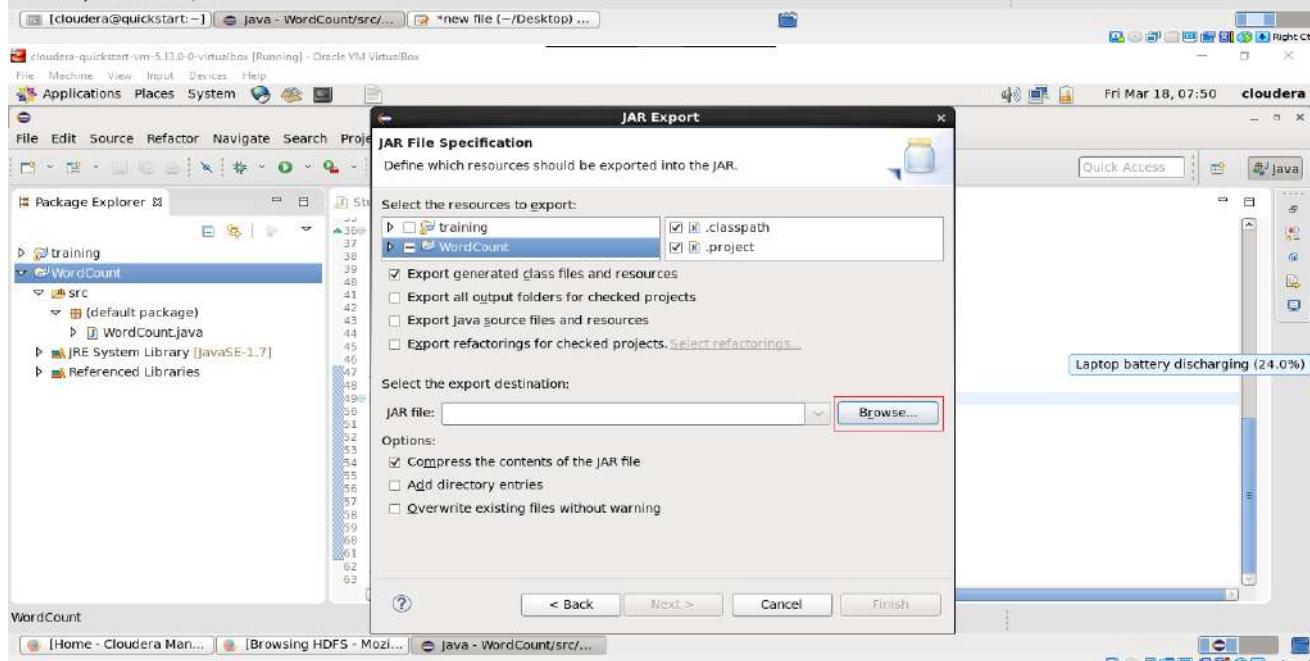
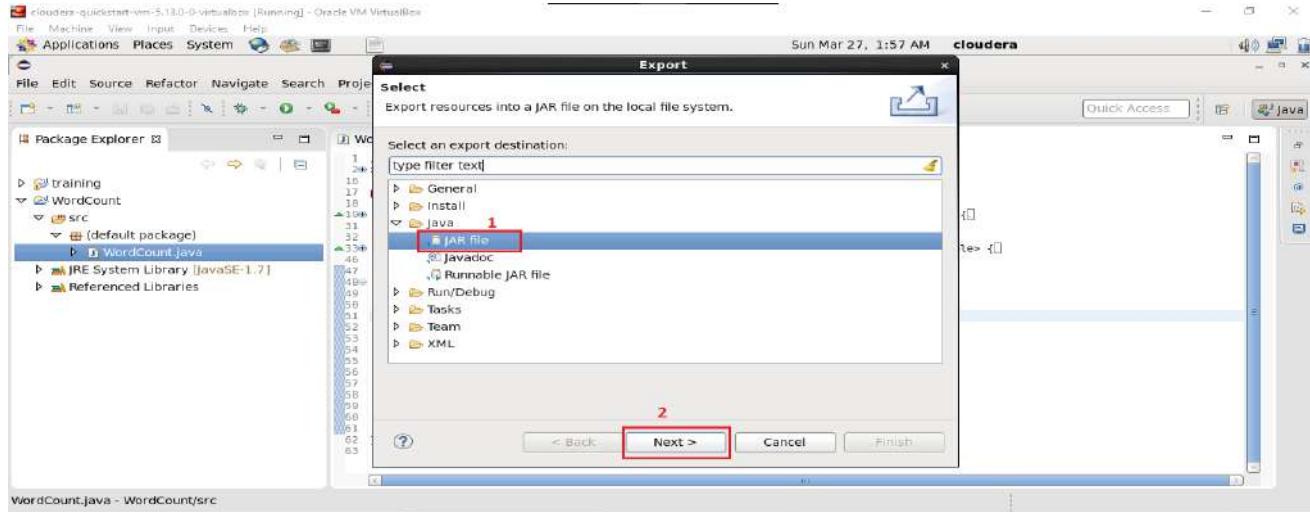
```

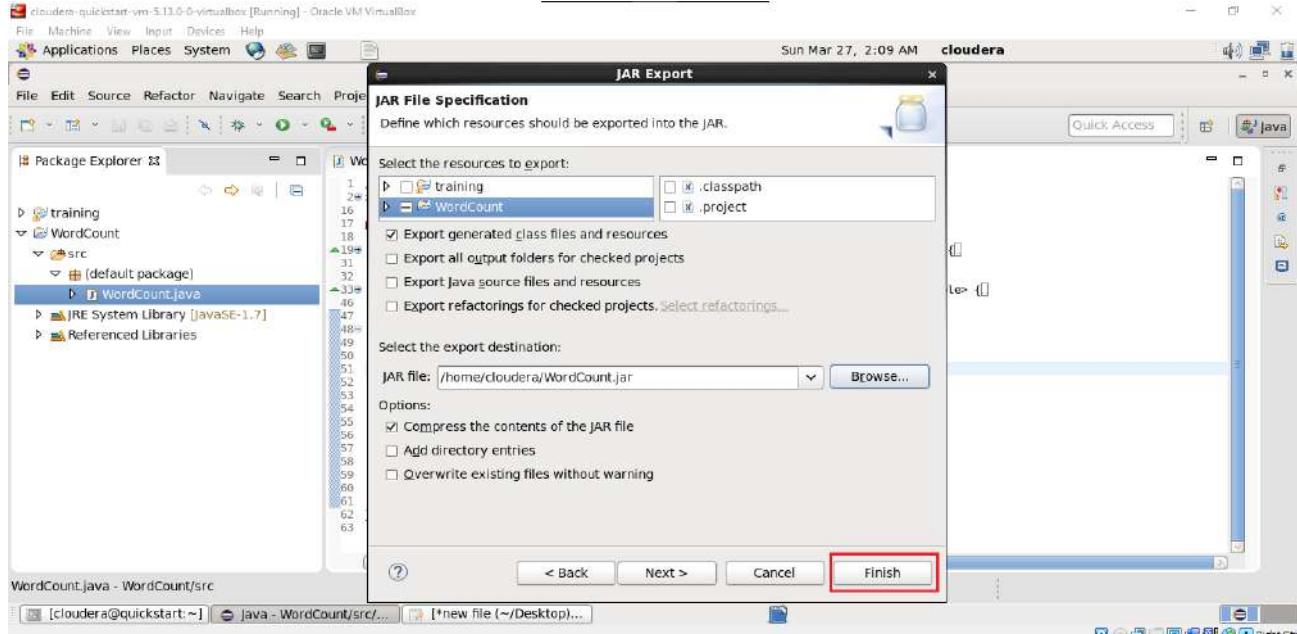
Note: We are running the code without combiner. That is why we commented the combiner line in main function.

7. Right click on the project name

WordCount > Export > Java > JAR File > Next > Jar file “/home/cloudera/WordCount.jar” > Finish.







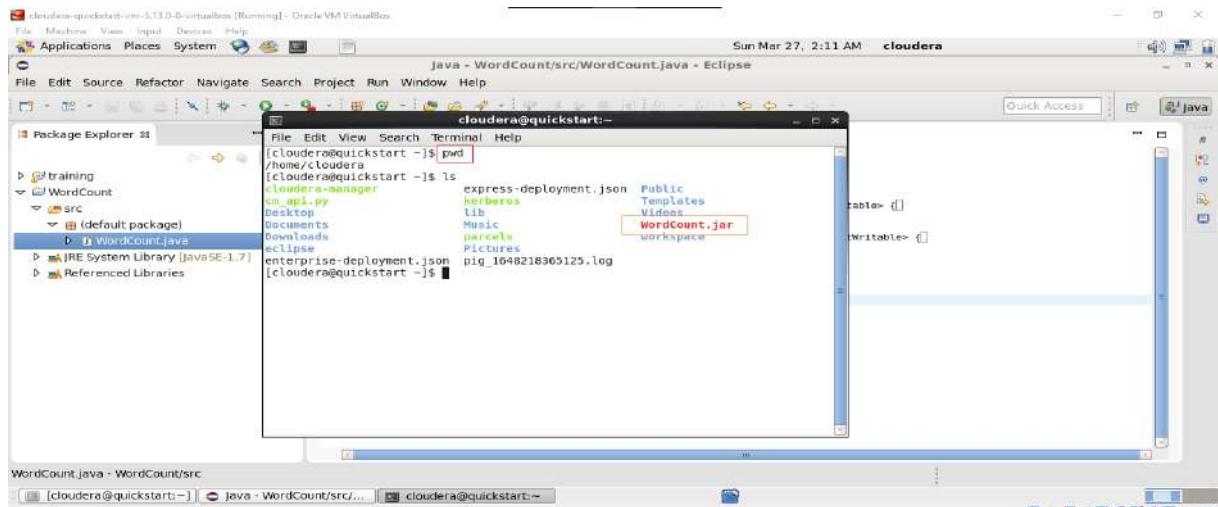
Verify jar file from terminal by using Open terminal & type "ls" There it will show WordCount.jar

Check current working directory

pwd

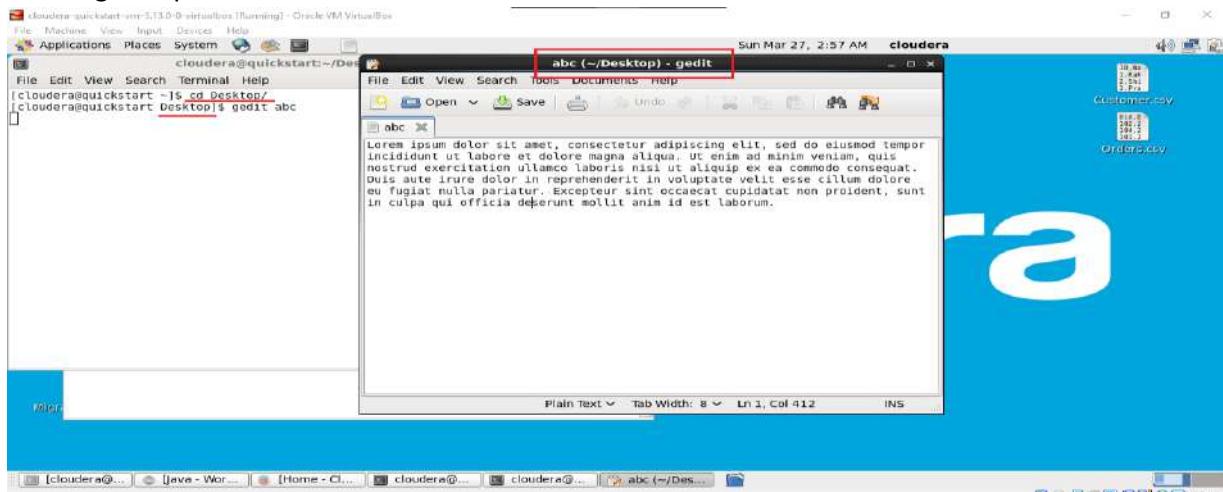
Check the list

ls



8. We need to create an input file in local file system (On Desktop)

Creating an input file named as “abc”.



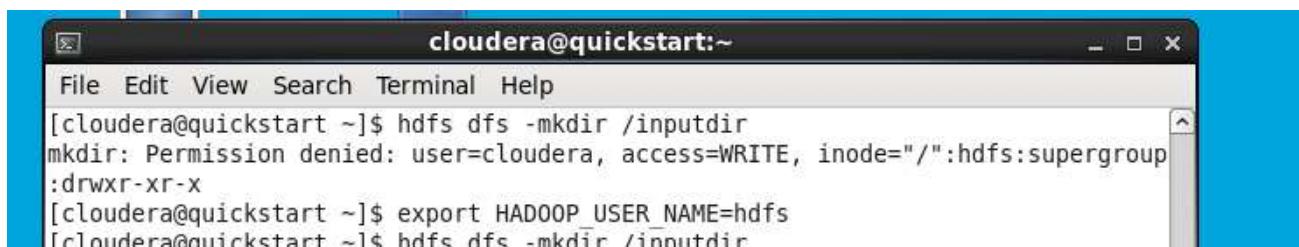
Here listing all the directory present in

hdfs using hdfs dfs -ls /

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase supergroup          0 2022-03-27 00:51 /hbase
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:58 /newdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-18 05:36 /rjclocal
drwxr-xr-x  - solr   solr              0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-25 08:12 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:20 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$
```

9. Now we have to move this input file to hdfs. For this we create a direcory on hdfs using command

hdfs dfs -mkdir /inputdir



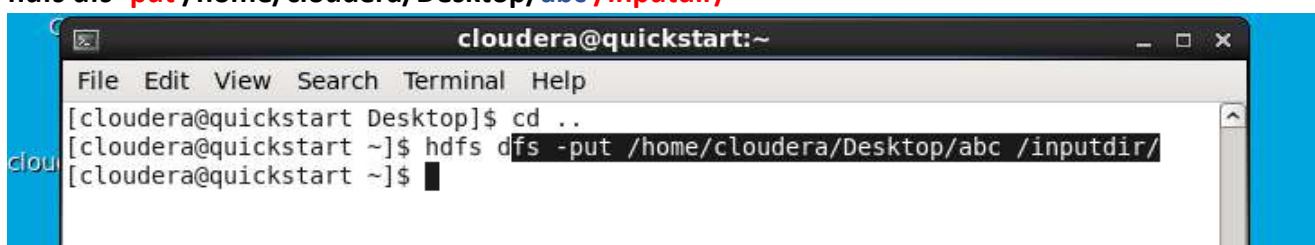
Then we can verify whether this directory is created or not using ls command

hdfs dfs -ls /

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 9 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase supergroup          0 2022-03-27 02:30 /hbase
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 02:46 /inputdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:58 /newdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-18 05:36 /rjclocal
drwxr-xr-x  - solr   solr              0 2017-10-23 09:18 /solr
drwxrwxrwt  - hdfs  supergroup          0 2022-03-25 08:12 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:20 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$
```

Move the input file to this directory created in hdfs by using either **put** command or **copyFromLocal** command.

hdfs dfs -put /home/cloudera/Desktop/abc /inputdir/



Now checking whether the “abc” present in /inputdir directory of hdfs or not using command

hdfs dfs -ls /inputdir

```
[cloudera@quickstart ~]$ hdfs dfs -ls /inputdir/
Found 1 items
-rw-r--r-- 1 hdfs supergroup          446 2022-03-27 02:55 /inputdir/abc
[cloudera@quickstart ~]$
```

As we can see “abc” file is present in /inputdir directory of hdfs. Now we will see the content of this file using command

hdfs dfs -cat /inputdir/abc

```
[cloudera@quickstart ~]$ hdfs dfs -cat /inputdir/abc
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor i-
ncididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostru-
d exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aut-
e irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat n-
ulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui-
officia deserunt mollit anim id est laborum.
[cloudera@quickstart ~]$
```

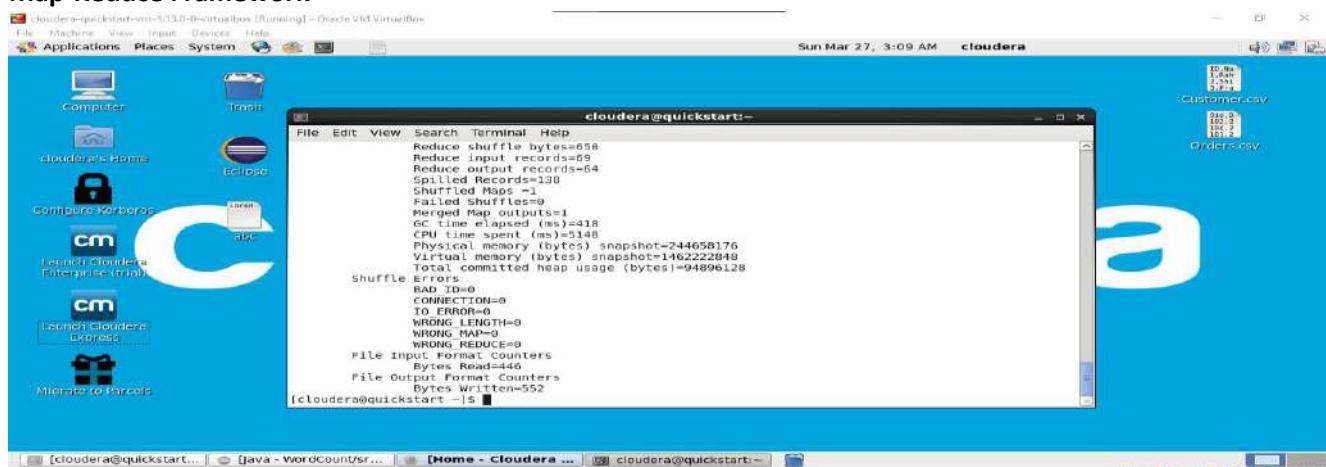
10. Running Mapreduce Program on Hadoop,

Syntax : **hadoop jar jarFileName.jar ClassName /InputFileAddress /outputdir**

hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /outputdir

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /outputdir
22/03/27 03:05:46 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
```

Map-Reduce Framework



As we can see in the above output,

Combine input records=0

Combine output records=0

Note: We are getting this because we have commented the Combiner line in main function.

And Reduce shuffle bytes coming as,

Reduce shuffle bytes=1876

So when we are not using combiner 1876 bytes acting as an input for the reducer.

11. Then we can verify the content of tempop1 directory and in that part-r file has the actual output by using the command Hdfs dfs -cat /tempop1/part-r-00000 This will give us final output. The same file can also be accessed using a browser. For every execution of this program we need to delete the output directory or give a new name to the output directory every time. 1st we are checking whether the tempop1 directory is created in hdfs or not using command

hdfs dfs -ls /

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 10 items
drwxrwxrwx  -  hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  -  hbase supergroup          0 2022-03-27 02:30 /hbase
drwxr-xr-x  -  hdfs  supergroup          0 2022-03-27 02:55 /inputdir
drwxr-xr-x  -  hdfs  supergroup          0 2022-03-17 07:58 /newdir
drwxr-xr-x  -  hdfs  supergroup          0 2022-03-27 03:08 /outputdir
drwxr-xr-x  -  hdfs  supergroup          0 2022-03-18 05:36 /rjclocal
drwxr-xr-x  -  solr   solr              0 2017-10-23 09:18 /solr
drwxrwxrwt  -  hdfs  supergroup          0 2022-03-25 08:12 /tmp
drwxr-xr-x  -  hdfs  supergroup          0 2022-03-17 07:20 /user
drwxr-xr-x  -  hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$ ls
```

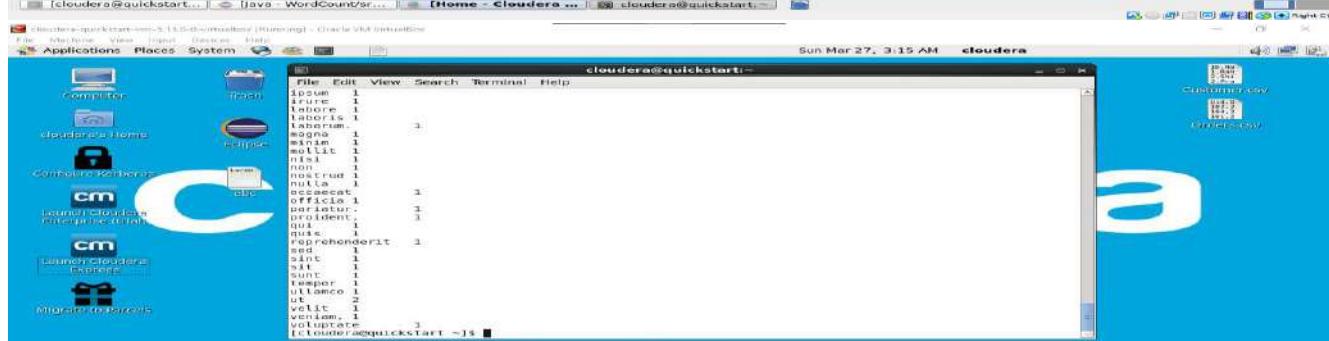
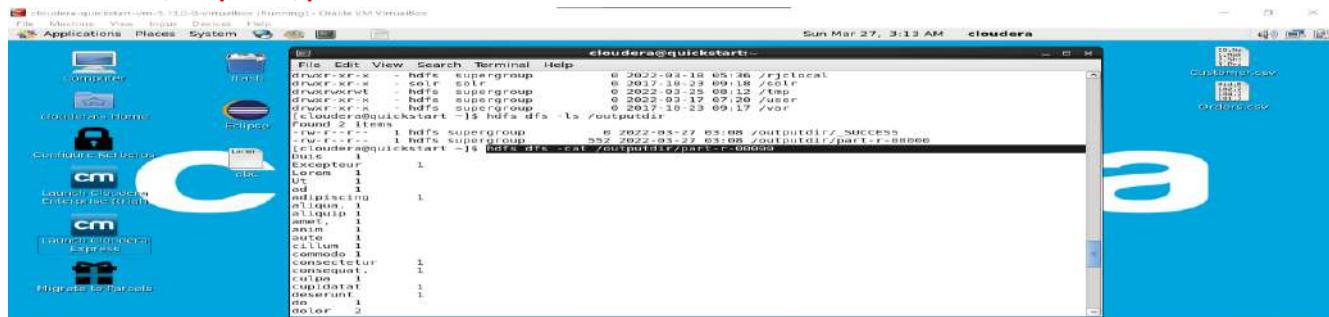
Now let's check what we have inside this **outputdir** directory using command as

hdfs dfs –ls /outputdir

```
[cloudera@quickstart ~]$ hdfs dfs -ls /outputdir
Found 2 items
-rw-r--r--  1 hdfs supergroup          0 2022-03-27 03:08 /outputdir/_SUCCESS
-rw-r--r--  1 hdfs supergroup  552 2022-03-27 03:08 /outputdir/part-r-00000
[cloudera@quickstart ~]$
```

Now we want to read the content of the part-r-00000 file which present inside the outputdir using command

hdfs dfs -cat /outputdir/part-r-00000

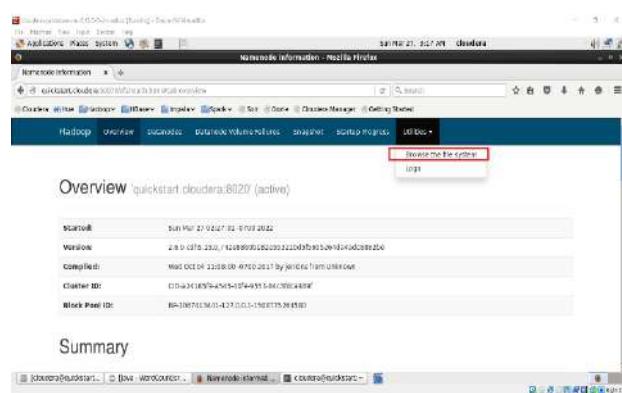
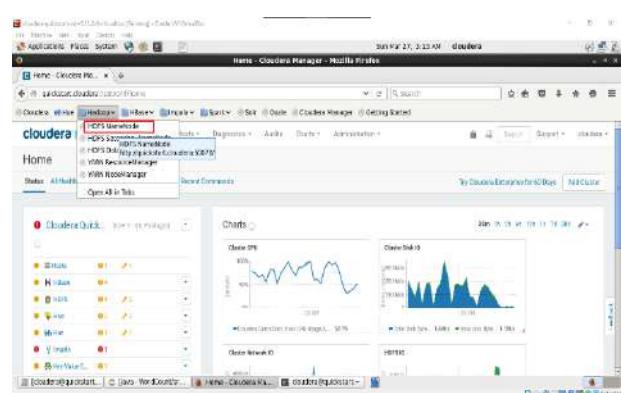


It will give the count of number of times each word has occurred as output

12. The same file can also be accessed using a browser.

Browse the Directory by:

Hadoop->HDFS Namenode->Utilities ->Browse the file system



Browsing HDFS - Mozilla Firefox

Sun Mar 27, 3:20 AM cloudera

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

/outputdir Got

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	hdfs	supergroup	0 B	Mon Oct 23 09:15:43 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Sun Mar 27 02:30:39 -0700 2022	0	0 B	hbase
drwxr-xr-x	hdfs	supergroup	0 B	Sun Mar 27 02:55:49 -0700 2022	0	0 B	inputdir
drwxr-xr-x	hdfs	supergroup	0 B	Thu Mar 17 07:58:45 -0700 2022	0	0 B	newdir
drwxr-xr-x	hdfs	supergroup	0 B	Sun Mar 27 03:08:55 -0700 2022	0	0 B	outputdir
drwxr-xr-x	hdfs	supergroup	0 B	Fri Mar 18 05:36:11 -0700 2022	0	0 B	rjlocal
drwxr-xr-x	solr	solr	0 B	Mon Oct 23 09:18:01 -0700 2017	0	0 B	solr
drwxrwxrwt	hdfs	supergroup	0 B	Fri Mar 25 08:12:06 -0700 2022	0	0 B	tmp
drwxr-xr-x	hdfs	supergroup	0 B	Thu Mar 17 07:20:17 -0700 2022	0	0 B	user
drwxr-xr-x	hdfs	supergroup	0 B	Mon Oct 23 09:17:24 -0700 2017	0	0 B	var

[cloudera@quickstart... Java - WordCount/sr... Browsing HDFS - Mozilla Firefox cloudera@quickstart:~]

File Machine View Input Devices Help Applications Places System Browsing HDFS - Mozilla Firefox Sun Mar 27, 3:22 AM cloudera

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

Hadoop overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/outputdir Got

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hdfs	supergroup	0 B	Sun Mar 27 03:08:55 -0700 2022	1	128 MB	SUCCESS
-rw-r--r--	hdfs	supergroup	552 B	Sun Mar 27 03:08:52 -0700 2022	1	128 MB	part-r-00000

Hadoop, 2017.

Now downloading the part-r-00000 file.

Browsing HDFS - Mozilla Firefox

Sun Mar 27, 3:23 AM cloudera

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

Hadoop overview Datanodes Snapshot Startup Progress Utilities

File information - part-r-00000

Download

Opening part-r-00000

You have chosen to open:
part-r-00000
which is: BIN file (552 bytes)
from: http://quickstart.cloudera:50070

Would you like to save this file?

Cancel Save File

Availability:
quickstart.cloudera

Block Size Name

128 MB SUCCESS

128 MB part-r-00000

[cloudera@quickstart... Java - WordCount/sr... Browsing HDFS - Mozilla Firefox cloudera@quickstart:~]

File Machine View Input Devices Help Applications Places System Browsing HDFS - Mozilla Firefox Sun Mar 27, 3:25 AM cloudera

part-r-00000 (~/Downloads) gedit

File Edit View Search Tools Documents Help

part-r-00000

```
laborum, 1
laboris, 1
laborum, 1
magna, 1
mollit, 1
nisi, 1
non, 1
nostrud, 1
nulla, 1
officia, 1
pariatur, 1
prudent, 1
qui, 1
quis, 1
reprehenderit, 1
tempor, 1
sint, 1
sit, 1
ut, 1
tempor, 1
ullamco, 1
ut, 2
velit, 1
veniam, 1
voluptate, 1
```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

For every execution of this program we need to delete the output directory or give a new name to the output directory every time.

Inside the part-r-00000 file it will have the same output as we are getting after executing using command
hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /op1

```

cloudera@quickstart:~$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdir/abc /op1
22/03/27 03:28:46 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera:8082
22/03/27 03:28:46 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed
22/03/27 03:28:46 WARN mapreduce.JobResourceUploader: To get help, run /usr/lib/hadoop/bin/hadoop --help
22/03/27 03:29:00 INFO input.FileInputFormat: Total input paths to process : 1
22/03/27 03:29:00 INFO mapreduce.JobSubmitter: number of splits:1
22/03/27 03:29:00 INFO mapreduce.JobSubmitter: Submitting application application_1648373372244_0002
22/03/27 03:29:10 INFO impl.YarnClientImpl: Submitted application application_1648373372244_0002
22/03/27 03:29:12 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/
22/03/27 03:29:12 INFO mapreduce.Job: Running job: job_1648373372244_0002
22/03/27 03:29:12 INFO mapreduce.Job: Job job_1648373372244_0002 running in uber mode : false
22/03/27 03:29:12 INFO mapreduce.Job: map 0% reduce 0%
22/03/27 03:29:49 INFO mapreduce.Job: map 100% reduce 0%
22/03/27 03:31:11 INFO mapreduce.Job: map 100% reduce 16%
22/03/27 03:31:13 INFO mapreduce.Job: Job job_1648373372244_0002 completed successfully
22/03/27 03:31:13 INFO mapreduce.Job: Counters:
FILE: Number of bytes read=54
FILE: Number of bytes written=205227
FILE: Number of read operations=0
FILE: Number of large read operations=0

File System counters:
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=17188672
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=33550
Total time spent by all reduce tasks (ms)=19515
Total memory-milliseconds taken by all map tasks=23556
Total vcore-milliseconds taken by all reduce tasks=19515
Total megabyte-milliseconds taken by all map tasks=17188672
Total megabyte-milliseconds taken by all reduce tasks=9991688

Map-Reduce Framework
Map input records=0
Reduce input records=0
Map output records=0
Map output bytes=0
Map output materialized bytes=0
Timeline:
Combine input records=0
Combine output records=0
Reduce shuffle bytes=0
Reduce input records=0
Reduce output bytes=0
Spilled Records=128
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=3
GC Time elapsed (ms)=0
CPU time spent (ms)=4070
Physical memory (bytes) snapshot=244066368
Virtual memory (bytes) snapshot=1481924688
Total committed heap usage (bytes)=50593280

[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 11 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-27 02:30 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-27 02:55 /inputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:31 /op1
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:08 /outputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-18 05:36 /rjcclocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-25 08:12 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$ hdfs dfs -ls /op1
Found 2 items
-rw-r--r-- 1 hdfs supergroup 0 2022-03-27 03:31 /op1/_SUCCESS
-rw-r--r-- 1 hdfs supergroup 552 2022-03-27 03:31 /op1/part-r-00000
[cloudera@quickstart ~]$

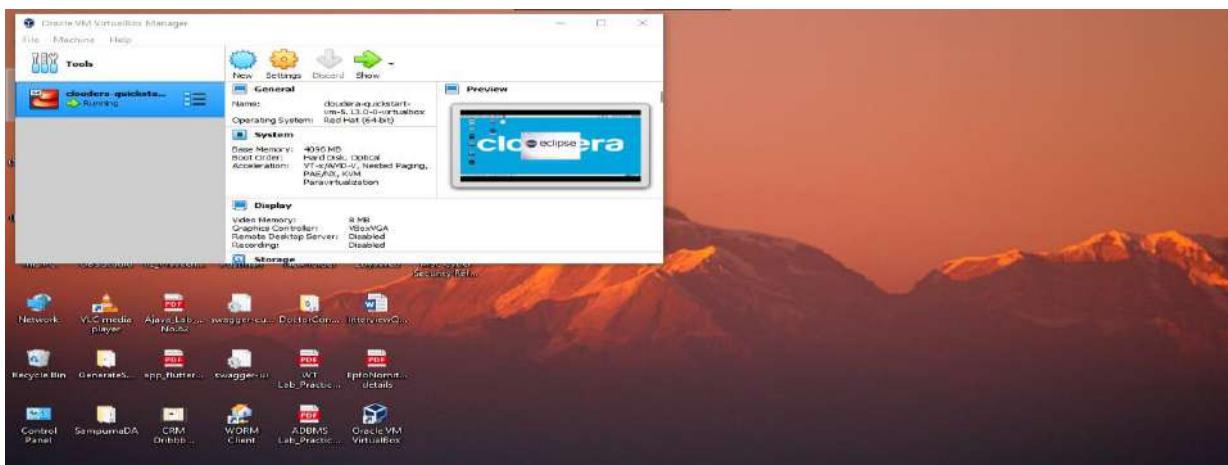
```

Aim: To determine maximum temperature using Hadoop MapReduce**Dataset – temperature.txt**

0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999
 0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999
 0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999
 0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999
 0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

Steps to determine maximum temperature using Hadoop MapReduce in Cloudera:

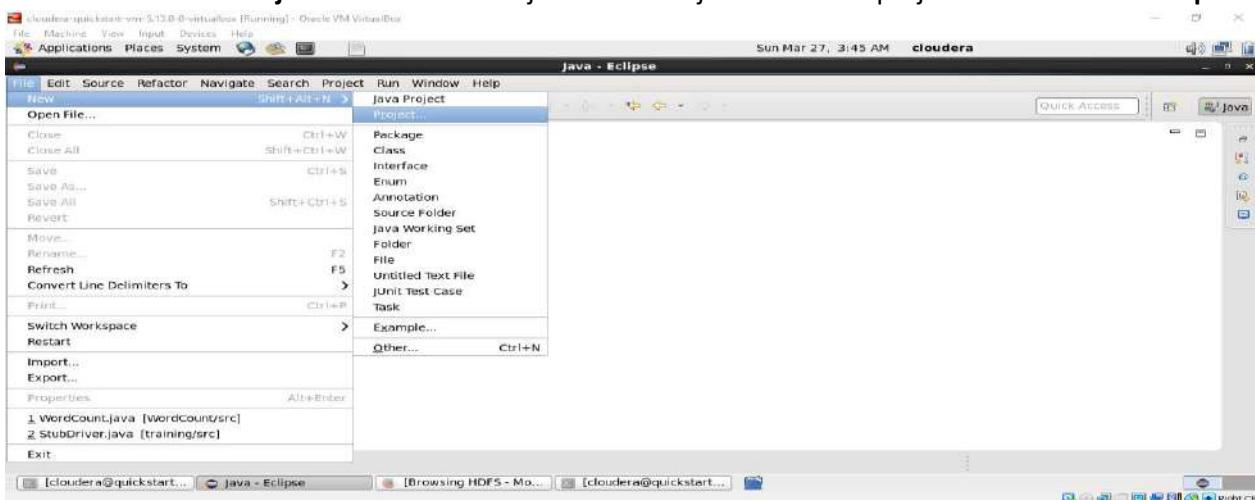
1. Open VirtualBox and then start Cloudera VM

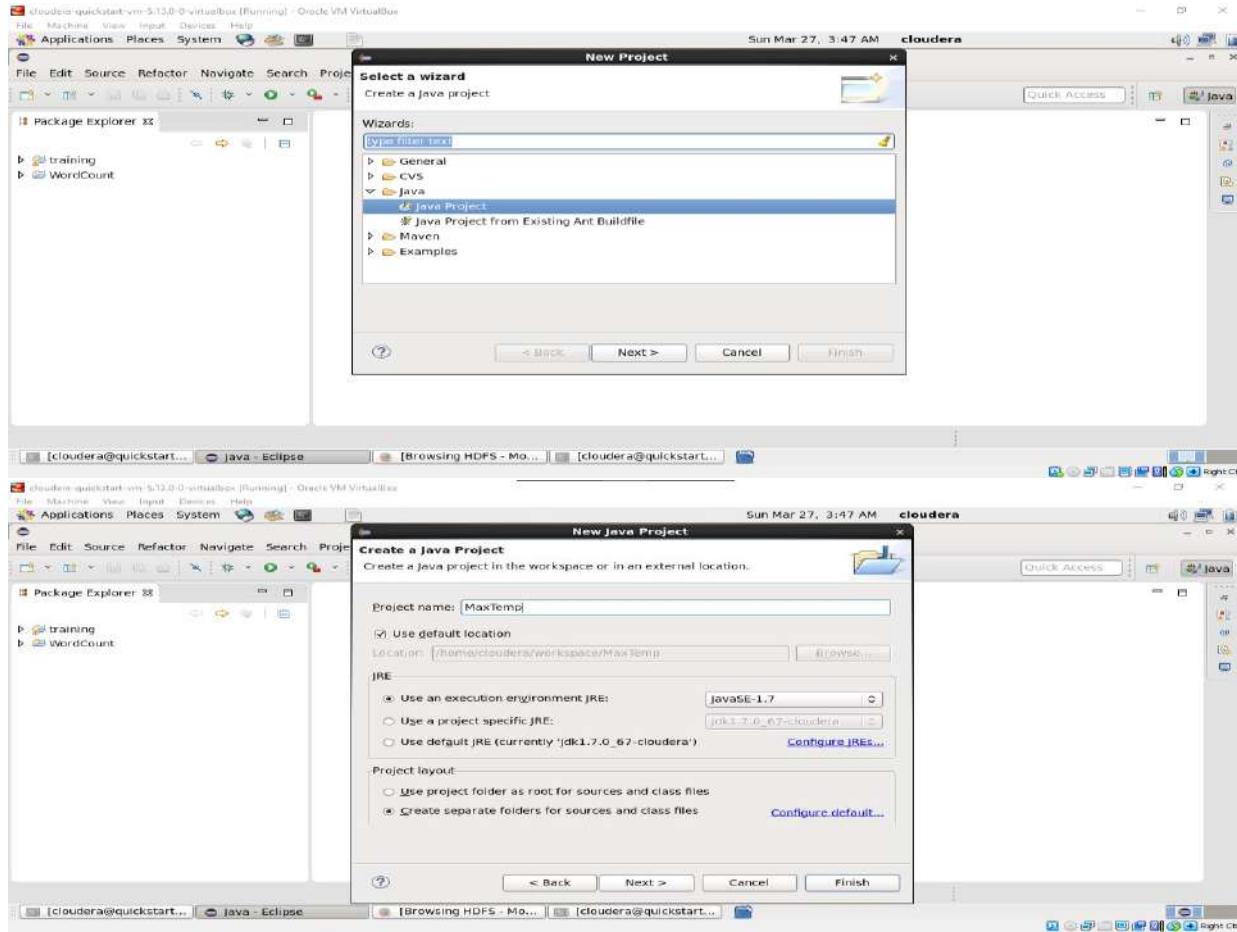


2. Open Eclipse



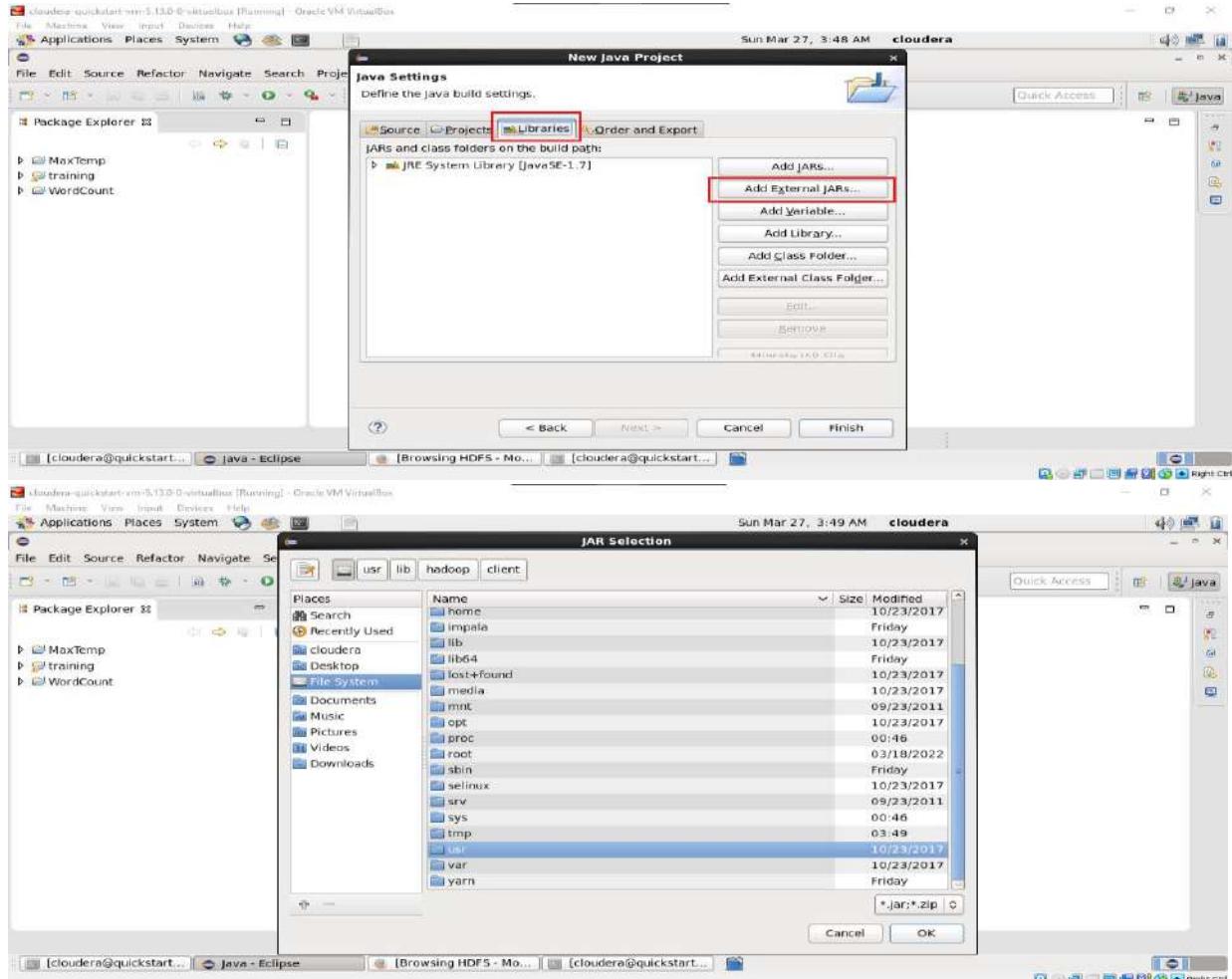
3. Create a New Java Project: File > New > Project > Java Project > Next Set project name as "MaxTemp".

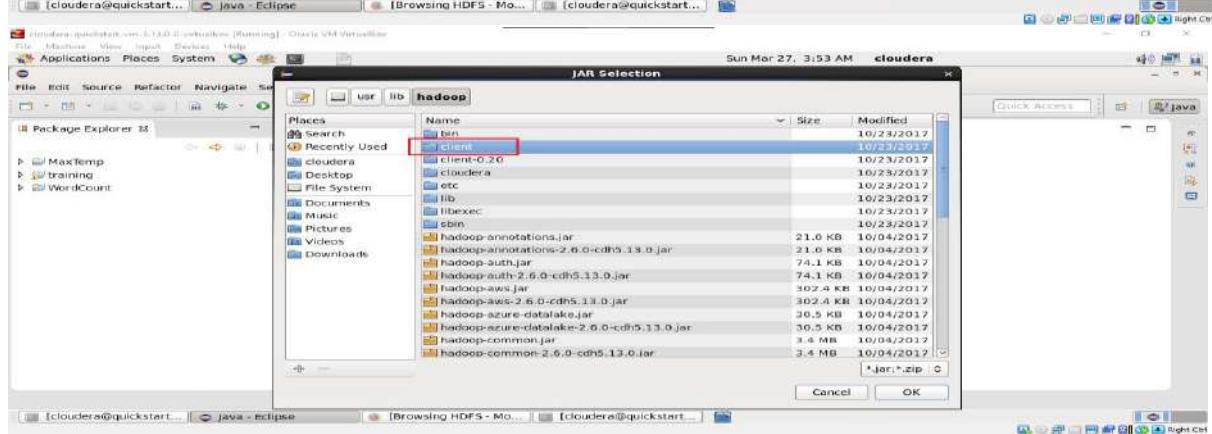
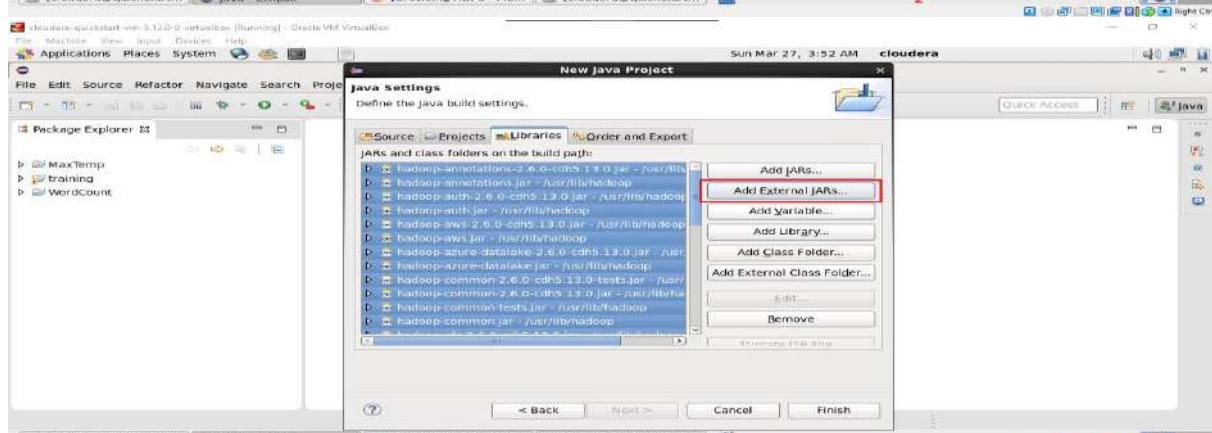
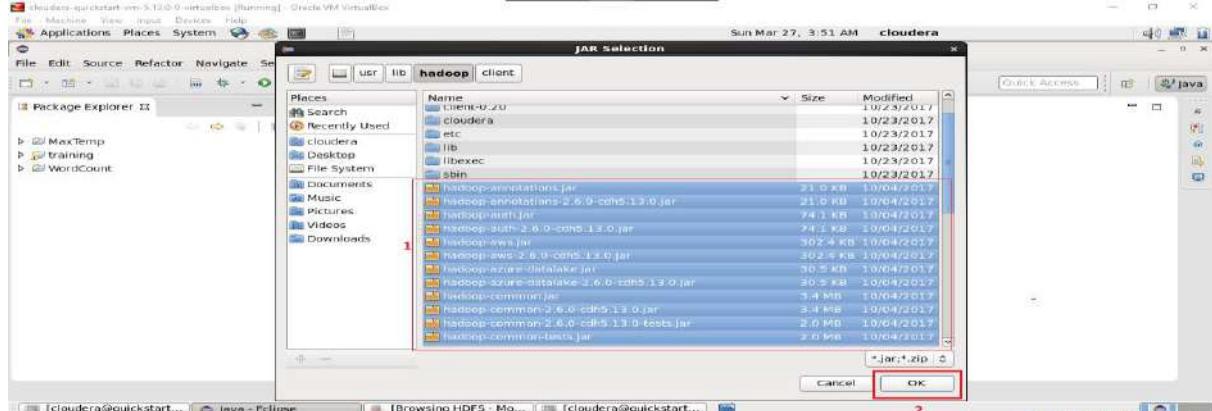
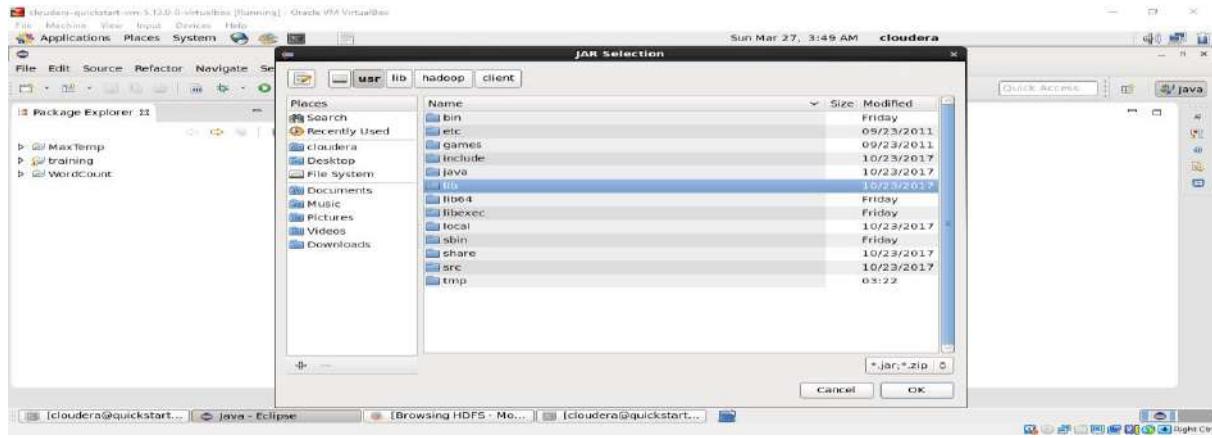


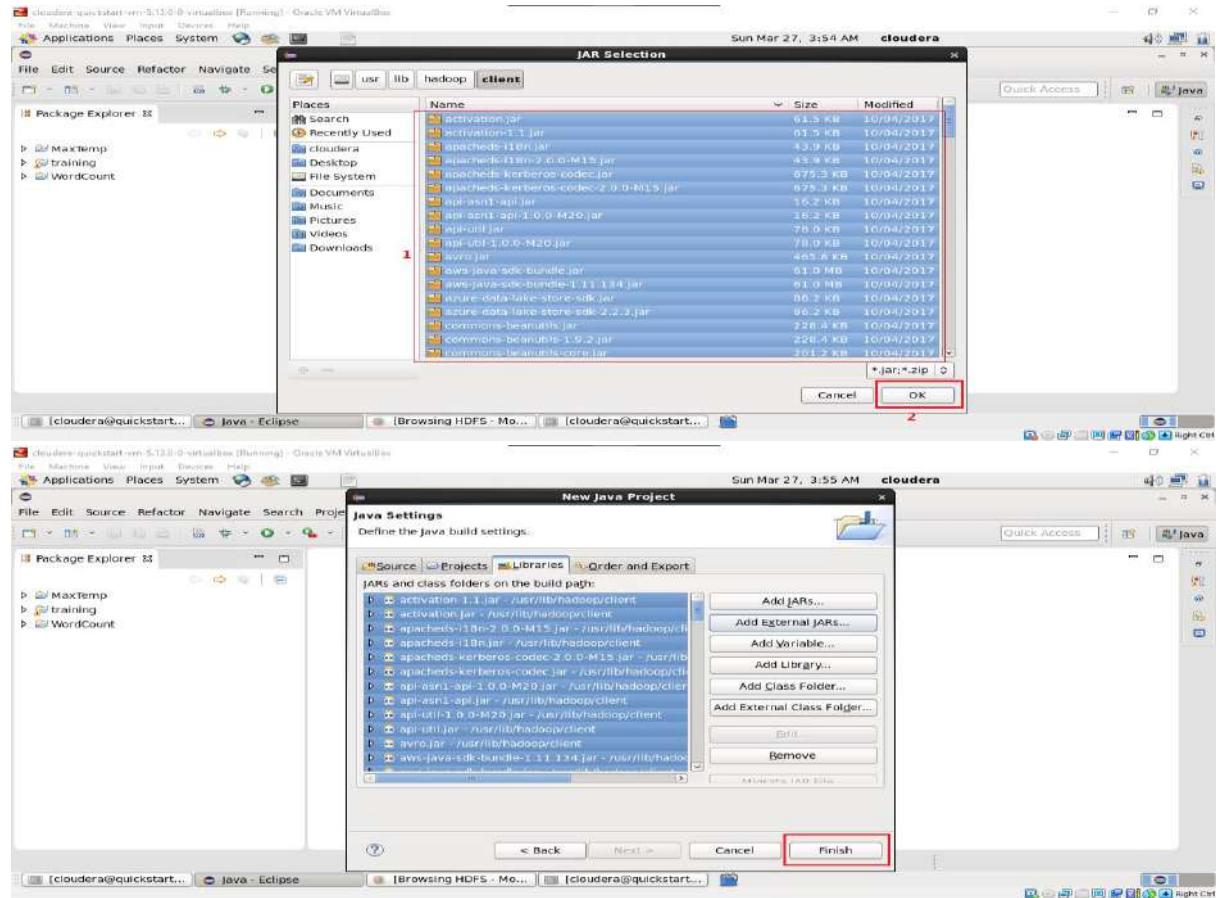


4. Adding the hadoop libraries to the project click on:

Libraries > Add External JARs > File System > usr > lib > hadoop > client > select all JAR files > OK > Finish.



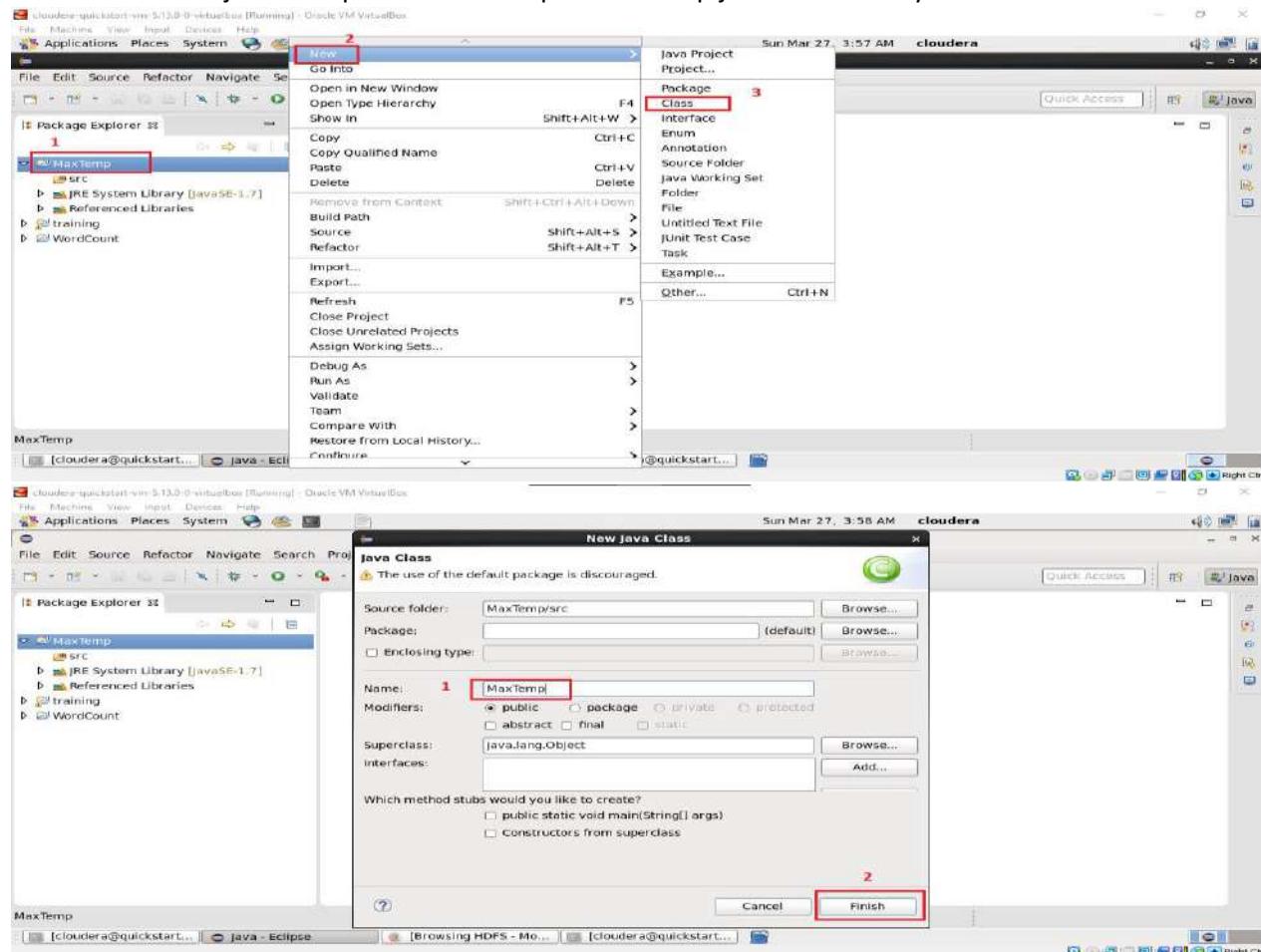




5. Right click on the name of project

"MaxTemp" > New > Class (Don't write anything for package) > class name "MaxTemp" > Finish

Then WordCount.java will open if not then open MaxTemp.java file manually



6. Source Code:

```
1 //Packages
2
3+import java.io.IOException;
18
19 public class MaxTemp {
20     //Map Logic
21+    public static class MaxTemperatureMapper extends Mapper < LongWritable, Text, Text, IntWritable > {
39
40     //Reducer
41+    public static class MaxTemperatureReducer extends Reducer < Text, IntWritable, Text, IntWritable > {
50         public void reduce(Text key, Iterable < IntWritable > values, Context context) throws IOException, InterruptedException {
51             int maxvalue = Integer.MIN_VALUE;
52+            public static void main(String[] args) throws Exception {
70 }
71 }
```

```
//Packages

import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MaxTemp {
    //Map Logic
    public static class MaxTemperatureMapper extends Mapper < LongWritable, Text, Text,
IntWritable > {

        public static final int MISSING = 9999;

        public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
            String line = value.toString();
            String year = line.substring(15, 19);
            int airtemp;
            if (line.charAt(87) == '+') {
                airtemp = Integer.parseInt(line.substring(88, 92));
            } else
                airtemp = Integer.parseInt(line.substring(87, 92));
            String q = line.substring(92, 93);
            if (airtemp != MISSING && q.matches("[01459]")) {
                context.write(new Text(year), new IntWritable(airtemp));
            }
        }
    }

    //Reducer
    public static class MaxTemperatureReducer extends Reducer < Text, IntWritable, Text,
IntWritable > {
```

```

        public void reduce(Text key, Iterable < IntWritable > values, Context context) throws
IOException, InterruptedException {
    int maxvalue = Integer.MIN_VALUE;
    for (IntWritable value: values) {
        maxvalue = Math.max(maxvalue, value.get());
    }
    context.write(key, new IntWritable(maxvalue));
}

//Main Function
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = Job.getInstance(conf, "weather example");
    job.setJarByClass(MaxTemp.class);
    job.setMapperClass(MaxTemperatureMapper.class);

    job.setReducerClass(MaxTemperatureReducer.class);

    job.setInputFormatClassTextInputFormat.class);

    job.setOutputFormatClass(TextOutputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
}

```

```

1 //Packages
2
3 import java.io.IOException;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.LongWritable;
6
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.io.IntWritable;
9 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
13
14 import org.apache.hadoop.mapreduce.Job;
15 import org.apache.hadoop.mapreduce.Mapper;
16 import org.apache.hadoop.mapreduce.Reducer;
17 import org.apache.hadoop.conf.Configuration;
18
19 //map logic
20 public static class MaxTemperatureMapper extends Mapper < LongWritable, Text, Text, IntWritable > {
21
22     public static final int MISSING = 9999;
23
24
25     public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
26         String line = value.toString();
27         String year = line.substring(15, 19);
28         int airtemp;
29         if (line.charAt(87) == '+') {
30             airtemp = Integer.parseInt(line.substring(88, 92));
31         } else
32             airtemp = Integer.parseInt(line.substring(87, 92));
33         String q = line.substring(92, 93);
34         if (airtemp != MISSING && q.matches("[01459]")) {
35             context.write(new Text(year), new IntWritable(airtemp));
36         }
37     }
38 }

```

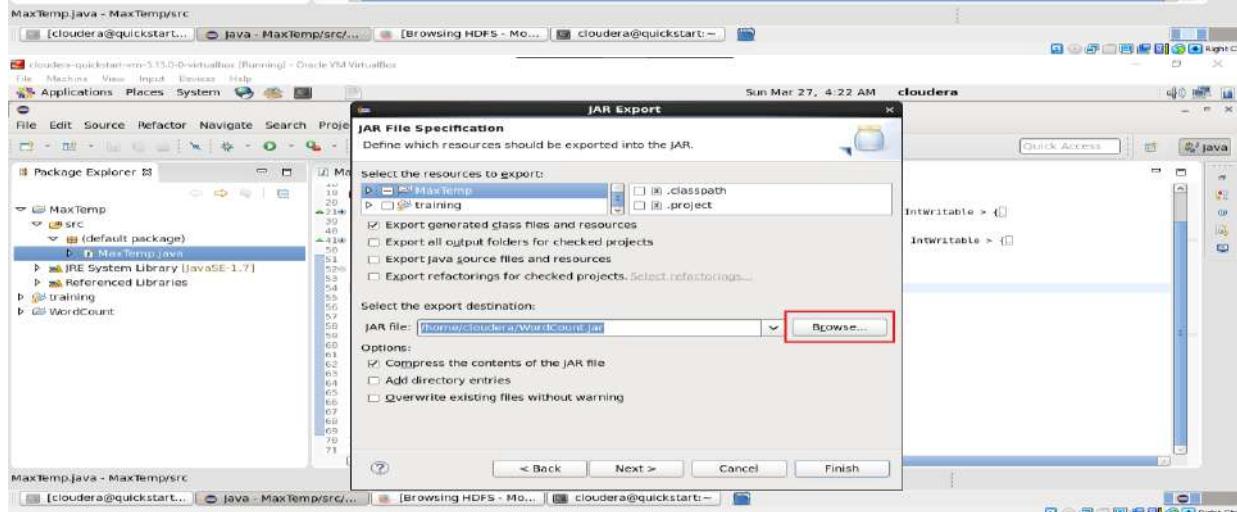
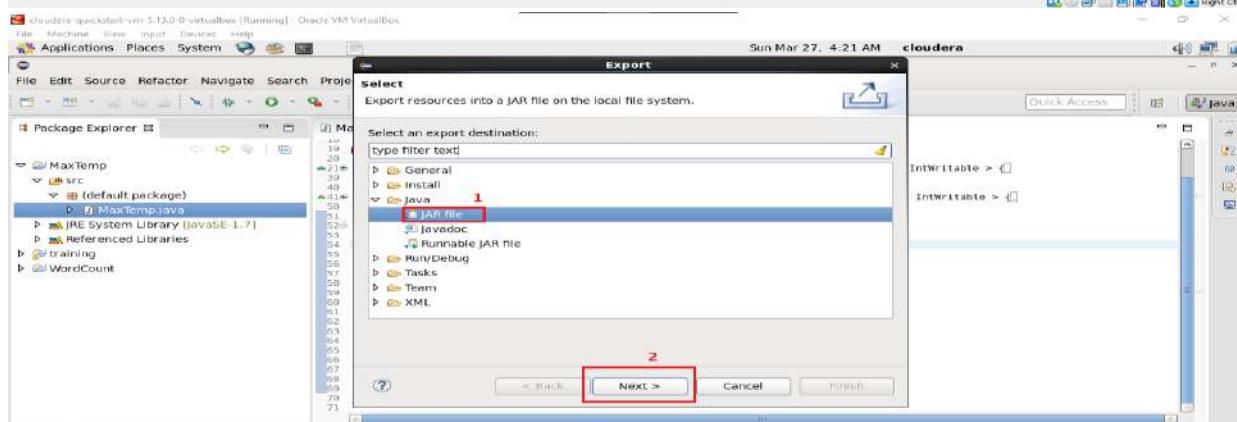
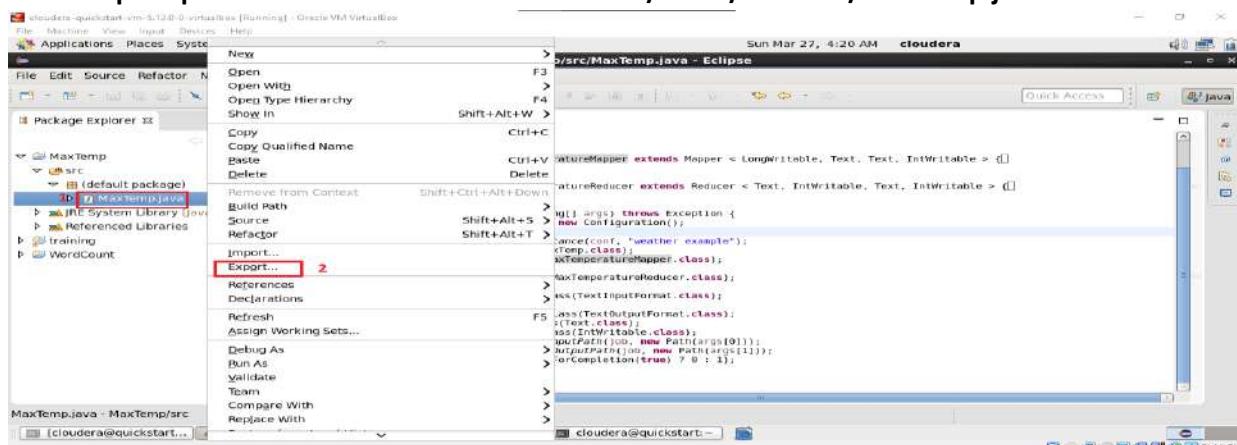
```

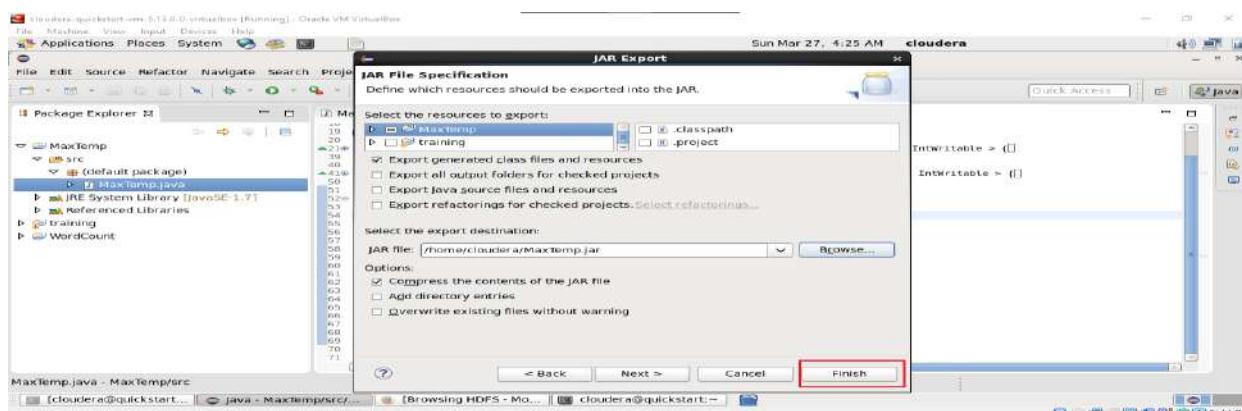
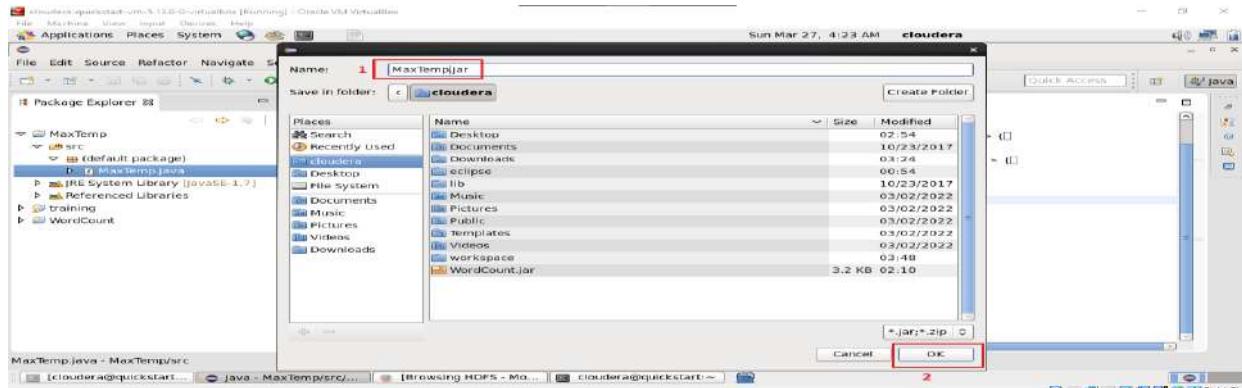
40 //Reducer
41 public static class MaxTemperatureReducer extends Reducer < Text, IntWritable, Text, IntWritable > {
42     public void reduce(Text key, Iterable < IntWritable > values, Context context) throws IOException, InterruptedException {
43         int maxvalue = Integer.MIN_VALUE;
44         for (IntWritable value: values) {
45             maxvalue = Math.max(maxvalue, value.get());
46         }
47         context.write(key, new IntWritable(maxvalue));
48     }
49 }
50
51 //Main Function
52 public static void main(String[] args) throws Exception {
53     Configuration conf = new Configuration();
54
55     Job job = Job.getInstance(conf, "weather example");
56     job.setJarByClass(MaxTemp.class);
57     job.setMapperClass(MaxTemperatureMapper.class);
58
59     job.setReducerClass(MaxTemperatureReducer.class);
60
61     job.setInputFormatClass(TextInputFormat.class);
62
63     job.setOutputFormatClass(TextOutputFormat.class);
64     job.setOutputKeyClass(Text.class);
65     job.setOutputValueClass(IntWritable.class);
66     TextInputFormat.addInputPath(job, new Path(args[0]));
67     FileOutputFormat.setOutputPath(job, new Path(args[1]));
68     System.exit(job.waitForCompletion(true) ? 0 : 1);
69 }

```

7. Right click on the project name

MaxTemp> Export > Java > JAR File > Next > Jar file “/home/cloudera/MaxTemp.jar”> Finish.





Verify jar file from terminal by using Open terminal & type "ls" There it will show MaxTemp.jar

Check current working directory

pwd

Check the list

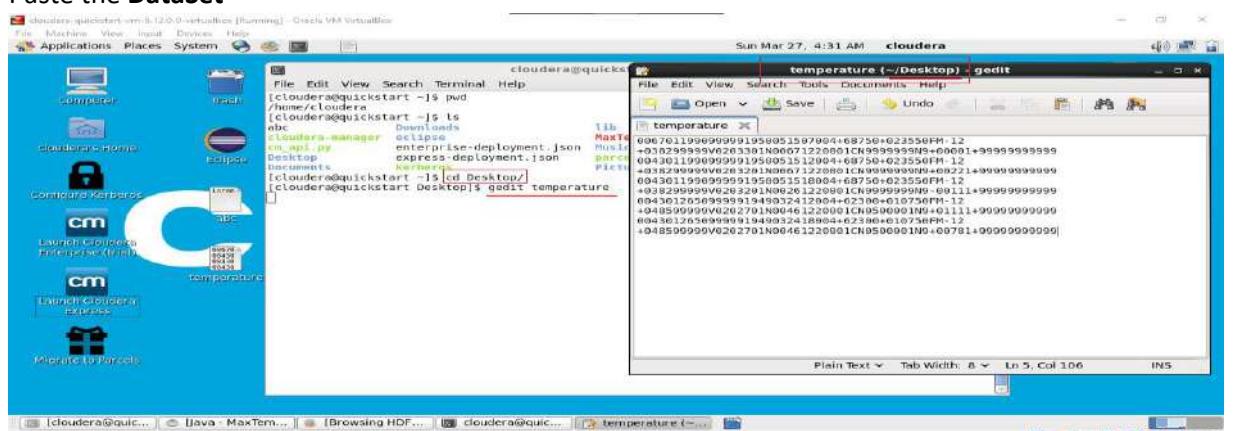
ls



8. We need to create an input file in local file system (On Desktop)

Creating an input file named as "temperature".

Paste the DataSet



Here listing all the directory present in

hdfs using hdfs dfs -ls /

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 11 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-27 02:30 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-27 02:55 /inputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:31 /op1
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:08 /outputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-18 05:36 /rjclocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2022-03-25 08:12 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$
```

9. Now we have to move this input file to hdfs. For this we create a direcory on hdfs using command

hdfs dfs -mkdir /tempip

```
[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /tempip
[cloudera@quickstart Desktop]$
```

If getting any permission related error than execute below command

export HADOOP_USER_NAME=hdfs

Then we can verify whether this directory is created or not using ls command

hdfs dfs -ls /

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 12 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2022-03-27 02:30 /hbase
drwxr-xr-x - hdfs supergroup 0 2022-03-27 02:55 /inputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:58 /newdir
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:31 /op1
drwxr-xr-x - hdfs supergroup 0 2022-03-27 03:08 /outputdir
drwxr-xr-x - hdfs supergroup 0 2022-03-18 05:36 /rjclocal
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxr-xr-x - hdfs supergroup 0 2022-03-27 04:37 /tempip
drwxrwxrwt - hdfs supergroup 0 2022-03-25 08:12 /tmp
drwxr-xr-x - hdfs supergroup 0 2022-03-17 07:20 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$
```

Move the input file to this directory created in hdfs by using either **put** command or **copyFromLocal** command

hdfs dfs -put /home/cloudera/Desktop/temperature /tempip/

```
[cloudera@quickstart Desktop]$ hdfs dfs -put /home/cloudera/Desktop/temperature /tempip/
[cloudera@quickstart Desktop]$
```

Now checking whether the “temperature” present in /tempip directory of hdfs or not using command

hdfs dfs -ls /tempip

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /tempip
Found 1 items
-rw-r--r-- 1 hdfs supergroup 530 2022-03-27 04:37 /tempip/temperature
[cloudera@quickstart Desktop]$
```

As we can see “temperature” file is present in / tempip directory of hdfs. Now we will see the content of this file using command

hdfs dfs -cat /tempip/temperature

```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /tempip/temperature
0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999
999
0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999
999
0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999
999
0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999
999
0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999
999
[cloudera@quickstart Desktop]$
```

10. Running Mapreduce Program on Hadoop,

Syntax : **hadoop jar jarFileName.jar ClassName /InputFileAddress /outputdir**

hadoop jar /home/cloudera/MaxTemp.jar MaxTemp /tempip/temperature /tempop1

```
[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/MaxTemp.jar MaxTemp /tempip/temperature /tempop
22/03/27 05:53:49 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
22/03/27 05:53:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
```

Map-Reduce Framework

The screenshot shows the Eclipse IDE interface. In the top right, a terminal window displays the execution of the MaxTemp Java program. The log output includes:

```
22/03/27 05:54:15 INFO mapreduce.JobSubmitter: Submitting tokens=1
22/03/27 05:54:17 INFO mapreduce.JobSubmitter: Submitting total 1 job(s) to ResourceManager
22/03/27 05:54:17 INFO mapreduce.JobSubmitter: number of splits:1
22/03/27 05:54:17 INFO mapreduce.JobSubmitter: Job tracking url: http://quickstart.cloudera:8088/proxy/application_1640373372244_0003
22/03/27 05:54:17 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1640373372244_0003
22/03/27 05:54:17 INFO mapreduce.Job: Counters:
FILE: Number of bytes read=62
FILE: Number of bytes written=294993
FILE: Number of records read=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1048576
HDFS: Number of bytes written=17
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
22/03/27 05:56:29 INFO mapreduce.Job: map 100% reduce 100%
22/03/27 05:56:29 INFO mapreduce.Job: Job finished: job_1640373372244_0003
22/03/27 05:56:29 INFO mapreduce.Job: Counters:
FILE: Number of bytes read=62
FILE: Number of bytes written=294993
FILE: Number of records read=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1048576
HDFS: Number of bytes written=17
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters:
FILE: Number of bytes read=62
FILE: Number of bytes written=294993
FILE: Number of records read=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1048576
HDFS: Number of bytes written=17
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
```

The screenshot shows a terminal window titled "cloudera@quickstart:~\$". It displays the command "java -Xms1024m -Xmx1024m MaxTemp src/main/resources/temperature.txt /tmp/temperature /tmp/tempop1" and its execution log:

```
Map output bytes=3
Map output serialized bytes=38
Input file bytes=1048576
Combine input records=0
Combiner input bytes=0
Reduce input groups=2
Reduce shuffle bytes=38
Reduce shuffle records=2
Reduce output bytes=17
Reduce output records=2
SGD: Number of iterations=9
Shuffled Maps =1
Failed Shuffles=0
Map output bytes=3
Map output serialized bytes=38
Input file bytes=1048576
Physical memory (bytes) snapshot=246063104
Virtual memory (bytes) snapshot=142995216
Total committed heap usage (bytes)=55944764
shuffle errors=0
CONNECTION=0
IO ERROR=0
WRONG LENGTH=0
WRONG MAP=0
WRONG REDUCE=0
File Input Format Counters
Bytes Read=30
File Output Format Counters
Bytes Written=17
```

11. Then we can verify the content of outputdir directory and in that part-r file has the actual output by using the command `Hdfs dfs -cat /outputdir/part-r-00000` This will give us final output. The same file can also be accessed using a browser. For every execution of this program we need to delete the output directory or give a new name to the output directory every time. 1st we are checking whether the outputdir directory is created in hdfs or not using command

hdfs dfs -ls /

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /
Found 13 items
drwxrwxrwx  - hdfs  supergroup          0 2017-10-23 09:15 /benchmarks
drwxr-xr-x  - hbase supergroup          0 2022-03-27 05:48 /hbase
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 02:55 /inputdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:58 /newdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 03:31 /op1
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 03:08 /outputdir
drwxr-xr-x  - hdfs  supergroup          0 2022-03-18 05:36 /rjclocal
drwxr-xr-x  - solr   supergroup          0 2017-10-23 09:18 /solr
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 04:37 /tempip
drwxr-xr-x  - hdfs  supergroup          0 2022-03-27 05:56 /tempop
drwxrwxrwx  - hdfs  supergroup          0 2022-03-25 08:12 /tmp
drwxr-xr-x  - hdfs  supergroup          0 2022-03-17 07:20 /user
drwxr-xr-x  - hdfs  supergroup          0 2017-10-23 09:17 /var
[cloudera@quickstart Desktop]$
```

Now let's check what we have inside this **tempop** directory using command as

hdfs dfs -ls /tempop

```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /tempop
Found 2 items
-rw-r--r--  1 hdfs supergroup          0 2022-03-27 05:56 /tempop/_SUCCESS
-rw-r--r--  1 hdfs supergroup        17 2022-03-27 05:56 /tempop/part-r-00000
[cloudera@quickstart Desktop]$
```

Now we want to read the content of the part-r-00000 file which present inside the **tempop** using command **hdfs dfs -cat /tempop/part-r-00000**

```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /tempop/part-r-00000
1949    111
1950    22
[cloudera@quickstart Desktop]$
```

So the maximum temperature for the year 1949 is 111 and for the year 1950 is 22.

12. The same file can also be accessed using a browser.

Browse the Directory by

Hadoop->HDFS Namenode->Utilities ->Browse the file system

cloudera@quickstart:~\$ hdfs dfs -ls /tmp

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxr-x	hdfs	supergroup	0 B	Mon Oct 23 09:15:43 -0700 2017	0	0 B	benchmarks
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 05:48:39 -0700 2022	0	0 B	hbase
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 02:55:49 -0700 2022	0	0 B	impala
-rwxrwxr-x	hdfs	supergroup	0 B	Thu Mar 17 07:50:45 -0700 2022	0	0 B	newfile
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 03:11:09 -0700 2022	0	0 B	np3
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 03:08:55 -0700 2022	0	0 B	output
-rwxrwxr-x	hdfs	supergroup	0 B	Fri Mar 10 05:36:11 -0700 2022	0	0 B	Reduced
-rwxrwxr-x	root	root	0 B	Mon Oct 23 09:16:03 -0700 2017	0	0 B	user
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 04:37:58 -0700 2022	0	0 B	temp
-rwxrwxr-x	hdfs	supergroup	0 B	Sun Mar 27 05:56:21 -0700 2022	0	0 B	tempop
-rwxrwxr-x	hdfs	supergroup	0 B	Fri Mar 25 08:12:06 -0700 2022	0	0 B	tmp
-rwxrwxr-x	hdfs	supergroup	0 B	Thu Mar 17 07:20:17 -0700 2022	0	0 B	user
-rwxrwxr-x	hdfs	supergroup	0 B	Mon Oct 23 09:17:24 -0700 2017	0	0 B	visit

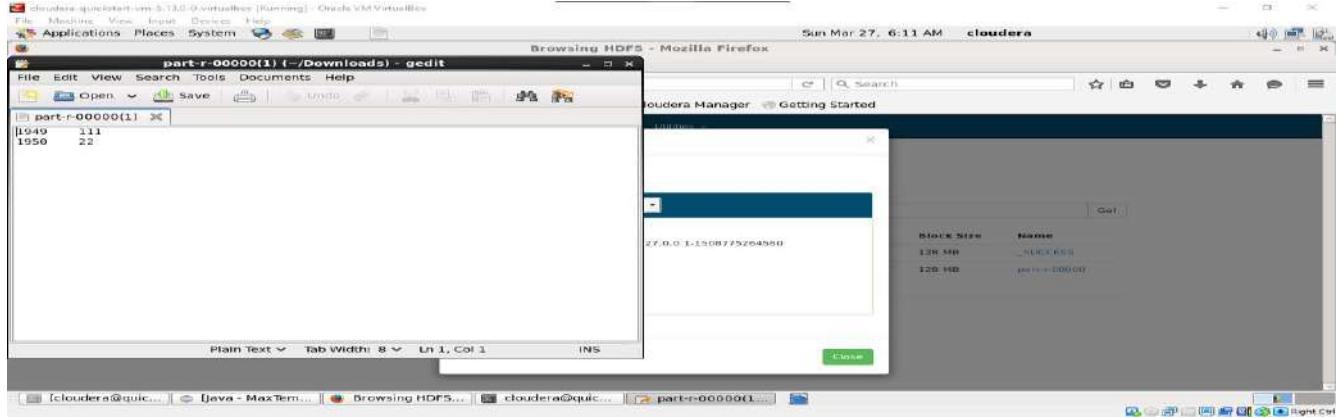
cloudera@quickstart:~\$ java -MaxTemp/src/... Browsing HDFS - Mozilla Firefox cloudera@quickstart:~\$

Now downloading the part-r-00000 file.

cloudera@quickstart:~\$ hdfs dfs -cat /tmp/part-r-00000

You have chosen to open:
part-r-00000
 which is: BIN file (17 bytes)
 from: http://quickstart.cloudera:50075
 Would you like to save this file?

Cancel Save File



For every execution of this program we need to delete the output directory or give a new name to the output directory every time.

Inside the part-r-00000 file it will have the same output as we are getting after executing using command.

hadoop jar /home/cloudera/MaximumTemp.jar MaxTemp /tempip/temperature /tempop1

Aim: A- Installing MongoDB

What is MongoDB?

MongoDB is a document-oriented **NoSQL database used for high volume data storage**. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.

Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

Download & Install MongoDB on Windows

- 1) Go to link (<https://www.mongodb.com/try/download/community>) and Download MongoDB Community Server. We will install the 64-bit version for Windows.

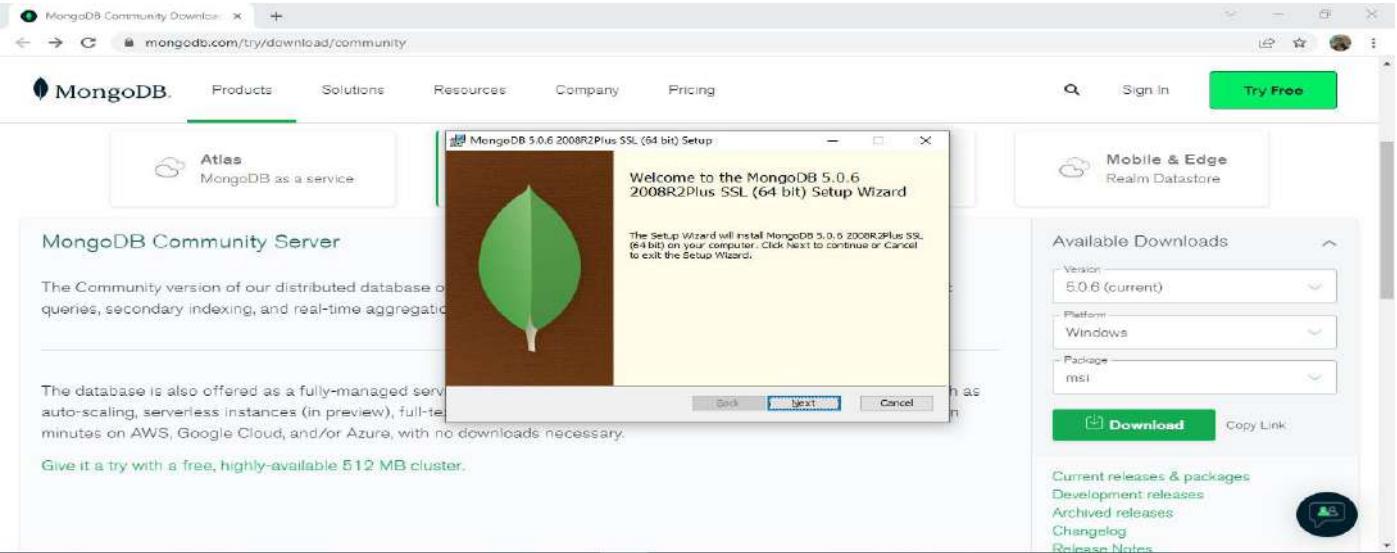
The screenshot shows the MongoDB Community Downloads page. At the top, there are four main categories: Atlas (MongoDB as a service), On-premises (MongoDB locally), Tools (Boost productivity), and Mobile & Edge (Realm Datastore). Below these, there are four dropdown menus for different MongoDB products: MongoDB Community Server, MongoDB Ops Manager, MongoDB Enterprise Kubernetes Operator, and MongoDB Community Kubernetes Operator. The 'MongoDB Community Server' dropdown is currently selected. At the bottom of the page, there is a progress bar indicating 'Waiting for bam.nr-data.net...'.

This screenshot shows a more detailed view of the MongoDB Community Downloads page. It features a grid of products under the 'Community Edition' category. The products listed are: Database, Enterprise Server, Community Server, Edge-to-Cloud Sync; Search, Ops Manager, Cloud Manager, Functions, APIs, and more; Data Lake, Enterprise Kubernetes Operator, Community Kubernetes Operator; and Charts. To the right of the grid, there is a 'Tools' section with options like Compass, Shell, VS Code Plugin, and Atlas CLI. At the bottom of the page, there is a message about MongoDB Atlas and a download button for a 'mongod.msi' file.

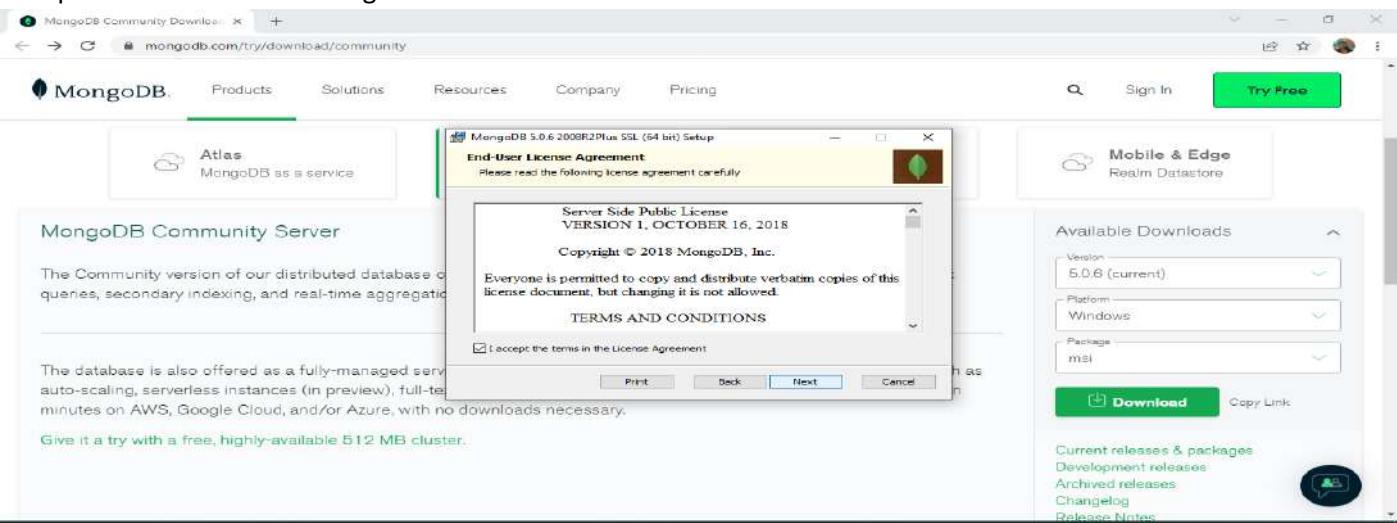
The screenshot displays the MongoDB Community Downloads page again, focusing on the 'MongoDB Community Server' section. It provides a brief description of the community version's features and a note about MongoDB Atlas. Below this, there is a 'Available Downloads' section where users can select the 'Version' (5.0.0), 'Platform' (Windows), and 'Package' (msi). A large 'Download' button is prominently displayed. At the very bottom of the page, there is a file download window for 'mongodb-windows-5.0.0.msi'.

- 2) Once download is complete open the msi file. Click Next in the start-up screen

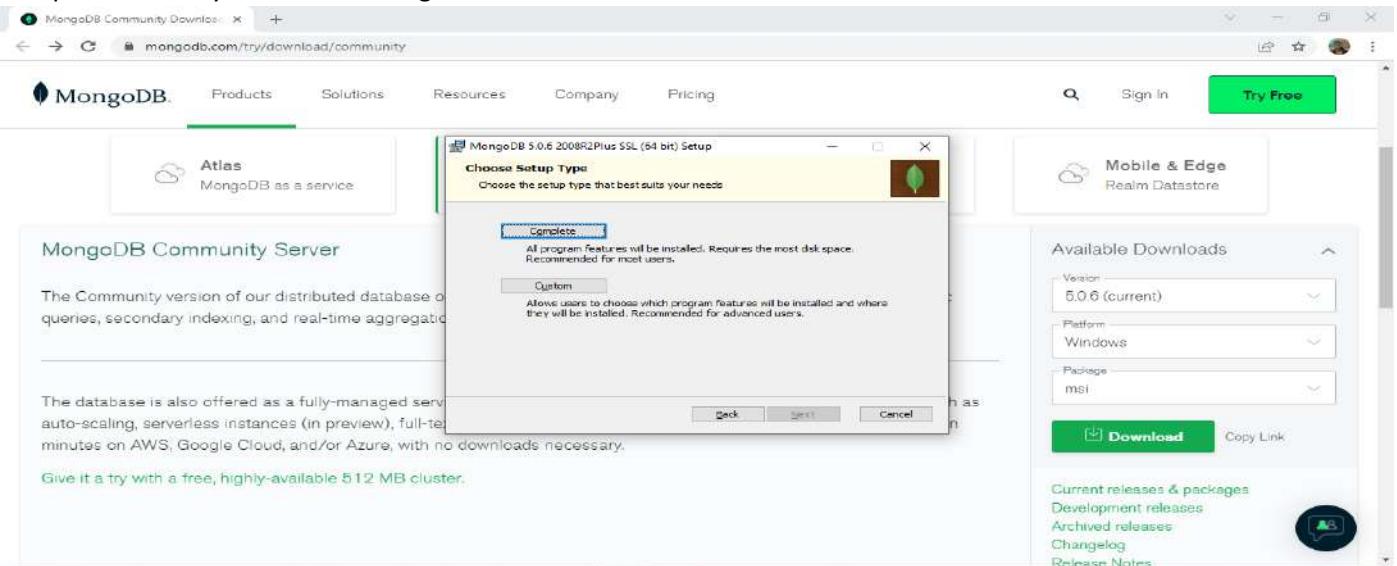
Name: Pavan Yadav



3) Accept the End-User License Agreement and Click Next



4) Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.



5) Select “Run service as Network Service user”. Make a note of the data directory, we’ll need this later and Click Next.

The screenshot shows the MongoDB Community Server download page. In the center, a 'Service Configuration' dialog box is open, prompting the user to specify optional settings for MongoDB as a service. It includes fields for 'Account Domain', 'Account Name' (set to 'MongoDB'), 'Account Password', 'Service Name' (set to 'MongoDB'), 'Data Directory' (set to 'C:\Program Files\MongoDB\Server\5.0\data'), and 'Log Directory' (set to 'C:\Program Files\MongoDB\Server\5.0\log'). Below the dialog, a note states: 'The database is also offered as a fully-managed service, auto-scaling, serverless instances (in preview), full-time active failover, minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.' At the bottom, a green link says 'Give it a try with a free, highly-available 512 MB cluster.'

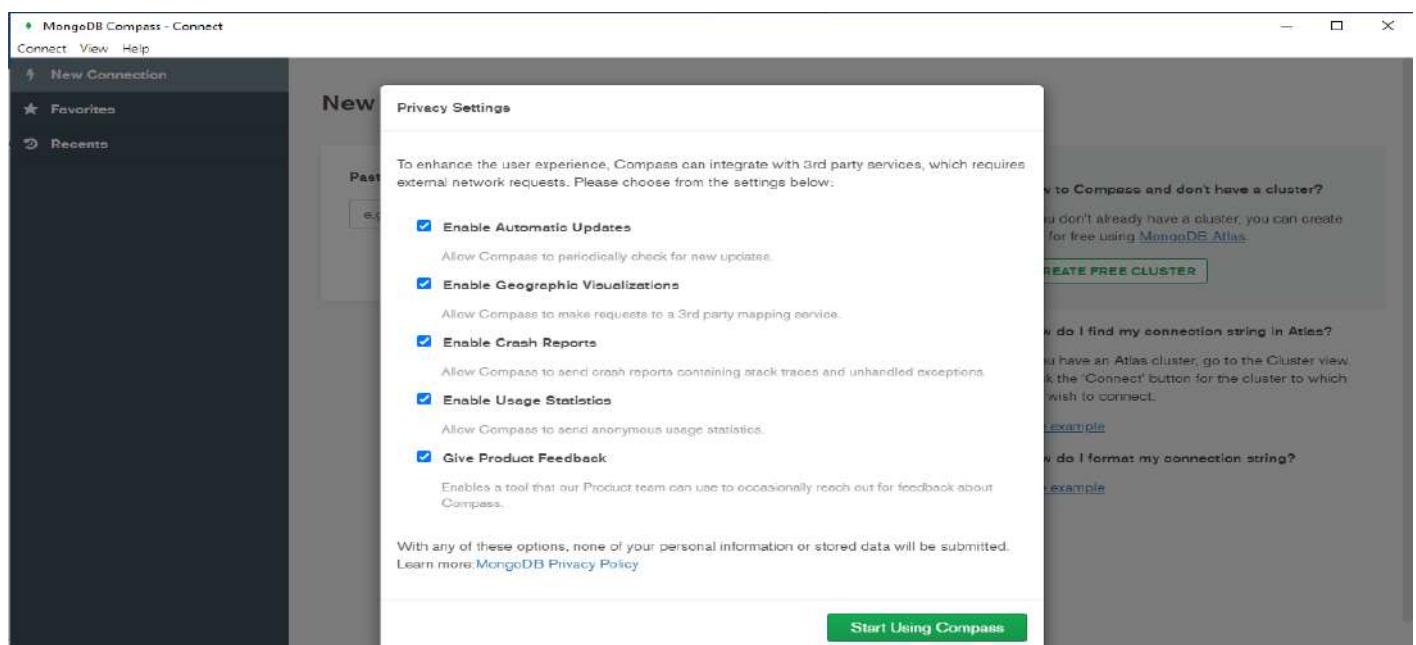
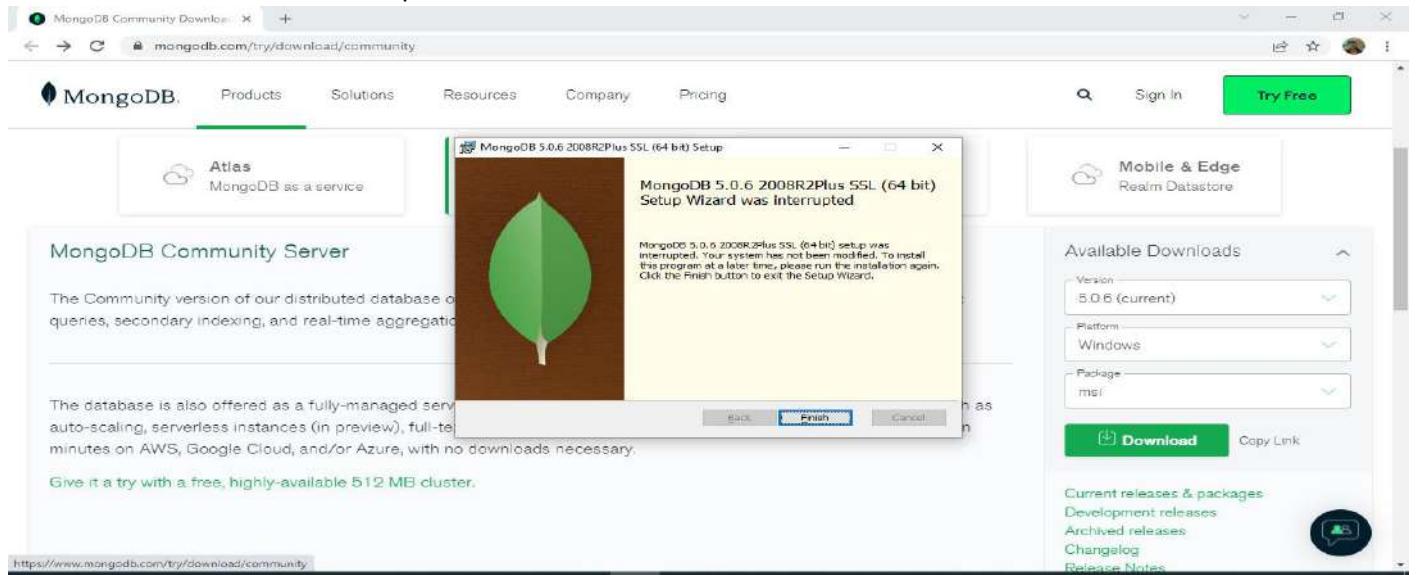
6) Click on the Install button to start the installation.

This screenshot shows the same MongoDB Community Server download page as above, but the 'Install MongoDB Compass' dialog box is now open. It displays a note about MongoDB Compass being the official graphical user interface for MongoDB. A 'Next >' button is visible at the bottom of the dialog.

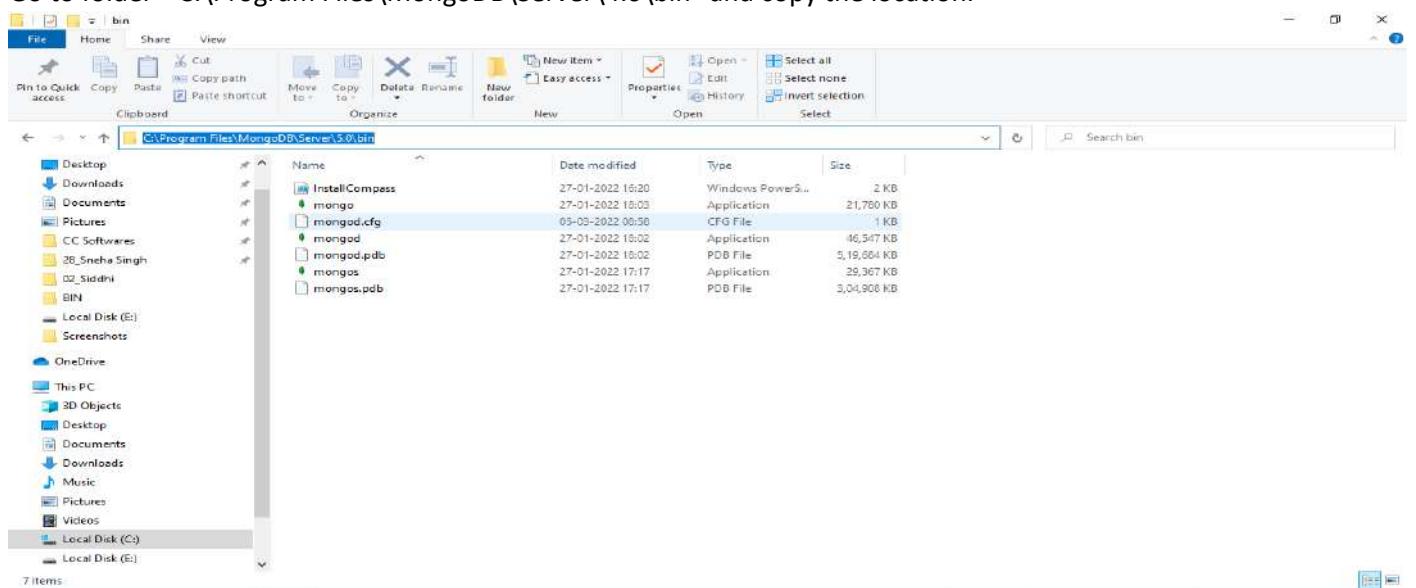
The screenshot shows the 'Ready to Install MongoDB 5.0.6 2008R2Plus SSL (64 bit)' dialog box. It contains a note: 'Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.' A large 'Install' button is prominently displayed in the center.

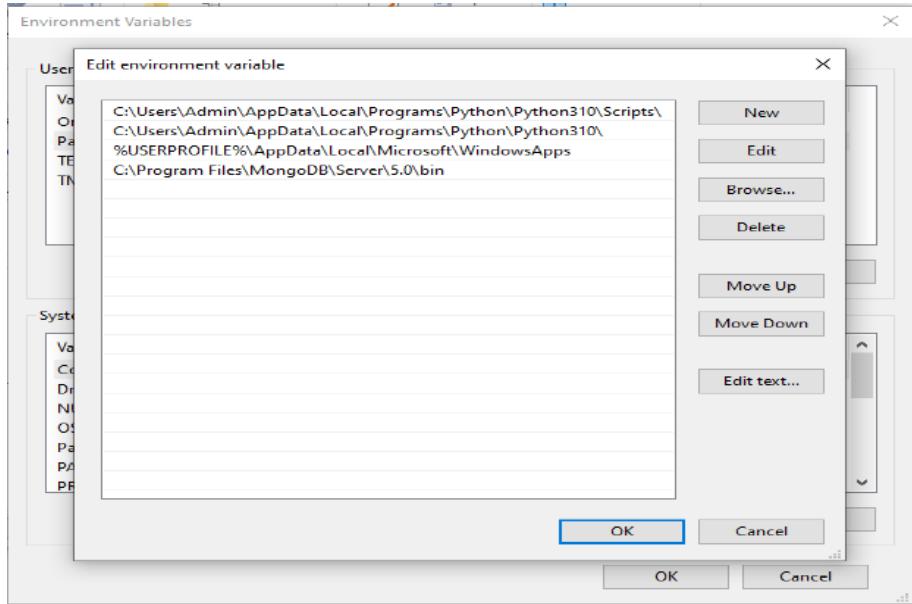
The final screenshot shows the 'Installing MongoDB 5.0.6 2008R2Plus SSL (64 bit)' progress dialog box. It displays a status message: 'Please wait while the Setup Wizard installs MongoDB 5.0.6 2008R2Plus SSL (64 bit).'. A 'Status:' progress bar is shown, and a 'Cancel' button is at the bottom right.

7) Click on the Finish button to complete the installation



8) Go to folder " C:\Program Files\MongoDB\Server\4.0\bin" and copy the location.





- 9) Open the command prompt and run the cd command to change directory and run the command "mongod" to start the server and run the command "mongo" to work mongoDB.

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.18363.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongod
{"t": {"$date": "2022-03-05T09:03:29.600+05:30"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "-", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t": {"$date": "2022-03-05T09:03:29.601+05:30"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "main", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}}, "isInternalClient": true}}
{"t": {"$date": "2022-03-05T09:03:29.607+05:30"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-03-05T09:03:29.608+05:30"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "main", "msg": "Implicit TCP FastOpen in use."}
{"t": {"$date": "2022-03-05T09:03:29.611+05:30"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"}
{"t": {"$date": "2022-03-05T09:03:29.611+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "ns": "config.tenantMigrationDonors"}}
{"t": {"$date": "2022-03-05T09:03:29.612+05:30"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientsService", "ns": "config.tenantMigrationRecipients"}}
{"t": {"$date": "2022-03-05T09:03:29.615+05:30"}, "s": "I", "c": "CONTROL", "id": 5045603, "ctx": "main", "msg": "Multi threading initialized"}
{"t": {"$date": "2022-03-05T09:03:29.618+05:30"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 3324, "port": 27017, "dbPath": "C:/data/db/", "architecture": "64-bit", "host": "PC-92"}}
{"t": {"$date": "2022-03-05T09:03:29.619+05:30"}, "s": "I", "c": "CONTROL", "id": 23398, "ctx": "initandlisten", "msg": "Target operating system minimum version", "attr": {"targetMinOS": "Windows 7/Windows Server 2008 R2"}}
{"t": {"$date": "2022-03-05T09:03:29.619+05:30"}, "s": "I", "c": "CONTROL", "id": 23403, "ctx": "initandlisten", "msg": "Build Info", "attr": {"buildInfo": {"version": "5.0.6", "gitVersion": "212a8dbb47f07427daef1949c75baec1d81d9259", "modules": [], "allocator": "tcmalloc", "environment": {"distmod": "windows", "distarch": "x86_64", "target_arch": "x86_64"}}}
{"t": {"$date": "2022-03-05T09:03:29.619+05:30"}, "s": "I", "c": "CONTROL", "id": 51765, "ctx": "initandlisten", "msg": "Operating System", "attr": {"os": {"name": "Microsoft Windows 10", "version": "10.0 (build 18363)"}}, "attr": {"options": {}}}
{"t": {"$date": "2022-03-05T09:03:29.619+05:30"}, "s": "I", "c": "CONTROL", "id": 21951, "ctx": "initandlisten", "msg": "Options set by command line", "attr": {"options": {}}}
{"t": {"$date": "2022-03-05T09:03:29.622+05:30"}, "s": "E", "c": "CONTROL", "id": 20557, "ctx": "initandlisten", "msg": "DBException in initAndListen, terminating", "attr": {"error": "NonExistentPath: Data directory C:\\data\\db\\ not found. Create the missing directory or specify another path using (1) the --dbpath command line option, or (2) by adding the 'storage.dbPath' option in the configuration file."}}
{"t": {"$date": "2022-03-05T09:03:29.623+05:30"}, "s": "I", "c": "REPL", "id": 4784900, "ctx": "initandlisten", "msg": "Stepping down the ReplicationCoordinator for shutdown", "attr": {"waitTimeMillis": 15000}}
{"t": {"$date": "2022-03-05T09:03:29.625+05:30"}, "s": "", "c": "COMMAND", "id": 4784901, "ctx": "initandlisten", "msg": "Shutting down the MirrorMaestro"}
{"t": {"$date": "2022-03-05T09:03:29.628+05:30"}, "s": "I", "c": "SHARDING", "id": 4784902, "ctx": "initandlisten", "msg": "Shutting down the WaitForMajorityService"}
{"t": {"$date": "2022-03-05T09:03:29.628+05:30"}, "s": "I", "c": "NETWORK", "id": 20562, "ctx": "initandlisten", "msg": "Shutdown: going to close listening sockets"}
{"t": {"$date": "2022-03-05T09:03:29.630+05:30"}, "s": "I", "c": "NETWORK", "id": 4784905, "ctx": "initandlisten", "msg": "Shutting down the global connection pool"}
{"t": {"$date": "2022-03-05T09:03:29.630+05:30"}, "s": "I", "c": "CONTROL", "id": 4784906, "ctx": "initandlisten", "msg": "Shutting down the FlowControlTicketholder"}
{"t": {"$date": "2022-03-05T09:03:29.631+05:30"}, "s": "I", "c": "-", "id": 20520, "ctx": "initandlisten", "msg": "Stopping further Flow Control ticket acquisitions."}

{"t": {"$date": "2022-03-05T09:03:29.631+05:30"}, "s": "", "c": "NETWORK", "id": 4784918, "ctx": "initandlisten", "msg": "Shutting down the ReplicaSetMonitor"}
{"t": {"$date": "2022-03-05T09:03:29.632+05:30"}, "s": "I", "c": "SHARDING", "id": 4784921, "ctx": "initandlisten", "msg": "Shutting down the MigrationUtilExecutor"}
{"t": {"$date": "2022-03-05T09:03:29.633+05:30"}, "s": "I", "c": "ASIO", "id": 22582, "ctx": "MigrationUtil-TaskExecutor", "msg": "Killing all outstanding egress activity"}
```

Aim: B- Performing CRUD Operations for unstructured data

For storing data in a MongoDB, you need to create a database first. It will allow you to systematically organize your data so that it can be retrieved as per requirement. If you wish to delete a database, MongoDB also allows you to delete that.

What is CRUD in MongoDB?

CRUD operations describe the conventions of a user-interface that let users view, search, and modify parts of the database.

MongoDB documents are modified by connecting to a server, querying the proper documents, and then changing the setting properties before sending the data back to the database to be updated.

CRUD is data-oriented, and it's standardized according to HTTP action verbs.

When it comes to the individual CRUD operations:

1. The Create operation is used to insert new documents in the MongoDB database.
2. The Read operation is used to query a document in the database.
3. The Update operation is used to modify existing documents in the database.
4. The Delete operation is used to remove documents in the database.

Creating a Database in MongoDB

MongoDB has no "create" command for creating a database. Also, it is essential to note that MongoDB does not provide any specific command for creating a database. This might seem a bit agitated if you are new to this subject and database tool or in case you have used that conventional SQL as your database where you are required to create a new database, which will contain table and, you will then have to use the INSERT INTO TABLE to insert values manually within your table.

In this MongoDB tool, you need not have to produce, or it is optional to create a database manually. This is because MongoDB has the feature of automatically creating it for the first time for you, once you save your value in that collection. So, explicitly, you do not need to mention or put a command to create a database; instead it will be created automatically once the collection is filled with values.

The "use" Command for Creating Database in MongoDB

You can make use of the "use" command followed by the database_name for creating a database. This command will tell the MongoDB client to create a database by this name if there is no database exists by this name. Otherwise, this command will return the existing database that has the name.

Show list of databases

First start the mongo Db using command **mongo**

1) show dbs;

```
Command Prompt - mongo
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
-----
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
    https://community.mongodb.com
-----
The server generated these startup warnings when booting:
2022-03-05T08:58:27.721+05:30: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
> show dbs;
admin      0.000GB
config     0.000GB
local      0.000GB
>
```

2) use RetailBikeDb

```
Command Prompt - mongo
> show dbs;
admin      0.000GB
config     0.000GB
local      0.000GB
> use RetailBikeDb
switched to db RetailBikeDb
> -
```

3) Creating and showing collections in MongoDB

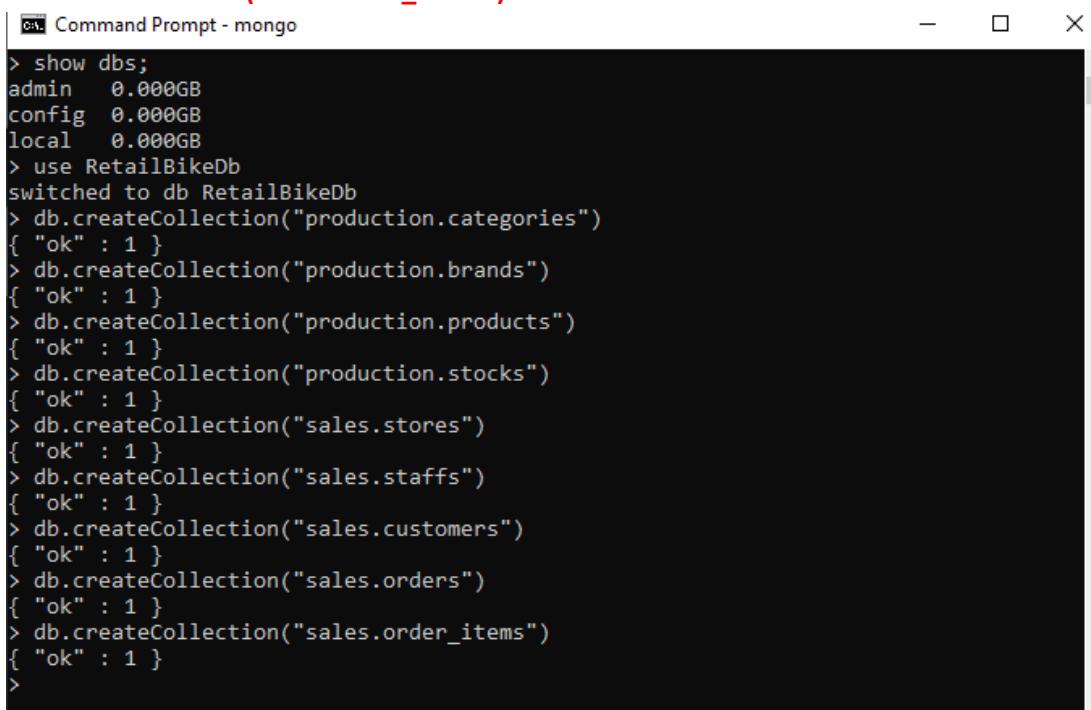
Collections are like that of tables of RDBMS and are capable enough to store documents of diverse or dissimilar types.

Creation and removal of collections in MongoDB can be done in specific ways. In this chapter, you will learn about the creation of collections in a database created using MongoDB.

Creation of collection can be done using

Syntax: db.createCollection(name)

```
db.createCollection("production.categories")
db.createCollection("production.brands")
db.createCollection("production.products")
db.createCollection("production.stocks")
db.createCollection("sales.stores")
db.createCollection("sales.staffs")
db.createCollection("sales.customers")
db.createCollection("sales.orders")
db.createCollection("sales.order_items")
```



The screenshot shows a terminal window titled "Command Prompt - mongo". The command line interface (CLI) is used to execute MongoDB commands. The user has switched to the "RetailBikeDb" database and then created nine collections: "production.categories", "production.brands", "production.products", "production.stocks", "sales.stores", "sales.staffs", "sales.customers", "sales.orders", and "sales.order_items". Each collection creation command is followed by a response object indicating success with an "ok" field set to 1.

```
> show dbs;
admin 0.000GB
config 0.000GB
local 0.000GB
> use RetailBikeDb
switched to db RetailBikeDb
> db.createCollection("production.categories")
{ "ok" : 1 }
> db.createCollection("production.brands")
{ "ok" : 1 }
> db.createCollection("production.products")
{ "ok" : 1 }
> db.createCollection("production.stocks")
{ "ok" : 1 }
> db.createCollection("sales.stores")
{ "ok" : 1 }
> db.createCollection("sales.staffs")
{ "ok" : 1 }
> db.createCollection("sales.customers")
{ "ok" : 1 }
> db.createCollection("sales.orders")
{ "ok" : 1 }
> db.createCollection("sales.order_items")
{ "ok" : 1 }
>
```

4) We can show list of collection using "Show collections" commands in MongoDB.

show collections

```
> show collections
production.brands
production.categories
production.products
production.stocks
sales.customers
sales.order_items
sales.orders
sales.staffs
sales.stores
>
```

5) CRUD operation

CRUD operation is one of the essential concepts of a database system. Inserting data in the database comes under one of the CRUD operations. If you do not insert data in your database, you will not be able to continue with other activities within your document.

1) Create Operations

For MongoDB CRUD, if the specified collection doesn't exist, the create operation will create the collection when it's executed. Create operations in MongoDB target a single collection, not multiple collections. Insert operations in MongoDB are atomic on a single document level.

MongoDB provides two different create operations that you can use to insert documents into a collection:

1. **db.collection.insertOne()**
2. **db.collection.insertMany()**

The insertOne() method

As the namesake, insertOne() allows you to insert one document into the collection. Example -

```
db.movie.insertOne({ _id: 2, writername: "Stan Lee", name: "Aquaman" })
```

The insertMany() method

It's possible to insert multiple items at one time by calling the insertMany() method on the desired collection. In this case, we pass multiple items into our chosen collection (RetailDB) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

I have taken example of Retail Bike store database to demonstrate insert crud operations.

1. **production.categories**
db.production.categories.insertMany(
[
{
"category_id": 1,
"category_name": "Road Bike"
},
{
"category_id": 2,
"category_name": "Mountain Bike"
},
{
"category_id": 3,
"category_name": "Hybrid Bike"
},
{
"category_id": 4,
"category_name": "Folding Bike"
},
{
"category_id": 5,
"category_name": "Touring Bike"
},
{
"category_id": 6,
"category_name": "Cruiser Bike"
},
{
"category_id": 7,
"category_name": "Women Bike"
}
]

```
Command Prompt - mongo
sales.stores
> db.production.categories.insertMany(
... [
... {
... "category_id": 1,
... "category_name": "Road Bike"
... },
... {
... "category_id": 2,
... "category_name": "Mountain Bike"
... },
... {
... "category_id": 3,
... "category_name": "Hybrid Bike"
... },
... {
... "category_id": 4,
... "category_name": "Folding Bike"
... },
... {
... "category_id": 5,
... "category_name": "Touring Bike"
... },
... {
... "category_id": 6,
... "category_name": "Cruiser Bike"
... },
... {
... "category_id": 7,
... "category_name": "Women Bike"
... }
... ]
... )
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("6222df12b5e511acae1d7891"),
        ObjectId("6222df12b5e511acae1d7892"),
        ObjectId("6222df12b5e511acae1d7893"),
        ObjectId("6222df12b5e511acae1d7894"),
        ObjectId("6222df12b5e511acae1d7895"),
        ObjectId("6222df12b5e511acae1d7896"),
        ObjectId("6222df12b5e511acae1d7897")
    ]
}
```

2. production.products

```
db.production.products.insertMany(
[
{
"product_id": 1,
"product_name": "Honda Superfast",
"brand_id": 1,
"category_id": 1,
"model_year": 1994,
"list_price": 25000
},
{
"product_id": 2,
"product_name": "6KU Bikes",
"brand_id": 2,
"category_id": 4,
"model_year": 2000,
"list_price": 30000
},
{
"product_id": 3,
"product_name": "Bianchi",
"brand_id": 3,
"category_id": 2,
"model_year": 2002,
"list_price": 30000
},
{
}
```

```
"product_id": 4,  
"product_name": "BMC Hybrid Bike",  
"brand_id": 4,  
"category_id": 3,  
"model_year": 2009,  
  
"list_price": 45000  
},  
{  
"product_id": 5,  
"product_name": "Huffy Women Bike",  
"brand_id": 5,  
"category_id": 7,  
"model_year": 2019,  
"list_price": 50000  
}  
]  
)
```

```
 Command Prompt - mongo  
... "product_name": "6KU Bikes",  
... "brand_id": 2,  
... "category_id": 4,  
... "model_year": 2000,  
... "list_price": 30000  
},  
{  
... "product_id": 3,  
... "product_name": "Bianchi",  
... "brand_id": 3,  
... "category_id": 2,  
... "model_year": 2002,  
... "list_price": 30000  
},  
{  
... "product_id": 4,  
... "product_name": "BMC Hybrid Bike",  
... "brand_id": 4,  
... "category_id": 3,  
... "model_year": 2009,  
... "list_price": 45000  
},  
{  
... "product_id": 5,  
... "product_name": "Huffy Women Bike",  
... "brand_id": 5,  
... "category_id": 7,  
... "model_year": 2019,  
... "list_price": 50000  
}  
]  
)  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222df37b5e511acae1d7898"),  
        ObjectId("6222df37b5e511acae1d7899"),  
        ObjectId("6222df37b5e511acae1d789a"),  
        ObjectId("6222df37b5e511acae1d789b"),  
        ObjectId("6222df37b5e511acae1d789c")  
    ]  
}
```

3. production.stocks

```
db.production.stocks.insertMany(  
[  
{  
    "store_id": 1,  
    "product_id": 1,  
    "quantity": 15  
},  
{  
    "store_id": 2,  
    "product_id": 4,  
    "quantity": 20  
},  
{  
    "store_id": 3,  
    "product_id": 5,  
    "quantity": 30  
},  
{  
    "store_id": 1,  
    "product_id": 5,  
    "quantity": 100  
},  
{  
    "store_id": 2,  
    "product_id": 2,  
    "quantity": 4  
},  
{  
    "store_id": 3,  
    "product_id": 3,  
    "quantity": 9  
}  
]  
)
```

```
... {  
...     "store_id": 1,  
...     "product_id": 1,  
...     "quantity": 15  
... },  
... {  
...     "store_id": 2,  
...     "product_id": 4,  
...     "quantity": 20  
... },  
... {  
...     "store_id": 3,  
...     "product_id": 5,  
...     "quantity": 30  
... },  
... {  
...     "store_id": 1,  
...     "product_id": 5,  
...     "quantity": 100  
... },  
... {  
...     "store_id": 2,  
...     "product_id": 2,  
...     "quantity": 4  
... },  
... {  
...     "store_id": 3,  
...     "product_id": 3,  
...     "quantity": 9  
... }  
... ]  
... )  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222dfd4b5e511acae1d789d"),  
        ObjectId("6222dfd4b5e511acae1d789e"),  
        ObjectId("6222dfd4b5e511acae1d789f"),  
        ObjectId("6222dfd4b5e511acae1d78a0"),  
        ObjectId("6222dfd4b5e511acae1d78a1"),  
        ObjectId("6222dfd4b5e511acae1d78a2")  
    ]  
}
```

4. sales.customers

```
db.sales.customers.insertMany(  
[  
{  
"customer_id": "Cus001",  
"first_name": "Jay",  
"last_name": "Mehta",  
"phone": 1234567890,  
"email": "jay@gmail.com",  
"street": "T.P Road",  
"city": "Mumbai",  
"state": "Maharashtra",  
"zip_code": 400072  
},  
{  
"customer_id": "Cus002",  
"first_name": "Ruhি",  
"last_name": "Singh",  
"phone": 7894561230,  
"email": "ruhi@yahoo.com",  
"street": "M.G Chauk",  
"city": "Mumbai",  
"state": "Maharashtra",  
"zip_code": 400072  
},  
{  
"customer_id": "Cus003",  
"first_name": "Aria",  
"last_name": "Josh",  
"phone": 1245789630,  
"email": "aria.josh@gmail.com",  
"street": "JVM",  
"city": "Gandhi Nagar",  
  
"state": "Gujrat",  
"zip_code": 401235  
},  
{  
"customer_id": "Cus004",  
"first_name": "Mahi",  
"last_name": "Kaur",  
"phone": 4567890123,  
"email": "kaur.mahi@hotmail.com",  
"street": "J.V.L.R",  
"city": "Mumbai",  
"state": "Maharashtra",  
"zip_code": 400072  
},  
{  
"customer_id": "Cus005",  
"first_name": "Aditya",  
"last_name": "Yadav",  
"phone": 9638527410,  
"email": "aditya@gmail.com",  
"street": "Koliwada",  
}]
```

```
        "city": "Pune",
        "state": "Maharashtra",
        "zip_code": 300075
    }
]
)
```

The screenshot shows a terminal window titled "Select Command Prompt - mongo". The output displays the results of an insertMany operation into a collection. The data inserted consists of five documents, each representing a customer with fields like last_name, first_name, email, phone, street, city, state, and zip_code. Each document also includes an "_id" field represented as an ObjectId. The final part of the output shows the response from the database, indicating successful insertion with acknowledged=true and a list of insertedIds.

```
... "last_name": "Josh",
... "phone": 1245789630,
... "email": "aria.josh@gmail.com",
... "street": "JVM",
... "city": "Gandhi Nagar",
...
... "state": "Gujrat",
... "zip_code": 401235
},
{
...
"customer_id": "Cus004",
"first_name": "Mahi",
"last_name": "Kaur",
"phone": 4567890123,
"email": "kaur.mahi@hotmail.com",
"street": "J.V.L.R",
"city": "Mumbai",
"state": "Maharashtra",
"zip_code": 400072
},
{
...
"customer_id": "Cus005",
"first_name": "Aditya",
"last_name": "Yadav",
"phone": 9638527410,
"email": "aditya@gmail.com",
"street": "Koliwada",
"city": "Pune",
"state": "Maharashtra",
"zip_code": 300075
}
]
)
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("6222e05db5e511acae1d78a3"),
        ObjectId("6222e05db5e511acae1d78a4"),
        ObjectId("6222e05db5e511acae1d78a5"),
        ObjectId("6222e05db5e511acae1d78a6"),
        ObjectId("6222e05db5e511acae1d78a7")
    ]
}
>
```

5. sales.order_items

```
db.sales.order_items.insertMany(
[
{
    "order_id": "ORD001",
    "product_id": 1,
    "quantity": 2,
    "list_price": 50000
},
{
    "order_id": "ORD002",
    "product_id": 2,
    "quantity": 3,
    "list_price": 90000
},
{
    "order_id": "ORD003",
    "product_id": 3,
```

```
        "quantity": 1,  
        "list_price": 30000  
    },  
    {  
        "order_id": "ORD004",  
        "product_id": 4,  
        "quantity": 8,  
        "list_price": 360000  
    },  
    {  
        "order_id": "ORD005",  
        "product_id": 5,  
        "quantity": 2,  
        "list_price": 100000  
    }  
  
]  
)
```

```
0: Command Prompt - mongo  
... {  
...     "order_id": "ORD001",  
...     "product_id": 1,  
...     "quantity": 2,  
...     "list_price": 50000  
... },  
... {  
...     "order_id": "ORD002",  
...     "product_id": 2,  
...     "quantity": 3,  
...     "list_price": 90000  
... },  
... {  
...     "order_id": "ORD003",  
...     "product_id": 3,  
...     "quantity": 1,  
...     "list_price": 30000  
... },  
... {  
...     "order_id": "ORD004",  
...     "product_id": 4,  
...     "quantity": 8,  
...     "list_price": 360000  
... },  
... {  
...     "order_id": "ORD005",  
...     "product_id": 5,  
...     "quantity": 2,  
...     "list_price": 100000  
... }  
... ]  
... )  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222e0bab5e511acae1d78a8"),  
        ObjectId("6222e0bab5e511acae1d78a9"),  
        ObjectId("6222e0bab5e511acae1d78aa"),  
        ObjectId("6222e0bab5e511acae1d78ab"),  
        ObjectId("6222e0bab5e511acae1d78ac")  
    ]  
}  
>
```

6. sales.orders

```
db.sales.orders.insertMany(  
[  
  {  
    "order_id": "ORD001",  
    "customer_id": "Cus001",  
    "order_status": "Completed",  
    "order_date": 43992,  
    "shipped_date": 43994,  
    "store_id": 1,  
    "staff_id": 1  
  },  
  {  
    "order_id": "ORD002",  
    "customer_id": "Cus002",  
    "order_status": "Completed",  
    "order_date": 44221,  
    "shipped_date": 44227,  
    "store_id": 2,  
    "staff_id": 2  
  },  
  {  
    "order_id": "ORD003",  
    "customer_id": "Cus003",  
    "order_status": "Completed",  
    "order_date": 44306,  
    "shipped_date": 44314,  
    "store_id": 2,  
    "staff_id": 2  
  },  
  {  
    "order_id": "ORD004",  
    "customer_id": "Cus004",  
    "order_status": "Pending",  
    "order_date": 44367,  
    "shipped_date": 44377,  
    "store_id": 3,  
    "staff_id": 3  
  },  
  {  
    "order_id": "ORD005",  
    "customer_id": "Cus005",  
    "order_status": "Pending",  
    "order_date": 44367,  
    "shipped_date": 44377,  
    "store_id": 1,  
    "staff_id": 1  
  }  
]
```

7. sales.staffs

```
db.sales.staffs.insertMany(  
[  
  {  
    "staff_id": 1,  
    "first_name": "Pushpa",  
    "last_name": "Yadav",  
    "email": "pushpa@gmail.com",  
    "phone": 9999999999,  
    "active": "Yes",  
    "store_id": 1,  
    "manager_id": 1  
  },  
  {  
    "staff_id": 2,  
    "first_name": "Sadiksha",  
    "last_name": "Singh",  
    "email": "sadiksha@gmail.com",  
    "phone": 8888888888,  
    "active": "Yes",  
    "store_id": 2,  
    "manager_id": 1  
  },  
  {  
    "staff_id": 3,  
    "first_name": "Priya",  
    "last_name": "Kumar",  
    "email": "priyakumar@gmail.com",  
    "phone": 9898989898,  
    "active": "Yes",  
    "store_id": 3,  
    "manager_id": 2  
  }  
]
```

```
"last_name": "Nadar",
"email": "priya@gmail.com",
"phone": 7777777777,
"active": "Yes",
"store_id": 3,
"manager_id": 1
```

```
}
```

```
]
```

```
)
```

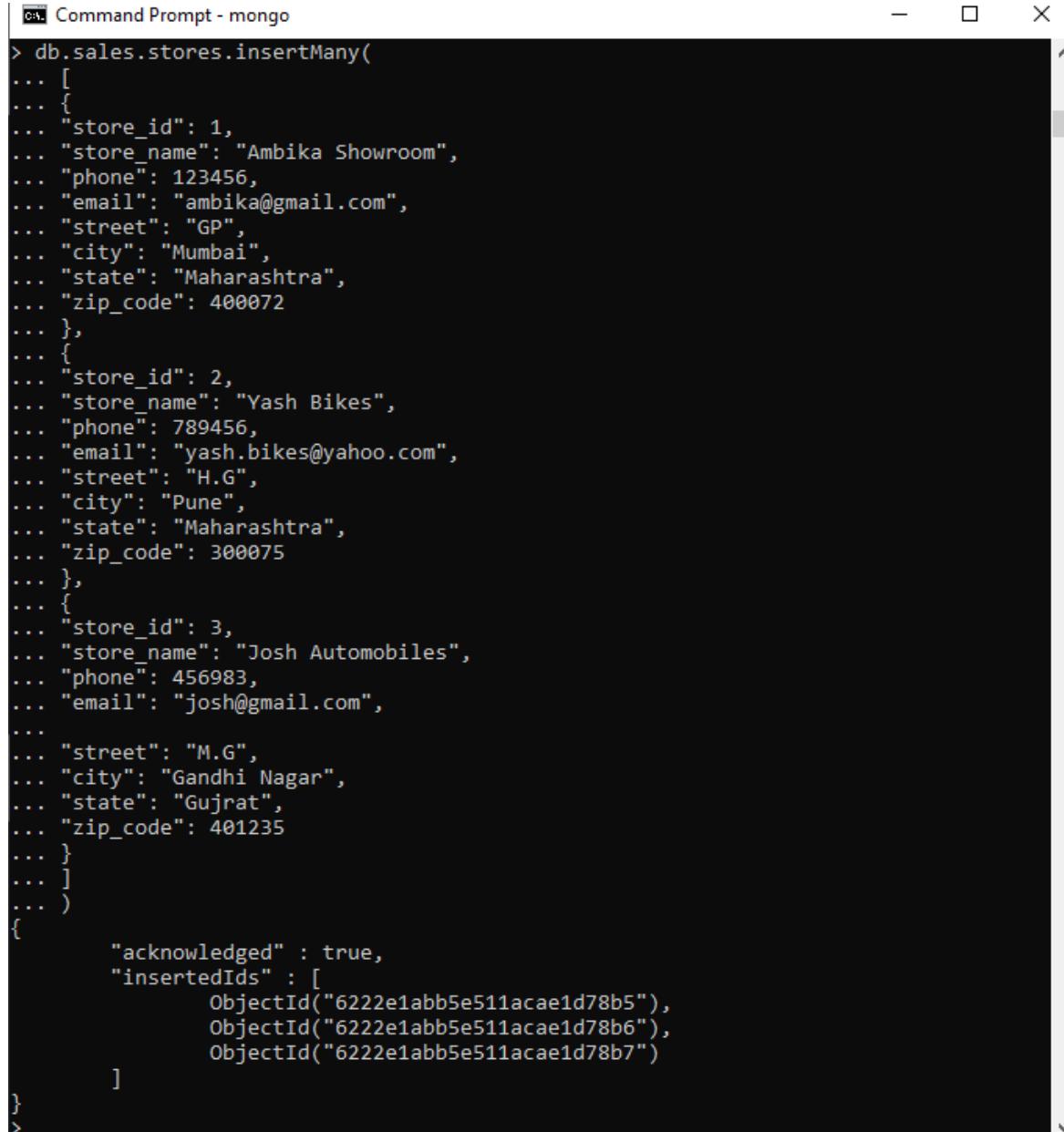
```
| C:\ Command Prompt - mongo
```

```
> db.sales.staffs.insertMany(
... [
... {
... "staff_id": 1,
... "first_name": "Pushpa",
... "last_name": "Yadav",
... "email": "pushpa@gmail.com",
... "phone": 9999999999,
... "active": "Yes",
... "store_id": 1,
... "manager_id": 1
... },
... {
... "staff_id": 2,
... "first_name": "Sadiksha",
... "last_name": "Singh",
... "email": "sadiksha@gmail.com",
... "phone": 8888888888,
... "active": "Yes",
... "store_id": 2,
... "manager_id": 1
... },
... {
... "staff_id": 3,
... "first_name": "Priya",
... "last_name": "Nadar",
... "email": "priya@gmail.com",
... "phone": 7777777777,
... "active": "Yes",
... "store_id": 3,
... "manager_id": 1
...
... }
... ]
... )
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("6222e161b5e511acae1d78b2"),
        ObjectId("6222e161b5e511acae1d78b3"),
        ObjectId("6222e161b5e511acae1d78b4")
    ]
}
>
```

8. sales.stores

```
db.sales.stores.insertMany(
[
{
"store_id": 1,
"store_name": "Ambika Showroom",
"phone": 123456,
"email": "ambika@gmail.com",
"street": "GP",
"city": "Mumbai",
"state": "Maharashtra",
"zip_code": 400072
},
{
```

```
"store_id": 2,  
"store_name": "Yash Bikes",  
"phone": 789456,  
"email": "yash.bikes@yahoo.com",  
"street": "H.G",  
"city": "Pune",  
"state": "Maharashtra",  
"zip_code": 300075  
},  
{  
"store_id": 3,  
"store_name": "Josh Automobiles",  
"phone": 456983,  
"email": "josh@gmail.com",  
  
"street": "M.G",  
"city": "Gandhi Nagar",  
"state": "Gujrat",  
"zip_code": 401235  
}  
]  
)
```



```
Command Prompt - mongo  
> db.sales.stores.insertMany(  
... [  
... {  
... "store_id": 1,  
... "store_name": "Ambika Showroom",  
... "phone": 123456,  
... "email": "ambika@gmail.com",  
... "street": "GP",  
... "city": "Mumbai",  
... "state": "Maharashtra",  
... "zip_code": 400072  
},  
... {  
... "store_id": 2,  
... "store_name": "Yash Bikes",  
... "phone": 789456,  
... "email": "yash.bikes@yahoo.com",  
... "street": "H.G",  
... "city": "Pune",  
... "state": "Maharashtra",  
... "zip_code": 300075  
},  
... {  
... "store_id": 3,  
... "store_name": "Josh Automobiles",  
... "phone": 456983,  
... "email": "josh@gmail.com",  
  
... "street": "M.G",  
... "city": "Gandhi Nagar",  
... "state": "Gujrat",  
... "zip_code": 401235  
}  
]  
)  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222e1abb5e511acae1d78b5"),  
        ObjectId("6222e1abb5e511acae1d78b6"),  
        ObjectId("6222e1abb5e511acae1d78b7")  
    ]  
}
```

9. production.brands

```
db.production.brands.insertMany(  
[  
{  
"brand_id": 1,  
"brand_name": "Honda"  
},  
{  
"brand_id": 2,  
"brand_name": "6KU Bikes"  
},  
{  
"brand_id": 3,  
"brand_name": "Bianchi"  
},  
{  
"brand_id": 4,  
"brand_name": "BMC"  
},  
{  
"brand_id": 5,  
"brand_name": "Huffy"  
}  
]
```

```
)
```

The screenshot shows a Command Prompt window titled "Command Prompt - mongo". The window displays the execution of a MongoDB command to insert multiple documents into the "brands" collection in the "production" database. The command is as follows:

```
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222e1abb5e511acae1d78b5"),  
        ObjectId("6222e1abb5e511acae1d78b6"),  
        ObjectId("6222e1abb5e511acae1d78b7")  
    ]  
}  
> db.production.brands.insertMany(  
... [  
... {  
... "brand_id": 1,  
... "brand_name": "Honda"  
... },  
... {  
... "brand_id": 2,  
... "brand_name": "6KU Bikes"  
... },  
... {  
... "brand_id": 3,  
... "brand_name": "Bianchi"  
... },  
... {  
... "brand_id": 4,  
... "brand_name": "BMC"  
... },  
... {  
... "brand_id": 5,  
... "brand_name": "Huffy"  
... }  
... ]  
... )  
{  
    "acknowledged" : true,  
    "insertedIds" : [  
        ObjectId("6222e1dc5e511acae1d78b8"),  
        ObjectId("6222e1dc5e511acae1d78b9"),  
        ObjectId("6222e1dc5e511acae1d78ba"),  
        ObjectId("6222e1dc5e511acae1d78bb"),  
        ObjectId("6222e1dc5e511acae1d78bc")  
    ]  
}
```

2) Read Operations

The read operations allow you to supply special query filters and criteria that let you specify which documents you want. The MongoDB documentation contains more information on the available query filters. Query modifiers may also be used to change how many results are returned.

MongoDB has two methods of reading documents from a collection:

- **db.collection.find()**
- **db.collection.findOne()**

find()

In order to get all the documents from a collection, we can simply use the find() method on our chosen collection. Executing just the find() method with no arguments will return all records currently in the collection.

Example - db.production.brands.find()

Here we can see that every record has an assigned “ObjectId” mapped to the “_id” key.

```
> db.production.brands.find()
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b8"), "brand_id" : 1, "brand_name" : "Honda" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b9"), "brand_id" : 2, "brand_name" : "6KU Bikes" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78ba"), "brand_id" : 3, "brand_name" : "Bianchi" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bb"), "brand_id" : 4, "brand_name" : "BMC" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bc"), "brand_id" : 5, "brand_name" : "Huffy" }
>
```

If you want to get more specific with a read operation and find a desired subsection of the records, you can use the previously mentioned filtering criteria to choose what results should be returned. One of the most common ways of filtering the results is to search by value.

Example: db.production.brands.find({"brand_name" : "Honda"})

```
> db.production.brands.find({"brand_name" : "Honda"})
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b8"), "brand_id" : 1, "brand_name" : "Honda" }
>
```

findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne() method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

Syntax- db.{collection}.findOne({query}, {projection})

Example : db.sales.order_items.findOne({"quantity": 2})

```
> db.sales.order_items.findOne({"quantity": 2})
{
    "_id" : ObjectId("6222e0bab5e511acae1d78a8"),
    "order_id" : "ORD001",
    "product_id" : 1,
    "quantity" : 2,
    "list_price" : 50000
}>
```

Query Documents

a. **Specify Equality Condition**

db.production.products.find() - to show all the documents

db.production.products.find({"list_price": 30000}) - to show only documents which have “list_price” as 30000

```
> db.sales.customers.find()
{ "_id" : ObjectId("6222e05db5e511acae1d78a3"), "customer_id" : "Cus001", "first_name" : "Jay", "last_name" : "Mehta", "phone" : 1234567890, "email" : "jay@gmail.com", "street" : "T.P Road", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a4"), "customer_id" : "Cus002", "first_name" : "Ruhi", "last_name" : "Singh", "phone" : 7894561230, "email" : "ruhi@yahoo.com", "street" : "M.G Chauk", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a5"), "customer_id" : "Cus003", "first_name" : "Aria", "last_name" : "Josh", "phone" : 1245789630, "email" : "aria.josh@gmail.com", "street" : "JVM", "city" : "Gandhi Nagar", "state" : "Gujrat", "zip_code" : 401235 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a6"), "customer_id" : "Cus004", "first_name" : "Mahi", "last_name" : "Kaur", "phone" : 4567890123, "email" : "kaur.mahi@hotmail.com", "street" : "J.V.L.R", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a7"), "customer_id" : "Cus005", "first_name" : "Aditya", "last_name" : "Yadav", "phone" : 9638527410, "email" : "aditya@gmail.com", "street" : "Koliwada", "city" : "Pune", "state" : "Maharashtra", "zip_code" : 300075 }
> db.production.products.find({list_price: 30000})
{ "_id" : ObjectId("6222df37b5e511acae1d7899"), "product_id" : 2, "product_name" : "GKU Bikes", "brand_id" : 2, "category_id" : 4, "model_year" : 2000, "list_price" : 30000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789a"), "product_id" : 3, "product_name" : "Bianchi", "brand_id" : 3, "category_id" : 2, "model_year" : 2002, "list_price" : 30000 }
>
```

b. Specify Conditions Using Query Operators

db.sales.customers.find() - to show all the documents

db.sales.customers.find({city: { \$in: ["Mumbai", "Pune"] } }) - to show all the documents where "city" is either "Mumbai" or "Pune"

```
> db.sales.customers.find()
{ "_id" : ObjectId("6222e05db5e511acae1d78a3"), "customer_id" : "Cus001", "first_name" : "Jay", "last_name" : "Mehta", "phone" : 1234567890, "email" : "jay@gmail.com", "street" : "T.P Road", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a4"), "customer_id" : "Cus002", "first_name" : "Ruhi", "last_name" : "Singh", "phone" : 7894561230, "email" : "ruhi@yahoo.com", "street" : "M.G Chauk", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a5"), "customer_id" : "Cus003", "first_name" : "Aria", "last_name" : "Josh", "phone" : 1245789630, "email" : "aria.josh@gmail.com", "street" : "JVM", "city" : "Gandhi Nagar", "state" : "Gujrat", "zip_code" : 401235 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a6"), "customer_id" : "Cus004", "first_name" : "Mahi", "last_name" : "Kaur", "phone" : 4567890123, "email" : "kaur.mahi@hotmail.com", "street" : "J.V.L.R", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a7"), "customer_id" : "Cus005", "first_name" : "Aditya", "last_name" : "Yadav", "phone" : 9638527410, "email" : "aditya@gmail.com", "street" : "Koliwada", "city" : "Pune", "state" : "Maharashtra", "zip_code" : 300075 }
> db.sales.customers.find({city: { $in: [ "Mumbai", "Pune" ] } })
{ "_id" : ObjectId("6222e05db5e511acae1d78a3"), "customer_id" : "Cus001", "first_name" : "Jay", "last_name" : "Mehta", "phone" : 1234567890, "email" : "jay@gmail.com", "street" : "T.P Road", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a4"), "customer_id" : "Cus002", "first_name" : "Ruhi", "last_name" : "Singh", "phone" : 7894561230, "email" : "ruhi@yahoo.com", "street" : "M.G Chauk", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a6"), "customer_id" : "Cus004", "first_name" : "Mahi", "last_name" : "Kaur", "phone" : 4567890123, "email" : "kaur.mahi@hotmail.com", "street" : "J.V.L.R", "city" : "Mumbai", "state" : "Maharashtra", "zip_code" : 400072 }
{ "_id" : ObjectId("6222e05db5e511acae1d78a7"), "customer_id" : "Cus005", "first_name" : "Aditya", "last_name" : "Yadav", "phone" : 9638527410, "email" : "aditya@gmail.com", "street" : "Koliwada", "city" : "Pune", "state" : "Maharashtra", "zip_code" : 300075 }
```

c. Specify AND Conditions

db.sales.order_items.find() - to show all the documents

db.sales.order_items.find({quantity: 2, list_price : { \$gt: 70000}}) - Here we are trying to find how sales order item documents for which quantity is 2 and list price is greater than 70000

```
> db.sales.order_items.find()
{ "_id" : ObjectId("6222e0bab5e511acae1d78a8"), "order_id" : "ORD001", "product_id"
: 1, "quantity" : 2, "list_price" : 5000 }
{ "_id" : ObjectId("6222e0bab5e511acae1d78a9"), "order_id" : "ORD002", "product_id"
: 2, "quantity" : 3, "list_price" : 9000 }
{ "_id" : ObjectId("6222e0bab5e511acae1d78aa"), "order_id" : "ORD003", "product_id"
: 3, "quantity" : 1, "list_price" : 3000 }
{ "_id" : ObjectId("6222e0bab5e511acae1d78ab"), "order_id" : "ORD004", "product_id"
: 4, "quantity" : 8, "list_price" : 36000 }
{ "_id" : ObjectId("6222e0bab5e511acae1d78ac"), "order_id" : "ORD005", "product_id"
: 5, "quantity" : 2, "list_price" : 10000 }
> db.sales.order_items.find({quantity: 2, list_price : { $gt: 70000}})
{ "_id" : ObjectId("6222e0bab5e511acae1d78ac"), "order_id" : "ORD005", "product_id"
: 5, "quantity" : 2, "list_price" : 10000 }
```

d. Specify OR Conditions

db.production.products.find() - to show all the documents

db.production.products.find({ \$or: [{ product_name: "Honda Superfast" }, { model_year : { \$lt: 2003 } }] }) - to show all the document which is either "product name" as "Honda Superfast" or "model year" is less than year 2003

```
Command Prompt - MONGO
> db.production.products.find()
{ "_id" : ObjectId("6222df37b5e511acae1d7898"), "product_id" : 1, "product_name" :
"Honda Superfast", "brand_id" : 1, "category_id" : 1, "model_year" : 1994, "list_pr
ice" : 25000 }
{ "_id" : ObjectId("6222df37b5e511acae1d7899"), "product_id" : 2, "product_name" :
"6KU Bikes", "brand_id" : 2, "category_id" : 4, "model_year" : 2000, "list_price" :
30000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789a"), "product_id" : 3, "product_name" :
"Bianchi", "brand_id" : 3, "category_id" : 2, "model_year" : 2002, "list_price" : 3
0000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789b"), "product_id" : 4, "product_name" :
"BMC Hybrid Bike", "brand_id" : 4, "category_id" : 3, "model_year" : 2009, "list_pr
ice" : 45000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789c"), "product_id" : 5, "product_name" :
"Huffy Women Bike", "brand_id" : 5, "category_id" : 7, "model_year" : 2019, "list_p
rice" : 50000 }
> db.production.products.find({ $or: [ { product_name: "Honda Superfast" }, { model
_year : { $lt:
... 2003 } } ] })
{ "_id" : ObjectId("6222df37b5e511acae1d7898"), "product_id" : 1, "product_name" :
"Honda Superfast", "brand_id" : 1, "category_id" : 1, "model_year" : 1994, "list_pr
ice" : 25000 }
{ "_id" : ObjectId("6222df37b5e511acae1d7899"), "product_id" : 2, "product_name" :
"6KU Bikes", "brand_id" : 2, "category_id" : 4, "model_year" : 2000, "list_price" :
30000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789a"), "product_id" : 3, "product_name" :
"Bianchi", "brand_id" : 3, "category_id" : 2, "model_year" : 2002, "list_price" : 3
0000 }
> -
```

3) Update Operations

Like create operations, update operations operate on a single collection, and they are atomic at a single document level. An update operation takes filters and criteria to select the documents you want to update.

You should be careful when updating documents, as updates are permanent and can't be rolled back. This applies to delete operations as well.

For MongoDB CRUD, there are three different methods of updating documents:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

1. updateOne()

We can update a currently existing record and change a single document with an update operation. To do this, we use the `updateOne()` method on a chosen collection. To update a document, we provide the method with two arguments: an update filter and an update action.

The update filter defines which items we want to update, and the update action defines how to update those items. We first pass in the update filter. Then, we use the "`$set`" key and provide the fields we want to update as a value. This method will update the first record that matches the provided filter.

Example –**db.sales.staffs.find()** - to show current document in the system**db.sales.staffs.updateOne({first_name: "Pushpa"}, {\$set:{phone: 999999988}})** - update mobile number from 9999999999 to 999999988 for document name where name is " first_name " in collection

Command Prompt - MONGO

```
> db.production.products.find()
{ "_id" : ObjectId("6222df37b5e511acae1d7898"), "product_id" : 1, "product_name" :
"Honda Superfast", "brand_id" : 1, "category_id" : 1, "model_year" : 1994, "list_pr
ice" : 25000 }
{ "_id" : ObjectId("6222df37b5e511acae1d7899"), "product_id" : 2, "product_name" :
"6KU Bikes", "brand_id" : 2, "category_id" : 4, "model_year" : 2000, "list_price" :
30000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789a"), "product_id" : 3, "product_name" :
"Bianchi", "brand_id" : 3, "category_id" : 2, "model_year" : 2002, "list_price" : 3
0000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789b"), "product_id" : 4, "product_name" :
"BMC Hybrid Bike", "brand_id" : 4, "category_id" : 3, "model_year" : 2009, "list_pr
ice" : 45000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789c"), "product_id" : 5, "product_name" :
"Huffy Women Bike", "brand_id" : 5, "category_id" : 7, "model_year" : 2019, "list_p
rice" : 50000 }
> db.production.products.find({ $or: [ { product_name: "Honda Superfast" }, { model
_year : { $lt:
... 2003 } } ] })
{ "_id" : ObjectId("6222df37b5e511acae1d7898"), "product_id" : 1, "product_name" :
"Honda Superfast", "brand_id" : 1, "category_id" : 1, "model_year" : 1994, "list_pr
ice" : 25000 }
{ "_id" : ObjectId("6222df37b5e511acae1d7899"), "product_id" : 2, "product_name" :
"6KU Bikes", "brand_id" : 2, "category_id" : 4, "model_year" : 2000, "list_price" :
30000 }
{ "_id" : ObjectId("6222df37b5e511acae1d789a"), "product_id" : 3, "product_name" :
"Bianchi", "brand_id" : 3, "category_id" : 2, "model_year" : 2002, "list_price" : 3
0000 }
> db.sales.staffs.find()
{ "_id" : ObjectId("6222e161b5e511acae1d78b2"), "staff_id" : 1, "first_name" : "Pus
hpaa", "last_name" : "Yadav", "email" : "pushpa@gmail.com", "phone" : 9999999999, "a
ctive" : "Yes", "store_id" : 1, "manager_id" : 1 }
{ "_id" : ObjectId("6222e161b5e511acae1d78b3"), "staff_id" : 2, "first_name" : "Sad
iksha", "last_name" : "Singh", "email" : "sadiksha@gmail.com", "phone" : 8888888888
, "active" : "Yes", "store_id" : 2, "manager_id" : 1 }
{ "_id" : ObjectId("6222e161b5e511acae1d78b4"), "staff_id" : 3, "first_name" : "Pri
ya", "last_name" : "Nadar", "email" : "priya@gmail.com", "phone" : 7777777777, "act
ive" : "Yes", "store_id" : 3, "manager_id" : 1 }
> db.sales.staffs.updateOne({first_name: "Pushpa"}, {$set:{phone: 999999988}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

2. updateMany()

updateMany() allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.

Example –**db.sales.orders.find()** - to show current document in the system**db.sales.orders.updateMany({shipped_date:44377}, {\$set: {shipped_date: "1-July-2021"}})** – with the help of this command we are updating "shipped_date" to "1-July-2021" where shipped_date is 44377

Command Prompt - MONGO

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.sales.orders.find()
{ "_id" : ObjectId("6222e0f7b5e511acae1d78ad"), "order_id" : "ORD001", "customer_id
" : "Cus001", "order_status" : "Completed", "order_date" : 43992, "shipped_date" :
43994, "store_id" : 1, "staff_id" : 1 }
{ "_id" : ObjectId("6222e0f7b5e511acae1d78ae"), "order_id" : "ORD002", "customer_id
" : "Cus002", "order_status" : "Completed", "order_date" : 44221, "shipped_date" :
44227, "store_id" : 2, "staff_id" : 2 }
{ "_id" : ObjectId("6222e0f7b5e511acae1d78af"), "order_id" : "ORD003", "customer_id
" : "Cus003", "order_status" : "Completed", "order_date" : 44306, "shipped_date" :
44314, "store_id" : 2, "staff_id" : 2 }
{ "_id" : ObjectId("6222e0f7b5e511acae1d78b0"), "order_id" : "ORD004", "customer_id
" : "Cus004", "order_status" : "Pending", "order_date" : 44367, "shipped_date" : 44
377, "store_id" : 3, "staff_id" : 3 }
{ "_id" : ObjectId("6222e0f7b5e511acae1d78b1"), "order_id" : "ORD005", "customer_id
" : "Cus005", "order_status" : "Pending", "order_date" : 44367, "shipped_date" : 44
377, "store_id" : 1, "staff_id" : 1 }
> db.sales.orders.updateMany({shipped_date:44377}, {$set: {shipped_date: "1-July-20
21"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
>
```

3. replaceOne()

The replaceOne() method is used to replace a single document in the specified collection. replaceOne() replaces the entire document, meaning fields in the old document not contained in the new will be lost.

4) Delete Operations

Delete operations operate on a single collection, like update and create operations. Delete operations are also atomic for a single document. You can provide delete operations with filters and criteria in order to specify which documents you would like to delete from a collection. The filter options rely on the same syntax that read operations utilize.

MongoDB has two different methods of deleting records from a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

1. deleteOne()

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criteria is used to specify the item to delete. It deletes the first record that matches the provided filter.

Example –

db.production.brands.find() - to show current documents in collection

db.production.brands.deleteOne({brand_id:6}) - delete the document where "brand_id" is 6

```
> db.production.brands.find()
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b8"), "brand_id" : 1, "brand_name" : "Hon
da" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b9"), "brand_id" : 2, "brand_name" : "6KU
Bikes" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78ba"), "brand_id" : 3, "brand_name" : "Bia
nchi" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bb"), "brand_id" : 4, "brand_name" : "BMC
" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bc"), "brand_id" : 5, "brand_name" : "Huf
fy" }
> db.production.brands.deleteOne({brand_id:6})
{ "acknowledged" : true, "deletedCount" : 0 }
```

2. deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in deleteOne().

Example –

db.production.brands.find() - to show current documents in collection

db.production.brands.deleteMany({brand_id: {\$gt: 5}}) - delete documents for which brand id is greater than 5

```
Command Prompt - MONGO
> db.production.brands.deleteOne({brand_id:6})
{ "acknowledged" : true, "deletedCount" : 0 }
> db.production.brands.find()
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b8"), "brand_id" : 1, "brand_name" : "Hon
da" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78b9"), "brand_id" : 2, "brand_name" : "6KU
Bikes" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78ba"), "brand_id" : 3, "brand_name" : "Bia
nchi" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bb"), "brand_id" : 4, "brand_name" : "BMC
" }
{ "_id" : ObjectId("6222e1dcb5e511acae1d78bc"), "brand_id" : 5, "brand_name" : "Huf
fy" }
> db.production.brands.deleteMany({brand_id: {$gt: 5}})
{ "acknowledged" : true, "deletedCount" : 0 }
```

Aim: Joins, Sorting, Subqueries using HiveQL

JOINS

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database.

There are different types of joins given as follows:

- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- **JOIN**

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

➤ **LEFT OUTER JOIN**

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table. A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

➤ **RIGHT OUTER JOIN**

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate

➤ **FULL OUTER JOIN**

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

SUB QUERIES:

A Query present within a Query is known as a sub query. The main query will depend on the values returned by the subqueries.

Subqueries can be classified into two types

- Subqueries in **FROM** clause
- Subqueries in **WHERE** clause

When to use:

- To get a particular value combined from two column values from different tables
- Dependency of one table values on other tables
- Comparative checking of one column values from other tables

SORTING

The SORT BY syntax is similar to the syntax of ORDER BY in SQL language.

Hive supports SORT BY which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results. Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The sort order will be dependent on the column types. If the column is of numeric type, then the sort order is also in numeric order. If the column is of string type, then the sort order will be lexicographical order

Before the perform the practical first perform 3 steps of the given following

1. sudo /home/cloudera/cloudera-manager --force --express

```

cloudera@quickstart:~$ sudo /home/cloudera/cloudera-manager --force --express
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Starting Cloudera Manager agent...
[QuickStart] Configuring deployment...
Submitted jobs: 41
[QuickStart] Deploying client configuration...
Submitted jobs: 42
[QuickStart] Starting Cloudera Management Service...
Submitted jobs: 50
[QuickStart] Enabling Cloudera Manager daemons on boot...

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:
http://quickstart.cloudera:7180

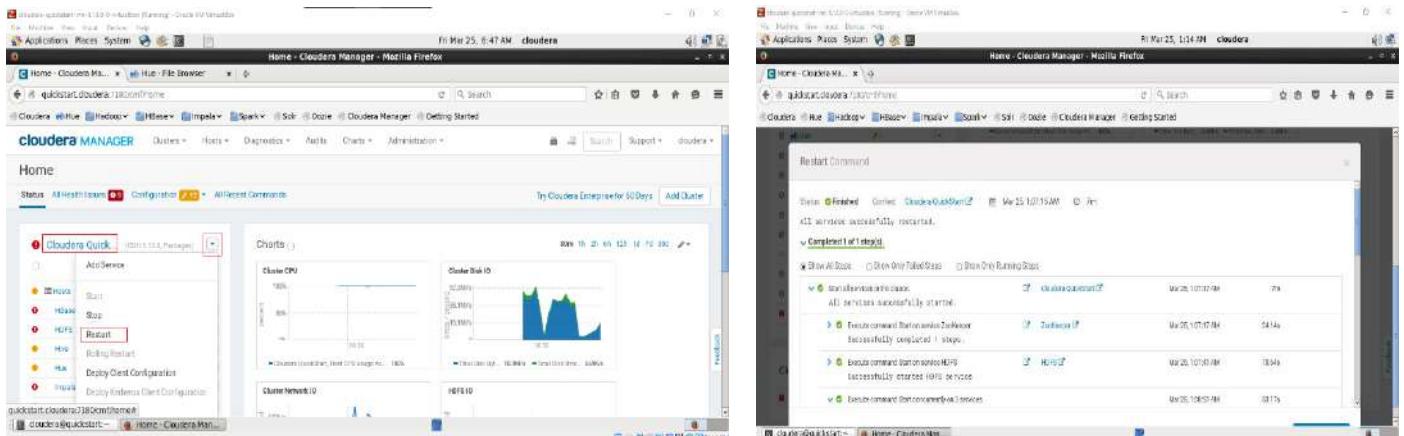
Username: cloudera
Password: cloudera

[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

22/03/25 01:04:22 WARN ipc.Client: Failed to connect to server: quickstart.cloudera/10.0.2.15:8020: try once and fail.
java.net.ConnectException: Connection refused
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:739)
    at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:266)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)
    at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:648)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:744)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:396)

```

2. Start all services



3. sudo -u hdfs hadoop dfsadmin -safemode leave

```

cloudera@quickstart:~$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

safeMode: Call From quickstart.cloudera/10.0.2.15 to quickstart.cloudera:8020 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$

```

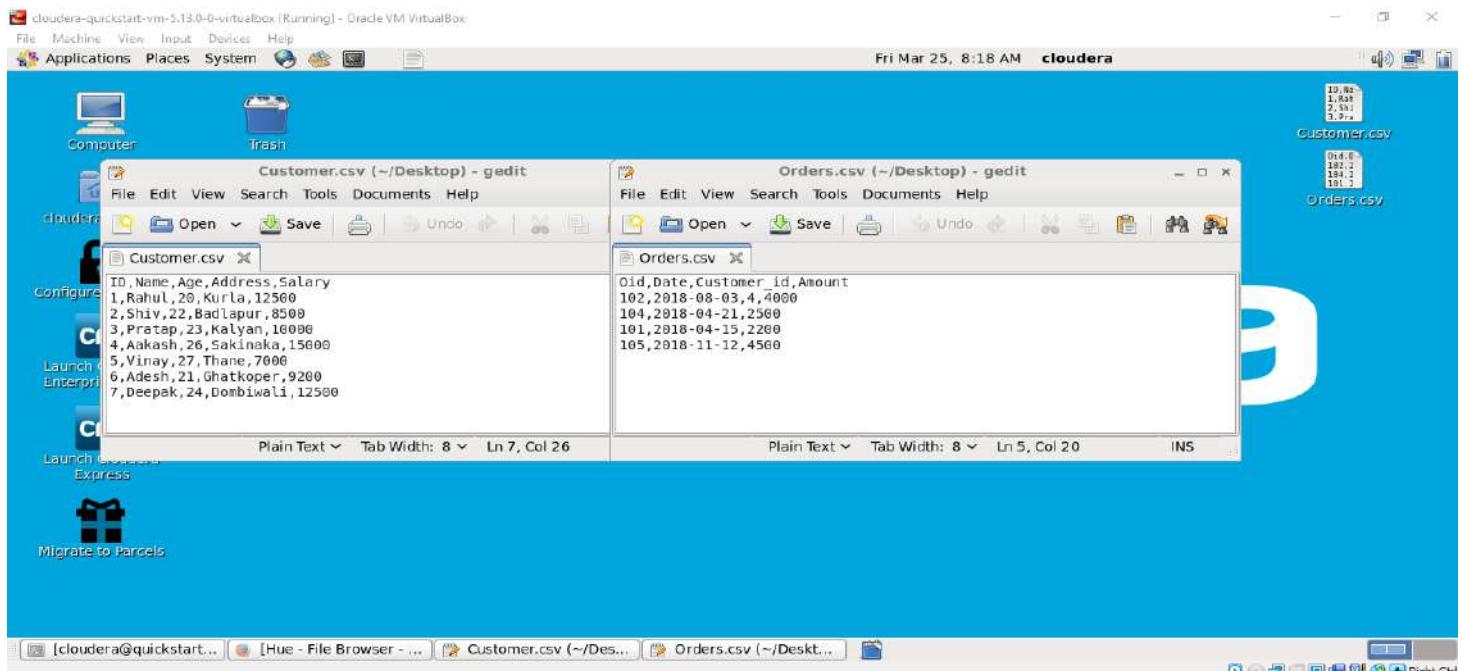
First we will create the **Customer.csv** and **Order.csv** file.

Customer.csv

ID	Name	Age	Address	Salary
1	Rahul	20	Kurla	12500
2	Shiv	22	Badlapur	8500
3	Pratap	23	Kalyan	10000
4	Aakash	26	Sakinaka	15000
5	Vinay	27	Thane	7000
6	Adesh	21	Ghatkoper	9200
7	Deepak	24	Dombiwali	12500

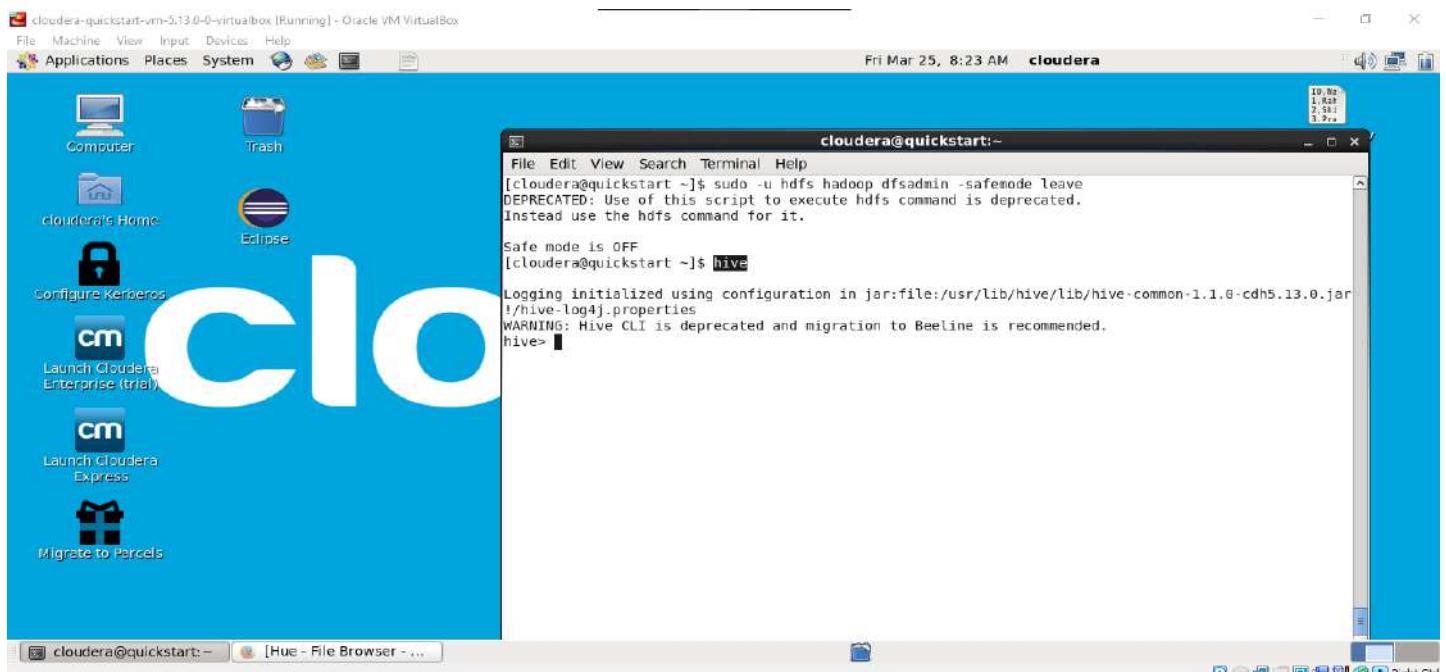
Orders.csv

Oid	Date	Customer_id	Amount
102	2018-08-03	4	4000
104	2018-04-21	1	2500
101	2018-04-15	1	2200
105	2018-11-12	2	4500



Open the terminal, now we use **hive** command to enter the **hive shell prompt** and in hive shell we could execute all of the hive commands.

1. **hive**



```

cloudera@quickstart:~$ show databases;
OK
default
Time taken: 3.497 seconds, Fetched: 1 row(s)
hive> 

```

The screenshot shows a desktop environment for Cloudera QuickStart VM. A terminal window titled "cloudera@quickstart:~" is open, displaying the output of a Hive command. The command "show databases;" was run, followed by "OK". The output also includes the time taken for the query. The desktop background features the Cloudera logo.

Now we will be creating a new database named as rjc_joins using below command,

2. create database rjc_joins;

```

cloudera@quickstart:~$ show databases;
OK
default
Time taken: 3.497 seconds, Fetched: 1 row(s)
hive> create database rjc_joins;
OK
Time taken: 0.371 seconds
hive> 

```

The screenshot shows a desktop environment for Cloudera QuickStart VM. A terminal window titled "cloudera@quickstart:~" is open, displaying the output of a Hive command. The command "create database rjc_joins;" was run, followed by "OK". The output also includes the time taken for the query. The desktop background features the Cloudera logo.

And then showing the databases.

show databases;

```

cloudera@quickstart:~$ show databases;
OK
default
Time taken: 3.497 seconds, Fetched: 1 row(s)
hive> create database rjc_joins;
OK
Time taken: 0.371 seconds
hive> show databases;
OK
default
[ rjc_joins ]
Time taken: 0.061 seconds, Fetched: 2 row(s)
hive> 

```

The screenshot shows a desktop environment for Cloudera QuickStart VM. A terminal window titled "cloudera@quickstart:~" is open, displaying the output of a Hive command. The command "show databases;" was run, followed by "OK". The output lists the database "rjc_joins" and its status. The desktop background features the Cloudera logo.

Now to work inside this database we use below command;

3. use rjc_joins;

```

cloudera@quickstart:~$ hive> show databases;
OK
default
Time taken: 3.497 seconds. Fetched: 1 row(s)
hive> create database rjc_joins;
OK
Time taken: 0.371 seconds
hive> show databases;
OK
default
rjc_joins
Time taken: 0.061 seconds. Fetched: 2 row(s)
hive> use rjc_joins;
OK
Time taken: 0.249 seconds
hive>

```

Now we will create two tables in one table we will load the **Customer.csv** file and in the other table we will load **Orders.csv** file.

4. create table customers(ID int, Name string, Age int, Address string, Salary float)

row format delimited

fields terminated by ','

tblproperties("skip.header.line.count"="1");

```

cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ Fri Mar 25, 8:36 AM cloudera
cloudera@quickstart:~$ hive> create table customers(ID int, Name string, Age int, Address string, Salary float)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count"="1");
OK
Time taken: 2.289 seconds
hive>

```

show tables;

```

cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ Fri Mar 25, 8:36 AM cloudera
cloudera@quickstart:~$ hive> create table customers(ID int, Name string, Age int, Address string, Salary float)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count"="1");
OK
Time taken: 2.289 seconds
hive> show tables;
OK
customers
Time taken: 0.759 seconds, Fetched: 1 row(s)
hive>

```

Now we will see the schema of the table using describe command, **describe customers;**

describe customers;

```
hive> describe customers;
OK
id          int
name        string
age         int
address    string
salary      float
Time taken: 0.782 seconds, Fetched: 5 row(s)
hive> [REDACTED]
```

Now loading data in the **customers** table from **Customer.csv** file which present inside **/home/cloudera/<dir_name>** directory.

5. load data local inpath "/home/cloudera/Desktop/Customer.csv" into table customers;

```
hive> load data local inpath "/home/cloudera/Desktop/Customer.csv" into table customers
-> ;
Loading data to table rjc_joins.customers
Table rjc_joins.customers stats: [numFiles=1, totalSize=203]
OK
Time taken: 4.24 seconds
hive> [REDACTED]
```

select * from customers;

```
hive> select * from customers;
OK
1    Rahul  20    Kurla   12500.0
2    Shiv   22    Badlapur  8500.0
3    Pratap 23    Kalyan   10000.0
4    Aakash 26    Sakinaka  15000.0
5    Vinay  27    Thane    7000.0
6    Adesh  21    Ghatkoper 9200.0
7    Deepak 24    Dombiwali 12500.0
NULL NULL NULL NULL NULL
Time taken: 1.7 seconds, Fetched: 8 row(s)
hive> [REDACTED]
```

Creating a second table named as **orders** using below command,

6. create table orders(oid int, odate date, cid int, amount float)

row format delimited

fields terminated by ','

tblproperties("skip.header.line.count"="1");

```
hive> create table orders(oid int, odate date, cid int, amount float)
-> row format delimited
-> fields terminated by ','
-> tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.558 seconds
hive> [REDACTED]
```

Now we will see the schema of the table using **describe** command,

describe orders;

```
hive> describe orders
-> ;
OK
oid          int
odate       date
cid          int
amount      float
Time taken: 0.443 seconds, Fetched: 4 row(s)
hive> [REDACTED]
```

Now loading data in the **orders** table from **Orders.csv** file which present inside `/home/cloudera/<dir_name>` directory.

7. load data local inpath "/home/cloudera/Desktop/Orders.csv" into table orders;

```
hive> load data local inpath "/home/cloudera/Desktop/Orders.csv" into table orders
> ;
Loading data to table rjc_joins.orders
Table rjc_joins.orders stats: [numFiles=1, totalSize=111]
OK
Time taken: 1.422 seconds
hive>
```

`select * from orders;`

```
hive> Select * from orders;
OK
102 2018-08-03 4 4000.0
104 2018-04-21 2500 NULL
101 2018-04-15 2200 NULL
105 2018-11-12 4500 NULL
Time taken: 0.381 seconds, Fetched: 4 row(s)
hive> truncate table orders;
OK
Time taken: 1.605 seconds
hive> load data local inpath "/home/cloudera/Desktop/Orders.csv" into table orders;
Loading data to table rjc_joins.orders
Table rjc_joins.orders stats: [numFiles=1, numRows=0, totalSize=116, rawDataSize=0]
OK
Time taken: 3.16 seconds
hive> select * from orders;
OK
102 2018-08-03 4 4000.0
104 2018-04-21 1 2500.0
101 2018-04-15 1 2200.0
105 2018-11-12 2 4500.0
Time taken: 0.851 seconds, Fetched: 4 row(s)
hive>
```

JOIN

Now first we apply the normal joins on the two tables using below command, we want to retrieve customer id, name, age from customers table and amount from the orders table and join perform on id of the customers and orders table.

8. select c.id, c.name, c.age, o.amount

from customers c JOIN orders o

on (c.id = o.cid);

```
hive> select c.id, c.name, c.age, o.amount
> from customers c JOIN orders o
> on (c.id = o.cid);
Query ID = cloudera_20220325085858_6ac28064-2d67-4188-9d19-e62b7ee0c754
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20220325085858_6ac28064-2d67-4188-9d19-e62b7ee0c754.log
```

Mapreduce task is performed

```
cloudera-quickstart-vm-5:13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System Terminal Help
cloudera@quickstart:~$ Fri Mar 25, 9:01 AM cloudera
File Edit View Search Terminal Help
37064
2022-03-25 08:59:32 Dump the side-table for tag: 1 with group count: 3 into file: file:/tmp/cloud
69Table-Stage-3/MapJoinMapfile01--hashtable
2022-03-25 08:59:33 Uploaded 1 File to: file:/tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/h
ive.2022-03-25.00:58:49_824_409203809531254827-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.
hashtable (338 bytes)
2022-03-25 08:59:33 End of local task; Time Taken: 13.051 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job - job_1648216557393_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/applicati
on_1648216557393_0005
Kill Command: /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0005
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2022-03-25 09:00:27,605 Stage-3 map = 0%, reduce = 0%
2022-03-25 09:01:12,537 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 6.25 sec
MapReduce Total cumulative CPU time: 6 seconds 250 msec
Ended Job = job_1648216557393_0005
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Cumulative CPU: 6.25 sec HDFS Read: 7098 HDFS Write: 72 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 250 msec
OK
1 Rahul 20 2500.0
1 Rahul 20 2200.0
2 Shiv 22 4500.0
4 Akash 20 4000.0
Time taken: 155.387 seconds, Fetched: 4 row(s)
hive>
```

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

**9. select c.id, c.name, o.amount,o.odate
from customers c LEFT OUTER JOIN orders o
on (c.id = o.cid);**

```
hive> select c.id, c.name, o.amount,o.odate
> from customers c LEFT OUTER JOIN orders o
> on (c.id = o.cid);
Query ID = cloudera_20220325090202_75101a32-531d-4e2c-9d17-1e716cf37995
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20220325090202_75101a32-531d-4e2c-9d17-1e716cf37995.log
2022-03-25 09:03:25      Starting to launch local task to process map join;      maximum memory = 1195
37664
```

Mapreduce task is performed

```
cloudera@quickstart:~ [Hue - File Browser - ...] Fri Mar 25, 9:09 AM cloudera
File Edit View Search Terminal Help
2022-03-25 09:07:43      Uploaded 1 File to: file:/tmp/cloudera/2a5ce55c-e04c-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-07-11_649_10688202630938277681-/local-1003/HashTable-Stage-3/MapJoin-mapfile21--.hashtable (350 bytes)
2022-03-25 09:07:43      End of local task; Time Taken: 8.349 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Steaming Job = job_1648216557393_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0007/
Kill Command: /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0007
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2022-03-25 09:08:28,715 Stage-3 map = 0%, reduce = 0%
2022-03-25 09:09:01,145 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 5.11 sec
MapReduce Total cumulative CPU time: 5 seconds 110 msec
Ended Job = job_1648216557393_0007
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Cumulative CPU: 5.11 sec  HDFS Read: 7156 HDFS Write: 162 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 110 msec
OK
1      Rahul  2800.0  2018-04-21
1      Rahul  2200.0  2018-04-15
2      Shiv   4500.0  2018-11-12
3      Pratap  NULL    NULL
4      Akash  4000.0  2018-08-03
5      Vinay  NULL    NULL
6      Adesh  NULL    NULL
7      Deepak NULL    NULL
Time taken: 111.942 seconds, Fetched: 8 row(s)
hive>
```

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

**10. select c.id, c.name, o.amount,o.odate
from customers c RIGHT OUTER JOIN orders o
on (c.id = o.cid);**

```
hive> select c.id, c.name, o.amount,o.odate
> from customers c RIGHT OUTER JOIN orders o
> on (c.id = o.cid);
Query ID = cloudera_20220325090909_f1d30aa1-55a3-4835-ab4c-d7263d18f42d
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20220325090909_f1d30aa1-55a3-4835-ab4c-d7263d18f42d.log
2022-03-25 09:10:20      Starting to launch local task to process map join;      maximum memory = 1195
37664
2022-03-25 09:10:26      Dump the side-table for tag: 0 with group count: 7 into file: file:/tmp/cloud
era/2a5ce55c-e04c-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_406453300447223904-1/-local-10
03/HashTable-Stage-3/MapJoin-mapfile30--.hashtable
```

Mapreduce task is performed

```

cloudera@quickstart:~$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.7.1.jar sort -input /tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable
2022-03-25 09:10:26     Dump the side-table for tag: 0 with group count: 7 into file: file:/tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable
2022-03-25 09:10:27     Uploaded 1 File to: file:/tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable (437 bytes)
2022-03-25 09:10:27     End of local task; Time Taken: 6.154 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job job_1648216557393_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0008
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2022-03-25 09:11:17.568 Stage-3 map = 0%, reduce = 0%
2022-03-25 09:12:09.647 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 5.4 sec
MapReduce Total cumulative CPU time: 5 seconds 400 msec
Ended Job = job_1648216557393_0008
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 5.4 sec   HDFS Read: 7166 HDFS Write: 104 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 400 msec
OK
4      Akash  4600.0  2018-08-03
1      Rahul   2500.0  2018-04-21
1      Rohul   2200.0  2018-04-15
2      Shiv    4500.0  2018-11-12
Time taken: 125.63 seconds, Fetched: 4 row(s)
hive>

```

Now we will be using the concept of **subqueries** for finding the second largest salary from the customers table.

Sub Queries

A Query present within a Query is known as a sub query. The main query will depend on the values returned by the subqueries.

Subqueries can be classified into two types

- Subqueries in FROM clause
- Subqueries in WHERE clause

11. select max(salary) from customers where customers.salary not in(select max(salary) from customers);

```

hive> select max(salary) from customers where customers.salary not in(select max(salary) from customers);
Warning: Map Join MAPJOIN[62][bigTable=customers] in task 'Stage-8:MAPRED' is a cross product
Warning: Shuffle Join JOIN[24][tables = [customers, sq_1_notin_nullcheck]] in Stage 'Stage-1:MAPRED' is a cross product
Query ID = cloudera_20220325091414_55e79d66-e08d-419a-a4f1-7eaf6b4960fc
Total jobs = 7
Launching Job 1 out of 7
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648216557393_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0009

```

Mapreduce task is performed

```

cloudera@quickstart:~$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.7.1.jar sort -input /tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable
2022-03-25 09:10:26     Dump the side-table for tag: 0 with group count: 7 into file: file:/tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable
2022-03-25 09:10:27     Uploaded 1 File to: file:/tmp/cloudera/2a5ce55c-e04e-490f-a2e1-11dacd82dec6/hive_2022-03-25_09-09-58_105_4064533000447223904+1/-local-10003/HashTable-Stage-3/MapJoin-mapfile30--.hashtable (437 bytes)
2022-03-25 09:10:27     End of local task; Time Taken: 6.154 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job job_1648216557393_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0013
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2022-03-25 09:22:06.415 Stage-3 map = 0%, reduce = 0%
2022-03-25 09:22:46.757 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.23 sec
2022-03-25 09:23:14.132 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 8.68 sec
MapReduce Total cumulative CPU time: 8 seconds 680 msec
Ended Job = job_1648216557393_0013
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1 Reduce: 1   Cumulative CPU: 9.92 sec   HDFS Read: 7333 HDFS Write: 117 SUCCESS
Stage-Stage-3: Map: 1 Reduce: 1   Cumulative CPU: 10.50 sec   HDFS Read: 8796 HDFS Write: 114 SUCCESS
Stage-Stage-8: Map: 1 Cumulative CPU: 5.61 sec   HDFS Read: 5349 HDFS Write: 243 SUCCESS
Stage-Stage-6: Map: 1 Cumulative CPU: 5.06 sec   HDFS Read: 5116 HDFS Write: 117 SUCCESS
Stage-Stage-3: Map: 1 Reduce: 1   Cumulative CPU: 8.68 sec   HDFS Read: 4715 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 40 seconds 630 msec
OK
12500.0
Time taken: 536.151 seconds, Fetched: 1 row(s)
hive>

```

As we can see from the above output the second largest salary is **12500.00**.

Sorting

The SORT BY syntax is similar to the syntax of ORDER BY in SQL language.

Hive supports **SORT BY** which sorts the data per reducer. The difference between "order by" and "sort by" is that the former guarantees total order in the output while the latter only guarantees ordering of the rows within a reducer. If there are more than one reducer, "sort by" may give partially ordered final results.

Hive uses the columns in SORT BY to sort the rows before feeding the rows to a reducer. The sort order will be dependent on the column types. If the column is of numeric type, then the sort order is also in numeric order. If the column is of string type, then the sort order will be lexicographical order.

LIMIT can be used to minimize sort time.

Now finding the **fourth largest salary** from the customers table using **Sort by** clause.

12. select salary from customers sort by salary desc limit 4;

It will give the only 4 records in the output after sorting them in descending order. This is not a complete syntax only we are showing what output it will give.

```

cloudera@quickstart:~/Desktop$ hive> select salary from customers sort by salary desc limit 4;
hive> select salary from customers sort by salary desc limit 4;
Query ID = cloudera_20220325092626_a53cd55a-bcf3-4f4b-9011-e9b534a186c6
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648216557393_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0014/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0014

```

Mapreduce task is performed

```

cloudera@quickstart:~/Desktop$ hive> select salary from customers sort by salary desc limit 4;
hive> select salary from customers sort by salary desc limit 4;
Query ID = cloudera_20220325092626_a53cd55a-bcf3-4f4b-9011-e9b534a186c6
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648216557393_0015, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0015/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0015
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2022-03-25 09:28:51,481 Stage-2 map = 0%, reduce = 0%
2022-03-25 09:29:16,761 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 3.54 sec
2022-03-25 09:29:50,130 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 8.41 sec
MapReduce Total cumulative CPU time: 8 seconds 410 msec
Ended Job = job_1648216557393_0015
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.3 sec  HDFS Read: 6378 HDFS Write: 180 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 8.41 sec  HDFS Read: 4598 HDFS Write: 32 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 710 msec
OK
12500.0
12500.0
12500.0
12500.0
Time taken: 207.236 seconds. Fetched: 4 row(s)
hive> 
```

Now what records which we have got by executing the above queries now we will use this query as subqueries and we will now sort them in ascending order to find fourth largest salary of customer table.

13. select salary from(select salary from customers sort by salary desc limit 4) result sort by salary asc limit 1;

Now whatever result we get from subquery we will store them in result table and then it will sort the result table in ascending order and as we want fourth largest salary so we are limiting it to 1.

```

cloudera@quickstart:~$ Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2022-03-25 09:28:51,481 Stage-2 map = 0%, reduce = 0%
2022-03-25 09:29:16,701 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 3.54 sec
2022-03-25 09:29:50,130 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 8.41 sec
MapReduce Total cumulative CPU time: 8 seconds 416 msec
Ended Job = job_1648216557393_0015
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.3 sec HDFS Read: 6378 HDFS Write: 189 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 8.41 sec HDFS Read: 4598 HDFS Write: 32 SUCCESS
Total MapReduce CPU Time Spent: 16 seconds 710 msec
OK
15600.8
12500.0
12500.0
10000.0
Time taken: 207.236 seconds, Fetched: 4 row(s)
hive> select salary from(select salary from customers sort by salary desc limit 4) result sort by salary asc limit 1;
Query ID = cloudera_20220325093131_f5ecc9df_b29a-489c-aee9-0d651ff6ce21
Total jobs = 4
Launching Job 1 out of 4
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<numbers>
Starting Job = job_1648216557393_0016, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0016/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0016

```

```

cloudera@quickstart:~$ Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2022-03-25 09:36:23,718 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.21 sec
2022-03-25 09:36:52,878 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 7.36 sec
MapReduce Total cumulative CPU time: 7 seconds 360 msec
Ended Job = job_1648216557393_0018
Launching Job 4 out of 4
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648216557393_0019, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648216557393_0019/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648216557393_0019
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2022-03-25 09:37:24,623 Stage-4 map = 0%, reduce = 0%
2022-03-25 09:37:55,931 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 3.53 sec
2022-03-25 09:38:23,488 Stage-4 map = 100%, reduce = 100%, Cumulative CPU 8.74 sec
MapReduce Total cumulative CPU time: 8 seconds 740 msec
Ended Job = job_1648216557393_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.65 sec HDFS Read: 6395 HDFS Write: 180 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 8.21 sec HDFS Read: 3735 HDFS Write: 180 SUCCESS
Stage-Stage-3: Map: 1 Reduce: 1 Cumulative CPU: 7.36 sec HDFS Read: 3753 HDFS Write: 117 SUCCESS
Stage-Stage-4: Map: 1 Reduce: 1 Cumulative CPU: 8.74 sec HDFS Read: 4540 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 960 msec
OK
10000.0
Time taken: 393.461 seconds, Fetched: 1 row(s)
hive> 
```

Now we got the fourth largest salary i.e. 10000.0 as an output.

Aim: Querying, Sorting, Aggregating data using HiveQL

What is HIVE:

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive:

These are the following features of Hive:

- ❖ Hive is fast and scalable.
- ❖ It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- ❖ It is capable of analyzing large datasets stored in HDFS.
- ❖ It allows different storage types such as plain text, RCFile, and HBase.
- ❖ It uses indexing to accelerate queries.
- ❖ It can operate on compressed data stored in the Hadoop ecosystem.
- ❖ It supports user-defined functions (UDFs) where user can provide its functionality.

Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Before the perform the practical first perform 3 steps of the given following

1. sudo /home/cloudera/cloudera-manager --force --express

```

cloudera-quickstart-vm-5.11.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Fri Mar 25, 1:15 AM cloudera@quickstart:~
File Edit View Search Terminal Help
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Starting Cloudera Manager agent...
[QuickStart] Configuring deployment...
Submitted jobs: 41
[QuickStart] Deploying client configuration...
Submitted jobs: 42
[QuickStart] Starting Cloudera Management Service...
Submitted jobs: 50
[QuickStart] Enabling Cloudera Manager daemons on boot...

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:
http://quickstart.cloudera:7180
Username: cloudera
Password: cloudera

[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

22/03/25 01:04:22 WARN ipc.Client: Failed to connect to server: quickstart.cloudera/10.0.2.15:8020: try once and fail.
java.net.ConnectException: Connection refused
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:739)
    at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:206)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)
    at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:648)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:744)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:396)

```

2. Start all services

The screenshot shows the Cloudera Manager interface. On the left, the 'Clouds' sidebar has 'cloudera' selected. Under 'All Services', 'HDFS' is expanded, and 'Restart' is highlighted. In the center, there are two charts: 'Cluster CPU' and 'Cluster Disk IO'. On the right, a 'Restart Command' dialog is open, showing a table of completed commands. One command, 'Execute command: sudo -u hdfs hadoop dfsadmin -safemode leave', is listed with a status of 'Success'.

3. sudo -u hdfs hadoop dfsadmin -safemode leave

```
cloudera@quickstart-vmn-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ sudo -u hdfs hadoop dfsadmin -safemode leave
[sudo] password for cloudera: 
[sudo] password for hdfs: 
cloudera@quickstart:~$
```

The terminal window shows the command being run. It prompts for the sudo password ('cloudera') and the hdfs user password ('hdfs'). The output indicates that the command was successful, although it includes a warning about the 'dfsadmin' command being deprecated.

1. Open the terminal, Now we use hive command to enter the hive shell prompt and in hive shell we could execute all of the hive commands.

hive

```
cloudera@quickstart:~$ File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

The terminal window shows the 'hive' command being run. It prints a warning about the CLI being deprecated and recommends using Beeline instead. The prompt changes to 'hive>'.

2. Now we will see the databases which are already existing using below command.

show databases;

```
cloudera@quickstart:~$ File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hive> show databases;
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> OK
hive> default
Time taken: 3.088 seconds, Fetched: 1 row(s)
hive>
```

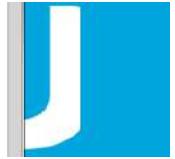
The terminal window shows the 'show databases' command being run. It prints a warning about the CLI being deprecated. The output shows a single database named 'default'.

Note: If remove database already present in hive then **drop database <database_name> cascade;**

3. Creating a database name 'RJC' using below command.

create database RJC;

```
hive> create database RJC;
OK
Time taken: 0.178 seconds
hive> [REDACTED]
```



4. So if we want to see whether this RJC database is created or not we will use below command,

show databases;

```
hive> show databases;
OK
default
rjc
Time taken: 0.036 seconds, Fetched: 2 row(s)
hive> [REDACTED]
```

5. Now we want to check whether we have any tables inside this rjc database or not. So first we will move to this database rjc using below command,

use rjc;

```
hive> use rjc;
OK
Time taken: 0.054 seconds
hive> [REDACTED]
```

6. Now we have moved inside this rjc database. Now we will check out which are the tables available using below command,

show tables;

```
hive> show tables;
OK
Time taken: 0.106 seconds
hive> [REDACTED]
```

So as we can see from the above output it give us OK as output because there are no tables created inside this rjc database. So as there are no tables we did not get anything in the output we simply got as OK.

No will drop this 'rjc' database using below command we do not mention here cascade because there are no tables in this and so there are no data available or present inside this rjc database.

drop database rjc;

and to check whether the data base dropped or not using below command,

show databases;

```

hive> drop database rjc;
OK
Time taken: 0.264 seconds
hive> show databases;
OK
default
Time taken: 0.024 seconds, Fetched: 1 row(s)
hive> 

```

cloudera@quickstart:~ [Home - Cloudera Man...]

7. Creating a database "rjc".

create database rjc;

Now let's move to this database using below command so now all the work we will do or perform it should be done within this rjc database.

use rjc;

```

cloudera's Home
hive> create database RJC;
OK
Time taken: 0.086 seconds
hive> show databases;
OK
default
rjc
Time taken: 0.023 seconds, Fetched: 2 row(s)
hive> use rjc;

```

8. Now we will create a table inside this rjc database named as employee using below command,

create table employee(ID int, name string, salary float, age int)

Note: After this we will not put **semicolon**, When we will be loading the data from some existing csv file or maybe some other text files so we have to mention that how that data has to be loaded here. We are simply creating the schema of the table with some certain fields or attributes along with their datatypes and then I'm mentioning

row format delimited

fields terminated by ',';

```

hive> create table employee(ID int, name string, salary float, age int)
   > row format delimited
   > fields terminated by ',';
OK
Time taken: 0.425 seconds
hive> show tables;
OK
employee
Time taken: 0.043 seconds, Fetched: 1 row(s)
hive> 

```

cloudera@quickstart:~ [Home - Cloudera Man...]

row format delimited means , every record is present in one row and **fields terminated by ','** and **fields are terminated by comma**. So as soon as it encounters one comma so that means that one is the value of some field and after comma it is encountering abc so that abc is the value of some another field. By default it is a "tab" character that means fields are separated by "tab".

9. So now we will see the schema of the table using below command,

describe employee;

```

hive> describe employee;
OK
id          int
name        string
salary      float
age         int
Time taken: 0.167 seconds, Fetched: 4 row(s)
hive> 

```

cloudera@quickstart:~ [Home - Cloudera Man...]

It will give different fields of table employee along with their respective datatypes.

10. By default the internal table would store in the warehouse directory of hive. Whereas the external tables are available in the hdfs. And if we drop the internal table so then the table data and the metadata associated with that table will be deleted from the hdfs. Whereas when we drop the external table then only the metadata associated with that table will be deleted whereas the table data will be untouched by hive as it would be residing in the hdfs and it would be outside the warehouse directory of the hive. So Now we will check how the table which we will be created is internal table or external table using below command,

describe formatted employee;

By default hive creates Internal table or **Managed Table**.

11. Now we will create the external table using below command,

Note: external

Do this

create external table employee2(ID int, name string, salary float, age int)

row format delimited

fields terminated by ','

stored as textfile;

```
Time taken: 0.749 seconds
hive> create external table employee2(ID int, name string, salary float, age int)
      > row format delimited
      > fields terminated by ','
      > stored as textfile;
OK
Time taken: 0.19 seconds
hive> [REDACTED]
```

Don't this

create table employee2(ID int, name string, salary float, age int)

row format delimited

fields terminated by ','

stored as textfile;

```
hive> create table employee2(ID int, name string, salary float, age int)
      > row format delimited
      > fields terminated by ','
      > stored as textfile;
OK
Time taken: 0.158 seconds
hive> [REDACTED]
```

12. Checking the schema of the table using below command,

describe employee2;

```
hive> describe employee2;
OK
id          int
name        string
salary      float
age         int
Time taken: 0.147 seconds, Fetched: 4 row(s)
hive> [REDACTED]
```

13. So Now we will check how the table which we will be created is internal table or external table using below command,

describe formatted employee2;

Don't



It is an Managed table. (If not write external)

Do



It is an External table.

14. So whatever we have done in terminal the same thing we can see in the browser as well.

Open the browser → click on Hue

The screenshot shows the Cloudera Manager dashboard. The top navigation bar includes 'File', 'Machine', 'View', 'Input', 'Devices', 'Help', 'Applications', 'Places', 'System', and 'Logout'. The main menu has sections for 'Clusters', 'Hosts', 'Diagnostics', 'Audits', 'Charts', and 'Administration'. A sub-menu for 'Hue' is selected. On the left, there's a sidebar with a tree view of services: Hosts (1), HBase (1), HDFS (1), Hive (2), Hue (1), Impala (1), Key-Value S... (1), and Oozie (1). The central area displays three charts: 'Cluster CPU' (Host CPU Usage Average 26%), 'Cluster Disk IO' (Total Disk Bytes Received Across All Nodes 139b/s, Total Disk Bytes Written 263b/s), and 'HDFS IO' (Total Bytes Received Across All Nodes 139b/s, Total Bytes Written 263b/s). The bottom status bar shows 'cloudera@quickstart:~' and 'Home - Cloudera Manager'.

Then click on Query->Editor->Hive

The screenshot shows the Hue Editor interface. The top navigation bar includes 'File', 'Machine', 'View', 'Input', 'Devices', 'Help', 'Applications', 'Places', 'System', and 'Logout'. The main menu has sections for 'Jobs', 'Assistant', and 'Functions'. A sub-menu for 'Hive' is selected. The right sidebar lists functions: Aggregate, Analytic, Collection, Complex Type, Conditional, Date, Mathematical, Misc, String, Data Masking, Table Generating, and Type Conversion. The central area is titled 'Hive' and shows a text input field with placeholder text: 'example: SELECT * FROM tablename, or press CTRL + space'. Below it are 'Query History' and 'Saved Queries' buttons. The bottom status bar shows 'cloudera@quickstart:~' and 'Hue - Editor - Mozilla Firefox'.

Write the query in query editor. Here we are showing the databases by using below command,

show databases;

To execute -> CTRL + ENTER

It will give the list of databases which are present.

The screenshot shows the Hue Editor interface in a Mozilla Firefox browser window. The URL is `quickstart.cloudera:8888/hue/editor/editor=3`. The left sidebar shows 'Hive Databases' with 'default' and 'rjc' listed. The main area has a query editor with the text 'show databases'. Below it is a results table:

database_name
1 default
2 rjc

There are only two databases present i.e. default and rjc which we created earlier. We can also see the list of databases in left side corner. And after clicking on the database name here it is “rjc” it will give the list of all tables present inside “rjc” database.

Click on the rjc database

This screenshot is identical to the one above, but the 'rjc' database is selected in the left sidebar. The results table remains the same:

database_name
1 default
2 rjc

The screenshot shows the Hue Editor interface in Mozilla Firefox. The left sidebar shows a 'Tables' section with 'employee' and 'employee2'. The main area displays the results of the query 'show databases;'. The results table has two rows: '1 default' and '2 rjc'. On the right, a sidebar titled 'Hive' lists various functions like Aggregate, Analytic, Collection, etc.

We can also Preview the sample data. Here it is showing blank because we have not inserted anything or data inside this employee table.

The screenshot shows the Hue Editor interface in Mozilla Firefox. The left sidebar shows a 'Tables' section with 'employee' and 'employee2'. The main area displays the results of the query 'select * from rjc.employee;'. The results table shows 'Done. 0 results.' On the right, a sidebar titled 'Hive' lists various functions like Aggregate, Analytic, Collection, etc.

When we click on employee table it will give the fields of employee table.

The screenshot shows the Hue Editor interface in Mozilla Firefox. The left sidebar shows a 'Tables' section with 'employee' and 'employee2'. The main area shows the 'rjc.employee' table details. The 'Columns' tab is selected, displaying four columns: 'id' (int), 'name' (string), 'salary' (float), and 'age' (int). A message '1:0 cannot recognize input near' is visible in the editor area. On the right, a sidebar titled 'Hive' lists various functions like Aggregate, Analytic, Collection, etc.

The screenshot shows the Hue Editor interface within a Mozilla Firefox browser window. The URL in the address bar is `quickstart.cloudera:8888/hue/editor?editor=11`. The main content area displays a table named `rjc.employee2` with four columns: `Name`, `Type`, `id` (int), `name` (string), `salary` (float), and `age` (int). To the right, a sidebar titled "Assistant" lists various Hive functions under the "Hive" category, such as Aggregate, Analytic, Collection, Complex Type, Conditional, Date, Mathematical, Misc, String, Data Masking, Table Generating, and Type Conversion. A red error message box is visible, stating "1:0 cannot recognize input near".

Above if not created **external table then first drop the table**

drop table employee2;

```
Time taken: 0.749 seconds
hive> create external table employee2(ID int, name string, salary float, age int
      > row format delimited
      > fields terminated by ','
      > stored as textfile;
```

OK
Time taken: 0.19 seconds

cloudera@quickstart:~

If view the properties **describe**

describe formatted employee2;

A screenshot of the Cloudera Manager Hue interface. The top navigation bar shows 'Home - Cloudera Manager' and 'Hue - Editor'. Below it, a toolbar has icons for Home, Hue, Hadoop, Impala, and Spark. The main area is titled 'hue' and contains a 'Query' button. On the left, there's a sidebar with 'Tables' and a tree view showing the 'employee' table structure: id (int), name (string), salary (float), and age (int). To the right, a list of database objects is shown, with 'Table T1' highlighted.

15. Creating a new external table named as employee3 in the specific location using below command,

create external table employee3(ID int, name string, salary float, age int)

row format delimited

fields terminated by ','

location '/user/cloudera/vj';

It will first create 'vj' directory inside the /user/cloudera and then inside 'vj' the employee3 table get stored.

```
> location '/user/cloudera/vj';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table employee2 already exists)
hive> create external table employee3(ID int, name string, salary float, age int)
> row format delimited
> fields terminated by ','
> location '/user/cloudera/vj';
OK
Time taken: 0.174 seconds
hive>
```

16. To see the schema of the employee3 table we use below command,

describe employee3;

```
hive> describe employee3;
OK
id          int
name        string
salary      float
age         int
Time taken: 0.16 seconds, Fetched: 4 row(s)
hive>
```

17. Then switch to browser and Refresh and select the rjc database and refresh, now it will display the employee3 table along with other two tables which we have created earlier.

describe formatted employee3;

col_name	data_type	com
10. Owner:	cloudera	NULL
11. CreateTime:	Sat Mar 26 00:57:33 PDT 2022	NULL
12. LastAccessTime:	UNKNOWN	NULL
13. Protect Mode:	None	NULL
14. Retention:	0	NULL
15. Location:	https://quickstart.cloudera:8020/user/cloudera/vj	NULL
16. Table Type:	EXTERNAL_TABLE	NULL
17. Table Parameters:	NULL	NULL
18.	EXTERNAL	TRUE
19.	transient_lastDdlTime	1648
20.	NULL	NULL
21. # Storage Information	NULL	NULL

Here we will see the properties of employee3 table here we can also see the location of the table where it is stored.

18. Now move to terminal and listing out all the tables using below command;

show tables;

```
hive> show tables;
OK
employee
employee2
employee3
Time taken: 0.025 seconds, Fetched: 3 row(s)
hive>
```

19. ALTER COMMANDS

If changing the name of the employee3 table to emptable using below command,

alter table employee3 RENAME TO emptable;

20. First we will see the fields of employee3 then we will add new column as surname in employee3 using below command,

alter table employee3 add columns(surname string);

```
hive> describe employee3;
OK
id          int
name        string
salary      float
age         int
Time taken: 0.15 seconds, Fetched: 4 row(s)
hive> alter table employee3 add columns(surname string);
OK
Time taken: 0.265 seconds
hive> describe employee3;
OK
id          int
name        string
salary      float
age         int
surname    string
Time taken: 0.186 seconds, Fetched: 5 row(s)
hive> ■
```

21. Now we will change field name of the emptable to first_name using alter command,

Syntax: alter table <table_name> change <old_coln_name> <new_coln_name> string;

alter table employee3 change name first_name string;

```
hive> alter table employee3 change name first_name string;
OK
Time taken: 0.572 seconds
hive> describe employee3;
OK
id          int
first name  string
salary      float
age         int
surname    string
Time taken: 0.151 seconds, Fetched: 5 row(s)
hive> ■
```

Loading the data in the table

22. Before loading the data in the table we will first create the csv file. Now open the new terminal , using ls command list out all the directories --> change the directory to document directory --> use ls command to list all the files present inside the document folder or directory

```
cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ cd /tmp
cloudera@quickstart:~/tmp$ touch Student.csv
cloudera@quickstart:~/tmp$ ls
enterprise-deployment.json  Music  Templates
express-deployment.json    parcels  Videos
Desktop                    Pictures workspace
eclipse                     Public
cloudera@quickstart:~/tmp$ cat > Student.csv
id          int
name        string
salary      float
age         int
surname    string
cloudera@quickstart:~/tmp$ ls
first_name  string
salary      float
age         int
surname    string
Time taken: 0.151 seconds, Fetched: 5 row(s)
hive> ■
```

23. Now creating new file as Student.csv using below command,

gedit Student.csv

As soon as we hit enter it will create a Student.csv file as it was not existing earlier.

1,Rahul,Hadoop,20

2,Shiv,Java,22

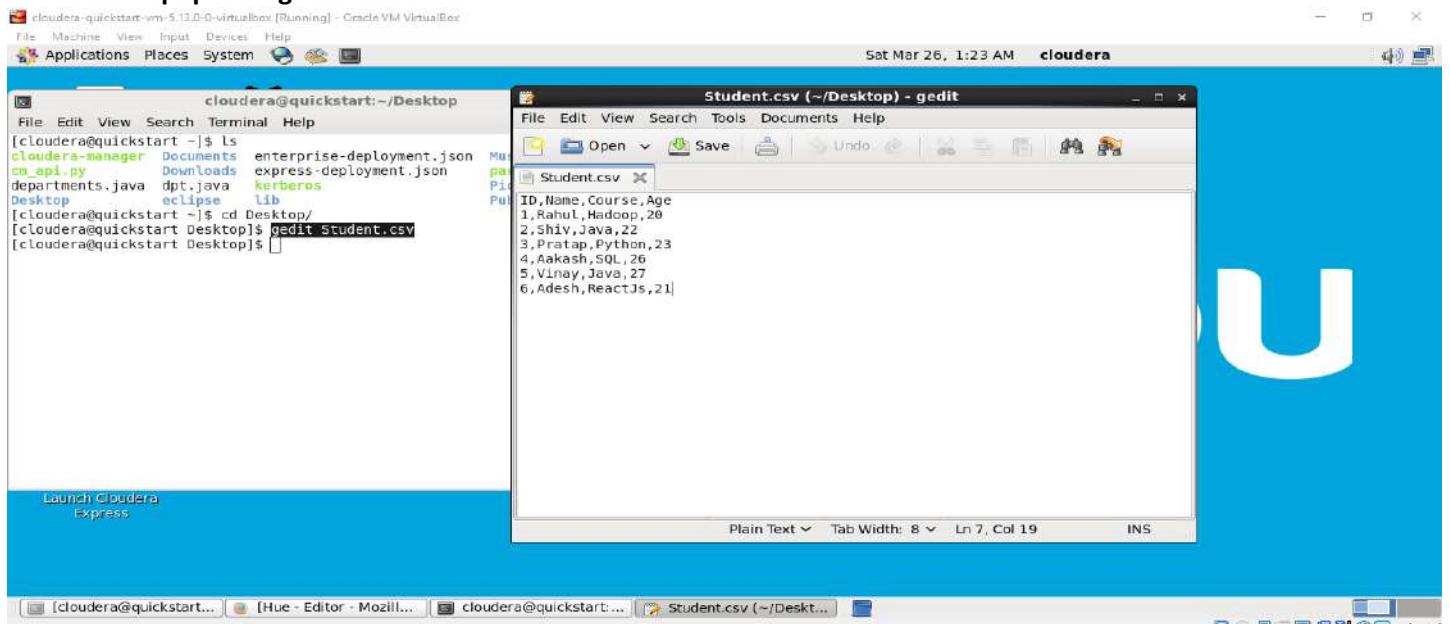
3,Pratap,Python,23

4,Aakash,SQL,26

5,Vinay,Java,27

6,Adesh,ReactJs,21

Now we are populating the Student.csv file with some data and save this file.



24. Creating a new database as rjcstudent.

create database rjcstudent;

```
hive> create database rjcstudent;
OK
Time taken: 0.109 seconds
hive> show databases;
OK
default
rjc
rjcstudent
Time taken: 0.022 seconds, Fetched: 3 row(s)
hive>
```

25. Using rjcstudent database.

use rjcstudent;

```
hive> show databases;
OK
default
rjc
rjcstudent
Time taken: 0.022 seconds, Fetched: 3 row(s)
hive> use rjcstudent;
OK
Time taken: 0.036 seconds
hive>
```

26. Creating new table student inside rjcstudent database.

```
create table student(ID int, Name string, Age int)
```

```
partitioned by(Course string)
```

```
row format delimited
```

```
fields terminated by ',';
```

```
hive> create table student(ID int, Name string, Age int)
> partitioned by(Course string)
> row format delimited
> fields terminated by ',';
```

```
OK
Time taken: 0.516 seconds
hive> ■
```

27. To see the structure or schema of the table,

```
describe student;
```

```
hive> describe student;
```

```
OK
```

id	int
name	string
age	int
course	string

# Partition Information		
# col_name	data_type	comment

course	string
--------	--------

```
Time taken: 0.243 seconds, Fetched: 9 row(s)
```

28. Loading data in the student table from Student.csv file which we have created in document directory. Here we are partitioning based on course = 'Hadoop'.

```
load data local inpath "/home/cloudera/Desktop/Student.csv" into table student
```

```
partition(Course="Hadoop");
```

```
hive> load data local inpath "/home/cloudera/Desktop/Student.csv" into table student
> partition(Course="Hadoop");
```

```
Loading data to table rjcstudent.student partition (course=Hadoop)
```

```
Partition rjcstudent.student{course=Hadoop} stats: [numFiles=1, numRows=0, totalSize=122, rawDataSize=0]
```

```
OK
```

```
Time taken: 2.081 seconds
```

```
hive> ■
```

cloudera@quickstart:~ Hue - Editor - Mozilla F... cloudera@quickstart:... ■

It is partitioning based on Hadoop.

```
select * from student;
```

```
hive> select * from student;
```

```
OK
```

NULL	Name	NULL	Hadoop
1	Rahul	NULL	Hadoop
2	Shiv	NULL	Hadoop
3	Pratap	NULL	Hadoop
4	Aakash	NULL	Hadoop
5	Vinay	NULL	Hadoop
6	Adesh	NULL	Hadoop

```
Time taken: 0.509 seconds, Fetched: 7 row(s)
```

```
hive> ■
```

cloudera@quickstart:~ Hue - Editor - Mozilla F... cloudera@quickstart:... ■

29. Now we similarly partition for course = Java and course = Python.

```
hive> load data local inpath "/home/cloudera/Desktop/Student.csv" into table student
      > partition(course="Python");
Loading data to table rjcstudent.student partition (course=Python)
Partition rjcstudent.student{course=Python} stats: [numFiles=1, numRows=0, totalSize=122, rawDataSize=0]
OK
Time taken: 1.687 seconds
hive> load data local inpath "/home/cloudera/Desktop/Student.csv" into table student
      > partition(course="Java");
Loading data to table rjcstudent.student partition (course=Java)
Partition rjcstudent.student{course=Java} stats: [numFiles=1, numRows=0, totalSize=122, rawDataSize=0]
OK
Time taken: 0.762 seconds
hive>
```

cloudera@quickstart:~/Desktop

30. Now go to browser refresh the page and select database as rjcstudent and click in

31. Now dropping the table student and creating the student table again normally (i.e. without partitioning).

drop table student;

```
create table student(ID int, Name string, Course string, Age int)
```

row format delimited

fields terminated by ','

tblproperties("skip.header.line.count"="1");

```
hive> create table student(ID int, Name string, Course string, Age int)
      > row format delimited
      > fields terminated by ','
      > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.145 seconds
hive>
```

cloudera@quickstart:~/Desktop

32. Loading data in the student table from Student.csv file which present inside

/home/cloudera/Desktop directory.

load data local inpath "/home/cloudera/Desktop/Student.csv" into table student;

```
hive> load data local inpath "/home/cloudera/Desktop/Student.csv" into table student;
Loading data to table rjcstudent.student
Table rjcstudent.student stats: [numFiles=1, totalSize=122]
OK
Time taken: 0.482 seconds
hive>
```

Student.csv (~/Desktop) - gedit

select * from student;

```
hive> select * from student;
OK
1    Rahul  Hadoop  20
2    Shiv   Java     22
3    Pratap Python   23
4    Aakash SQL     26
5    Vinay  Java     27
6    Adesh  ReactJS 21
Time taken: 0.14 seconds, Fetched: 6 row(s)
hive>
```

33. Now creating employee.csv file.

gedit Employee.csv

```
cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ ls
cloudera-manager  Documents  enterprise-deployment.json
cm_api.py         Downloads  express-deployment.json
departments.java  dpt.java   kerberos
Desktop          eclipse   lib
cloudera@quickstart:~$ cd Desktop/
[cloudera@quickstart Desktop]$ gedit Student.csv
[cloudera@quickstart Desktop]$ gedit Employee.csv
[cloudera@quickstart Desktop]$
```

ID	Name	Dept	Yoj	Salary
1	Rahul	IT	2012	20000
2	Shiv	HR	2012	23000
3	Pratap	Sales	2013	25000
4	Aakash	HR	2014	22000

```
3    Pratap Python  23
4    Aakash SQL     26
5    Vinay  Java     27
6    Adesh  ReactJS 21
Time taken: 0.14 seconds, Fetched: 6 row(s)
hive>
```

Querying :

34. Creating new database for performing querying operations.

create database hiveql;

```
hive> create database hiveql;
```

OK

Time taken: 0.082 seconds

```
hive> [cloudera@quickstart:~/Desktop]
```

cloudera@quickstart:~/Desktop

...

35. Using database hiveql and creating table employee inside the hiveql database.

create table employee(ID int, Name string, Department string, YOJ int, Salary float)

row format delimited

fields terminated by ','

tblproperties("skip.header.line.count"="1");

```
hive> create table employee(ID int, Name string, Department string, YOJ int, Salary float)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count"="1");
```

OK

Time taken: 0.113 seconds

```
hive> [Employee.csv (~/Desktop) - gedit]
```

Employee.csv (~/Desktop) - gedit

...

Now we will see the schema of employee table using below command,

describe employee;

```
hive> describe employee;
OK
id          int
name        string
department  string
yoj         int
salary      float
Time taken: 0.184 seconds, Fetched: 5 row(s)
hive> [cloudera@quickstart:~/Desktop]
```

cloudera@quickstart:~/Desktop

...

36. Loading the data into employee table from employee.csv file which we have created

earlier and it is present in /home/cloudera/Documents directory.

load data local inpath "/home/cloudera/Desktop/Employee.csv" into table employee;

```
hive> load data local inpath "/home/cloudera/Desktop/Employee.csv" into table employee;
Loading data to table rjcstudent.employee
Table rjcstudent.employee stats: [numFiles=1, totalSize=116]
OK
Time taken: 0.609 seconds
hive> [Employee.csv (~/Desktop) - gedit]
```

Employee.csv (~/Desktop) - gedit

...

Displaying the table using below command,

select * from employee;

```
hive> select * from employee;
OK
1    Rahul   IT     2012   20000.0
2    Shiv    HR     2012   23000.0
3    Pratap  Sales  2013   25000.0
4    Aakash  HR     2014   22000.0
Time taken: 0.141 seconds, Fetched: 4 row(s)
hive> [Employee.csv (~/Desktop) - gedit]
```

Employee.csv (~/Desktop) - gedit

...

Now we Perform some operations on this employee table.

37. We are printing or displaying the rows or records of employees whose salary is greater than and equal to 25000.

select * from employee where salary >=25000;

```
hive> select * from employee where salary >=25000;
Query ID = cloudera_20220326025858_4b236ef-bd75-4923-9285-d373dbb34163
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1648262911818_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0005/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-03-26 02:58:54,227 Stage-1 map = 0%, reduce = 0%
2022-03-26 02:59:16,852 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.44 sec
MapReduce Total cumulative CPU time: 4 seconds 440 msec
Ended Job = job_1648262911818_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.44 sec HDFS Read: 4735 HDFS Write: 28 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 440 msec
OK
3 Pratap Sales 2013 25000.0
Time taken: 55.82 seconds, Fetched: 1 row(s)
hive> [No Items in Trash]
```

38. Now we are printing or displaying the rows or records of employees whose salary is less than 25000.

select * from employee where salary <25000;

```
hive> select * from employee where salary <25000;
Query ID = cloudera_20220326030000_38afaa20-0321-413f-b867-18a4e55a7abb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1648262911818_0006, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0006/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-03-26 03:00:44,357 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:01:09,379 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.62 sec
MapReduce Total cumulative CPU time: 4 seconds 620 msec
Ended Job = job_1648262911818_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.62 sec HDFS Read: 4797 HDFS Write: 72 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 620 msec
OK
1 Rahul IT 2012 20000.0
2 Shiv HR 2012 23000.0
4 Aakash HR 2014 22000.0
Time taken: 58.944 seconds, Fetched: 3 row(s)
hive> [No Items in Trash]
```

Aggregating

39. Arithmetic operations:

We are **adding** 2000 in existing salary and displaying specific columns using below command,

select ID, Name, salary + 2000 from employee;

```
hive> select ID, Name, salary + 2000 from employee;
Query ID = cloudera_20220326030303_332565b8-4680-4457-aaeb-ad0c82963858
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1648262911818_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-03-26 03:52,983 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:04:17,932 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.43 sec
MapReduce Total cumulative CPU time: 4 seconds 430 msec
Ended Job = job_1648262911818_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.43 sec HDFS Read: 4881 HDFS Write: 65 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 430 msec
OK
1 Rahul 22000.0
2 Shiv 25000.0
3 Pratap 27000.0
4 Aakash 24000.0
Time taken: 64.708 seconds, Fetched: 4 row(s)
hive> [No Items in Trash]
```

40. Displaying the **maximum salary** using below command,

select max(salary) from employee;

```

hive> select max(salary) from employee;
Query ID = cloudera_20220326030505_1f91277d-6abe-467d-8a66-c8e7e234dde1
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:06:17,722 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:06:37,510 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.79 sec
2022-03-26 03:06:58,636 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.61 sec
MapReduce Total cumulative CPU time: 6 seconds 610 msec
Ended Job = job_1648262911818_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.61 sec HDFS Read: 8055 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 610 msec
OK
25000.0
Time taken: 71.963 seconds, Fetched: 1 row(s)
hive> [cloudera@quickstart:~] [Hue - Editor - Mozilla Firefox] [cloudera@quickstart...] [Employee.csv (-/De...]

```

Here we can see that 25000 is maximum salary.

41. Displaying the **minimum salary** using below command,

select min(salary) from employee;

```

hive> select min(salary) from employee;
Query ID = cloudera_20220326030707_6bbaa9ee-0051-4f2d-956a-197b44b53a46
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:08:30,966 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:08:51,622 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.84 sec
2022-03-26 03:09:13,784 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.81 sec
MapReduce Total cumulative CPU time: 6 seconds 810 msec
Ended Job = job_1648262911818_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.81 sec HDFS Read: 8055 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 810 msec
OK
20000.0
Time taken: 77.906 seconds, Fetched: 1 row(s)
hive> [cloudera@quickstart:~] [Hue - Editor - Mozilla Firefox] [cloudera@quickstart...] [Employee.csv (-/De...]

```

Here we can see that 20000 is minimum salary.

42. Now finding the **square root of salary** using below command,

select ID, Name, sqrt(salary) from employee;

```

hive> select ID, Name, sqrt(salary) from employee;
Query ID = cloudera_20220326031010_0793d885-79ca-4f9c-819d-f2d7598ed1f7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1648262911818_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0010
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-03-26 03:10:56,356 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:11:17,978 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.65 sec
MapReduce Total cumulative CPU time: 4 seconds 650 msec
Ended Job = job_1648262911818_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.65 sec HDFS Read: 4545 HDFS Write: 108 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 650 msec
OK
1      Rahul    141.4213562373095
2      Shiv     151.65750880103102
3      Pratap   158.11388300841898
4      Aakash   148.32396974191326
Time taken: 51.912 seconds, Fetched: 4 row(s)
hive> [cloudera@quickstart:~] [Hue - Editor - Mozilla Firefox] [cloudera@quickstart...] [Employee.csv (-/De...]

```

43. Now we are showing the name column in **upper case** using below command,

select ID, upper(name) from employee;

```

hive> select ID, upper(name) from employee;
Query ID = cloudera_20220326031212_8d0193e4-eea2-4fac-bdca-39b329e488b8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1648262911818_0011, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0011/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-03-26 03:12:32,143 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:12:53,889 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.75 sec
MapReduce Total cumulative CPU time: 4 seconds 750 msec
Ended Job = job_1648262911818_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.75 sec HDFS Read: 4331 HDFS Write: 33 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 750 msec
OK
1      RAHUL
2      SHIV
3      PRATAP
4      AAKASH
Time taken: 52.859 seconds, Fetched: 4 row(s)
hive> [Hue - Editor - Mozilla Firefox]

```

44. Now we are showing the name column in **lower case** using below command,

select ID, lower(name) from employee;

```
hive> select ID, lower(name) from employee;
Query ID = cloudera_20220326031313_fbbd273b-decf-40ac-be8f-06447f24bd5e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job 1648262911818_0012, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job 1648262911818_0012
Hadoop job information for Stage-1: number of mappers: 1, number of reducers: 0
2022-03-26 03:14:14,234 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:14:39,884 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.53 sec
MapReduce Total cumulative CPU time: 4 seconds 530 msec
Ended Job = job 1648262911818_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 4.53 sec HDFS Read: 4331 HDFS Write: 33 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 530 msec
OK
1      rahul
2      shiv
3      pratap
4      aakash
Time taken: 53.377 seconds, Fetched: 4 row(s)
hive> ■
```

45. Creating another **employee2.csv** file with the same data as **employee.csv** but adding one more column as **Country** to it.

46. Creating a new table as **empgroup**.

create table empgroup(ID int, Name string, Department string, YOJ int, Salary float, Country string)

row format delimited

fields terminated by ','

tblproperties("skip.header.line.count"="1");

```
hive> create table empgroup(ID int, Name string, Department string, YOJ int, Salary float, Country string)
> row format delimited
> fields terminated by ','
> tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.206 seconds
hive> ■
```

47. Loading the data into **empgroup** table from **employee2.csv** file which we have created and it is present in **/home/cloudera/Desktop** directory.

load data local inpath "/home/cloudera/Desktop/Employee2.csv" into table empgroup;

```
hive> load data local inpath "/home/cloudera/Desktop/Employee2.csv" into table empgroup;
Loading data to table rjcstudent.empgroup
Table rjcstudent.empgroup stats: [numFiles=1, totalSize=138]
OK
Time taken: 0.868 seconds
hive> ■
```

Displaying the table using below command,

Select * from empgroup;

```
hive> select * from empgroup;
OK
1      Rahul    IT        2012      20000.0  IND
2      Shiv     HR        2012      23000.0  UK
3      Pratap   Sales     2013      25000.0  UK
4      Aakash   HR        2014      22000.0  USA
Time taken: 0.129 seconds, Fetched: 4 row(s)
hive> ■
```

cloudera@quickstart:~/Desktop

48. Groupby clause

Now we display the total sum of salary of employees country wise using below command,

select Country, sum(Salary) from empgroup group by Country;

```
hive> Select Country, sum(Salary) from empgroup group by Country;
Query ID = cloudera_20220326032121_506b17ce-b13e-4ae0-98c7-d4ff365af9ee
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:22:03,266 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:22:25,810 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.87 sec
2022-03-26 03:22:43,484 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.87 sec
MapReduce Total cumulative CPU time: 6 seconds 870 msec
Ended Job = job_1648262911818_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.87 sec HDFS Read: 8629 HDFS Write: 35 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 870 msec
OK
IND      20000.0
UK      48000.0
USA      22000.0
Time taken: 69.464 seconds, Fetched: 3 row(s)
hive> ■
```

Groupby clause along with the having clause

Taking the total sum of salary countrywise using groupby clause and from that selecting or displaying those country whose **total sum of salary > 30000** using having clause.

select Country, sum(Salary) from empgroup group by Country having sum(Salary)>30000;

```
hive> select Country, sum(Salary) from empgroup group by Country having sum(Salary)>30000;
Query ID = cloudera_20220326032525_73a66c90-d2d6-4322-8a24-ab2fee5fe74e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0015, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0015/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0015
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:25:43,987 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:26:03,370 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.31 sec
2022-03-26 03:26:27,298 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.62 sec
MapReduce Total cumulative CPU time: 8 seconds 620 msec
Ended Job = job_1648262911818_0015
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.62 sec HDFS Read: 9138 HDFS Write: 11 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 620 msec
OK
UK      48000.0
Time taken: 69.319 seconds, Fetched: 1 row(s)
hive> ■
```

Sorting : Order by

49. Now we are displaying the table by sorting the salary in **descending order**.

select * from empgroup order by Salary desc;

```
hive> select * from empgroup order by Salary desc;
Query ID = cloudera_20220326032828_d75db91e-d98a-4a2a-b3ac-4f8ddad78660
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0016, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0016/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:28:41,036 Stage-1 map = 0%, reduce = 0%
2022-03-26 03:28:59,643 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.3 sec
2022-03-26 03:29:23,519 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.09 sec
MapReduce Total cumulative CPU time: 7 seconds 96 msec
Ended Job = job_1648262911818_0016
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.09 sec HDFS Read: 8624 HDFS Write: 114 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 90 msec
OK
3    Pratap  Sales   2013    25000.0 UK
2    Shiv    HR     2012    23000.0 UK
4    Aakash  HR     2014    22000.0 USA
1    Rahul   IT     2012    20000.0 IND
Time taken: 70.666 seconds, Fetched: 4 row(s)
hive> ■
```

We can see that number of reducers: 1 for the order by.

50. Now we are displaying the table by **sorting** the salary in descending order.

select * from empgroup sort by Salary desc;

```
hive> select * from empgroup sort by Salary desc;
Query ID = cloudera_20220326033030_b0aa6056-13eb-48bb-aa8c-c1e56c95caec
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1648262911818_0017, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1648262911818_0017/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1648262911818_0017
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-03-26 03:30:55,515 Stage-1 map = 0%,  reduce = 0%
2022-03-26 03:31:20,041 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.83 sec
2022-03-26 03:31:43,167 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.78 sec
MapReduce Total cumulative CPU time: 6 seconds 780 msec
Ended Job = job_1648262911818_0017
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 6.78 sec  HDFS Read: 8624 HDFS Write: 114 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 780 msec
OK
3      Pratap  Sales   2013    25000.0 UK
2      Shiv     HR      2012    23000.0 UK
4      Aakash   HR      2014    22000.0 USA
1      Rahul    IT      2012    20000.0 IND
Time taken: 79.267 seconds, Fetched: 4 row(s)
hive> ■
```

Now we can see the similar result as we got from order by and sort by so what is the difference between the two is that it depends on number of reducers in order by we got number of reducers is 1 and by using sort by here is also we got number of reducers is 1 so the difference between the two is that Order by will guarantee the total order in the output whereas sort by will only guarantee the ordering of the rows within the reducer.

Order by gives us completely sorted result whereas sort by give us partially sorted result.

quit

```
hive> quit
> ;
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
[cloudera@quickstart ~]$ ■
```

cloudera@quickstart:~/Desktop

A screenshot of a terminal window titled 'cloudera@quickstart:~'. The window shows the command 'quit' being entered and its execution results. The results include two 'WARN' messages about the SLF4JLogFactory release method and a final prompt '[cloudera@quickstart ~]\$ ■'.

Aim: To implement Word Count problem using Pig

What is Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

Features of Apache Pig

The various uses of Pig technology.

1) Ease of programming

Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.

2) Optimization opportunities

It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

3) Extensibility

A user-defined function is written in which the user can write their logic to execute over the data set.

4) Flexible

It can easily handle structured as well as unstructured data.

5) In-built operators

It contains various type of operators such as sort, filter and joins.

Advantages of Apache Pig

- **Less code** - The Pig consumes less line of code to perform any operation.
 - **Reusability** - The Pig code is flexible enough to reuse again.
 - **Nested data types** - The Pig provides a useful concept of nested data types like tuple, bag, and

Before the perform the practical first perform 3 steps of the given following

1. sudo /home/cloudera/cloudera-manager --force --express

```
cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System ⑤ Fri Mar 25, 1:15 AM cloudera
cloudera@quickstart:~ [QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Starting Cloudera Manager agent...
[QuickStart] Configuring deployment...
Submitted jobs: 41
[QuickStart] Deploying client configuration...
Submitted jobs: 42
[QuickStart] Starting Cloudera Management Service...
Submitted jobs: 50
[QuickStart] Enabling Cloudera Manager daemons on boot...

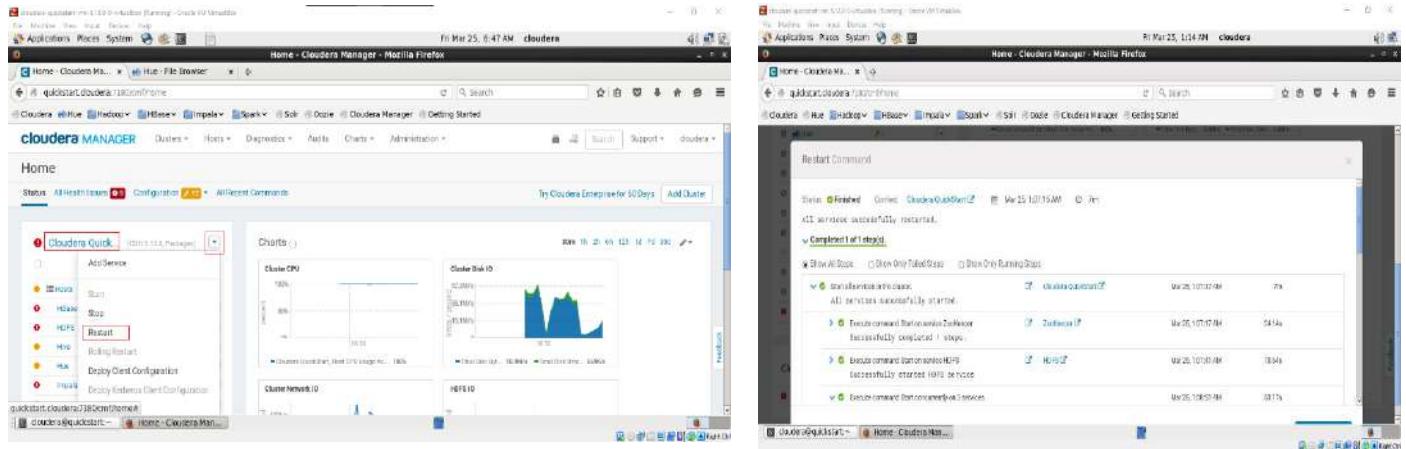
Success! You can now log into Cloudera Manager from the QuickStart VM's browser:
http://quickstart.cloudera:7180

Username: cloudera
Password: cloudera

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

22/03/25 01:04:22 WARN ipc.Client: Failed to connect to server: quickstart.cloudera/10.0.2.15:8620: try once and fail.
java.net.ConnectException: Connection refused
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:739)
    at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:266)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)
    at org.apache.hadoop.ipc.ClientsConnection.setupConnection(Client.java:648)
    at org.apache.hadoop.ipc.ClientsConnection.setupIOstreams(Client.java:744)
    at org.apache.hadoop.ipc.Client$Connection.access$300(Client.java:396)
```

2. Start all services

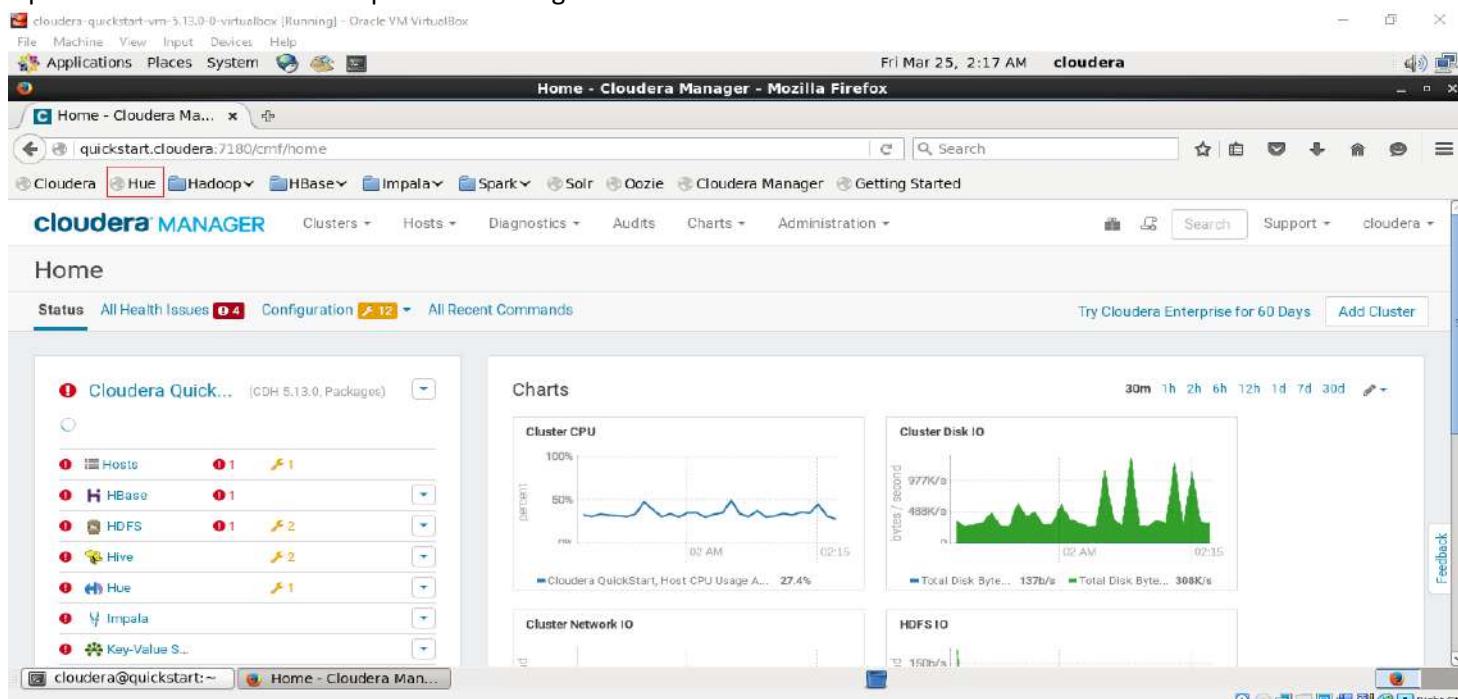


3. sudo -u hdfs hadoop dfsadmin -safemode leave

```
cloudera@quickstart:~$ hdfs dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$
```

1. Open the browser. And then open Hue and login.



Name: Pavan Yadav

Practical 8

Hue - Welcome to Hue - Mozilla Firefox

Pri Mar 25, 6:34 AM cloudera

Home - Cloudera Ma... Hue - Welcome to Hue

quickstart.cloudera:8888/accounts/login/?next=/hue

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

HUE

Query. Explore. Repeat.

cloudera

password

Sign In

Hue and the Hue logo are trademarks of Cloudera, Inc.

2. Now open the directory /user/cloudera

Type /hue/filebrowser

Hue - File Browser - Mozilla Firefox

Fri Mar 25, 6:39 AM cloudera

Home - Cloudera Ma... Hue - File Browser

quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

HUE

File Browser

Sources: Impala, Hive

Search for file name: Actions: Move to trash

Upload New

File Directory

Name, Size, User, Group, Permissions, Date

Name	Size	User	Group	Permissions	Date
..		hdbs	supergroup	drwxr-xr-x	March 17, 2022 07:20 AM
.		cloudera	cloudera	drwxr-xr-x	March 17, 2022 07:18 AM
.Trash		cloudera	cloudera	drwx-----	March 18, 2022 08:01 AM

Show 45 of 1 items

Page 1 of 1

3. Now we are creating the directory as **Training** inside /user/cloudera

Hue - File Browser - Mozilla Firefox

Fri Mar 25, 2:20 AM cloudera

Home - Cloudera Ma... Hue - File Browser

quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

HUE

File Browser

default

Tables

The database has no tables

Search for file name: Actions: Move to trash

Upload New

File Directory

Name, Size, User, Group, Permissions, Date

Name	Size	User	Group	Permissions	Date
..		hdbs	supergroup	drwxr-xr-x	October 23, 2017 09:17 AM
.		cloudera	cloudera	drwxr-xr-x	October 23, 2017 09:14 AM
Training					

Show 45 of 0 items

Page 1 of 1

Name: Pavan Yadav

Practical 8

Create Directory

Directory Name: Training

Name Size User Group Permissions Date

- hdfs
- cloudera
- .Trash

Search for file name

Name Size User Group Permissions Date

- hdfs
- cloudera
- .Trash
- Training**

4. After creating Training directory now creating the **pig** directory inside /user/cloudera/Training

File

Directory

Name Size User Group Permissions Date

- cloudera
- cloudera

Create Directory

Directory Name: pig

Name	Size	User	Group	Permissions	Date
pig		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:41 AM
pig		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:41 AM

pig directory has been created inside /user/cloudera/Training

Search data and saved documents...

Home / user / cloudera / Training

Name	Size	User	Group	Permissions	Date
pig		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:41 AM
.		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
..		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM

5. Creating input.txt file inside /usr/cloudera/training/pig directory

File Browser

Home / user / cloudera / Training / pig

Name	Size	User	Group	Permissions	Date
input.txt		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
..		cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM

Create File

File Name: input.txt

Name	Size	User	Group	Permissions	Date
input.txt		cloudera	cloudera	drwxr-x-x	March 25, 2022 06:43 AM
input2.txt		cloudera	cloudera	drwxr-x-x	March 25, 2022 06:43 AM

Note: If getting this type of error than start all services again

Create File

File Name: input.txt

IOException: Failed to find datanode, suggest to check cluster health. excludeDatanodes=null (error 403)

Refresh the Page

File Browser

Search for filename:

Name	Size	User	Group	Permissions	Date
input.txt		cloudera	cloudera	drwxr-x-x	March 25, 2022 06:43 AM
input2.txt		cloudera	cloudera	drwxr-x-x	March 25, 2022 06:43 AM

The screenshot shows the Hue File Browser interface. The current path is `/user/cloudera/Training/pig`. The table below shows the contents:

Name	User	Group	Permissions	Date
input	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
input.	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM

A modal window titled "Create File" is displayed, showing the "File Name" field set to "input.txt".

The screenshot shows the Hue File Browser interface. The current path is `/user/cloudera/Training/pig`. The table below shows the contents, including the newly created "input.txt" file.

Name	User	Group	Permissions	Date
input	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
input.	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
input.txt	cloudera	cloudera	-rw-r--r--	March 25, 2022 07:17 AM

6. Adding some contents to this input.txt file.

The screenshot shows the Hue File Browser interface. The current path is `/user/cloudera/Training/pig`. The table below shows the contents, with the "input.txt" file selected.

Name	User	Group	Permissions	Date
input	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
input.	cloudera	cloudera	drwxr-xr-x	March 25, 2022 06:43 AM
input.txt	cloudera	cloudera	-rw-r--r--	March 25, 2022 07:17 AM

Click on **Edit file** option

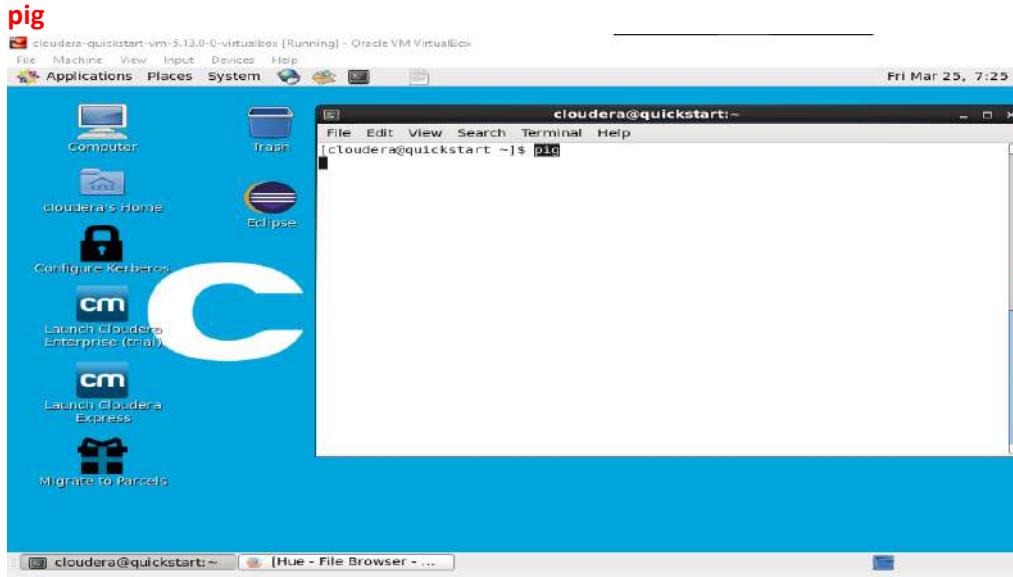
The screenshot shows the Hue File Browser interface. The URL in the address bar is `quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera/Training/pig/input.txt`. The left sidebar shows 'Sources' with 'Impala' and 'Hive' listed. The main area is titled 'File Browser' and shows a single file entry: '/user/cloudera/Training/pig/input.txt'. Below it, a message says 'The current file is empty.' There are buttons for 'Edit file', 'Download', 'View file location', and 'Refresh'.

Save the input.txt file

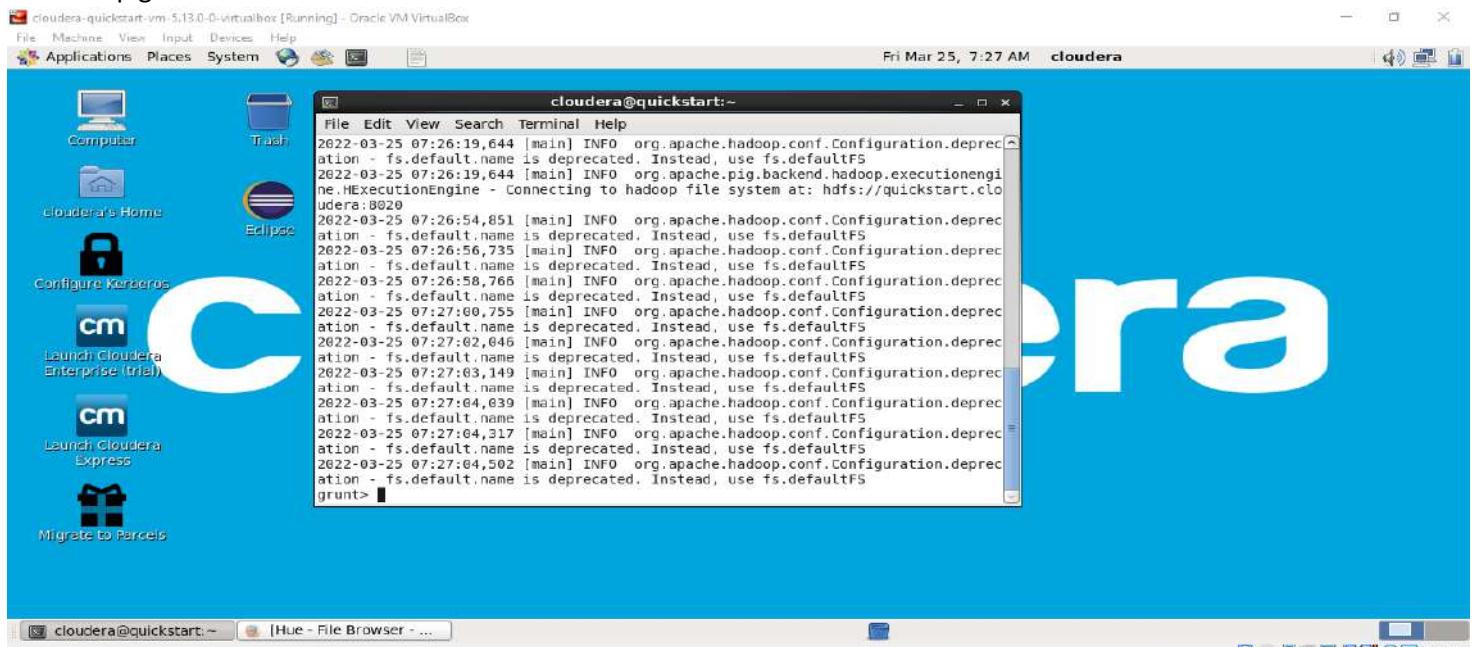
The screenshot shows the Hue File Browser interface. The URL in the address bar is `quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera/Training/pig/input.txt`. The left sidebar shows 'Sources' with 'Impala' and 'Hive' listed. The main area is titled 'File Browser' and shows the file content: 'We are studying Big Data Technology
We have covered the Hadoop ecosystem
We have executed Wordcount using MapReduce
We have also implemented CRUD operation using MongoDB'. Below the content, there are 'Save' and 'Save as' buttons.

The screenshot shows the Hue File Browser interface. The URL in the address bar is `quickstart.cloudera:8888/hue/filebrowser/view=/user/cloudera/Training/pig/input.txt`. The left sidebar shows 'Sources' with 'Impala' and 'Hive' listed. The main area is titled 'File Browser' and shows the same file content as the previous screenshot: 'We are studying Big Data Technology
We have covered the Hadoop ecosystem
We have executed Wordcount using MapReduce
We have also implemented CRUD operation using MongoDB'. The content is displayed in a scrollable text area.

7. Now Open the terminal. And start **Pig** by typing pig on terminal.

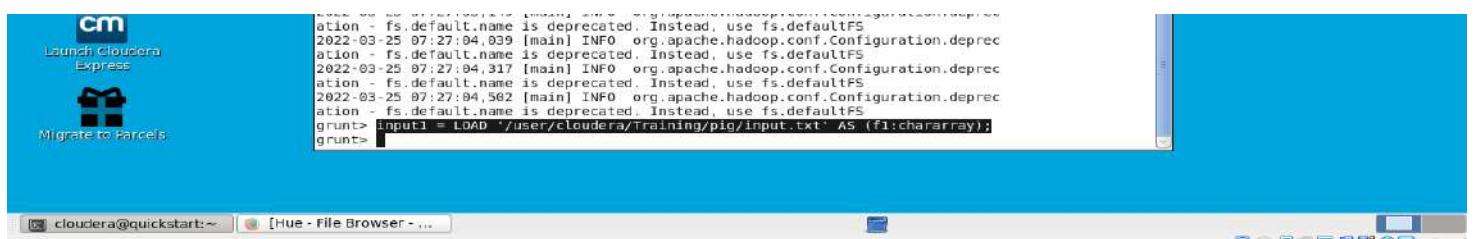


Now the pig started



8. Now we have to load that input file where ever it is stored. By typing the command

input1 = LOAD '/user/cloudera/Training/pig/input.txt' AS (f1:chararray);



9. Now we are dumping the data. It will done the mapreduce task.

DUMP input1

```
grunt> DUMP _input1
2022-03-25 07:30:44,873 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2022-03-25 07:30:45,102 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer
- {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitListInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]}
2022-03-25 07:30:45,772 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCCompiler - File concatenation threshold: 100 optimistic? false
2022-03-25 07:30:46,960 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - Map plan size before optimization: 1
2022-03-25 07:30:46,960 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - Map plan size after optimization: 1
2022-03-25 07:30:46,928 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to Resource Manager at cluster@10.8.2.15:8082
2022-03-25 07:30:48,027 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2022-03-25 07:30:48,341 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.reduce.markreset.buffer.percent is deprecated.. Instead, use mapreduce.reduce.markreset.buffer.percent
2022-03-25 07:30:48,341 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2022-03-25 07:30:48,341 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.out.put.compress is deprecated.. Instead, use mapreduce.output.fileoutputformat.compress
2022-03-25 07:30:55,142 [main] INFO org.apache.pio.backend.hadoop.executionengine.mapReduceLayer
```

```
[root] - Oracle VM VirtualBox      Fri Mar 25, 7:37 AM cloudera
File Edit View Search Terminal Help

cloudera@quickstart:~$ cat /tmp/partition.log
Counters:
Total records written : 4
Total bytes written : 192
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1648216557393_0001

2022-03-25 07:37:17,932 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2022-03-25 07:37:17,952 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-03-25 07:37:17,961 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key Ipig.schematuple was not found and will not generate code.
2022-03-25 07:37:18,099 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2022-03-25 07:37:18,101 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(We are studying Big Data Technology)
(We have covered the Hadoop ecosystem)
(We have executed Wordcount using MapReduce)
(We have also implemented CRUD operation using MongoDB)
grants...,
```

10. wordsInEachLine = FOREACH input1 GENERATE flatten(TOKENIZE(f1)) as word;

```
2022-03-25 07:37:17,934 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2022-03-25 07:37:17,952 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-03-25 07:37:17,961 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set., will not generate code.
2022-03-25 07:37:18,099 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2022-03-25 07:37:18,101 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(We are studying Big Data Technology)
(We have covered the Hadoop ecosystem)
(We have executed Wordcount using MapReduce)
(We have also implemented CRUD operation using MongoDB)
grunt> wordsInEachLine = FOREACH input1 GENERATE flatten(TOKENIZE(f1)) as word;
```

11. DUMP wordsInEachLine;

```
grunt> wordsInEachLine = FOREACH input1 GENERATE flatten(TOKENIZE(f1)) as word;
grunt> DUMP wordsInEachLine;
2022-03-25 07:41:25,391 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used
in the script: UNKNOWN
2022-03-25 07:41:25,408 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer
- [RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstrParal
lSetter, ImplicitSplitInsertter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach,
NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastIns
ertter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]}
2022-03-25 07:41:25,499 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.
MRCompiler - File concatenation threshold: 100 optimistic? false
2022-03-25 07:41:25,507 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.
MultiQueryOptimizer - MR plan size before optimization: 1
2022-03-25 07:41:25,507 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.
MultiQueryOptimizer - MR plan size after optimization: 1
2022-03-25 07:41:25,762 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to Resourc
eManager at quickstart.cloud-era/10.0.2.15:8082
2022-03-25 07:41:25,825 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settin
gs are added to the job
2022-03-25 07:41:26,032 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.
JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
```

```

cloudera@quickstart:~$ input paths to process : 1
2022-03-25 07:44:23,576 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapReduceUtil
- Total input paths to process : 1
(We)
(are)
(studying)
(Big)
(Data)
(Technology)
(We)
(have)
(covered)
(the)
(Hadoop)
(ecosystem)
(We)
(have)
(executed)
(Wordcount)
(using)
(MapReduce)
(We)
(have)
(also)
(implemented)
(CRUD)
(operation)
(using)
(MongoDB)
grunt>

```

12. Now grouping the words present in each line.

groupedWords = group wordsInEachLine by word;

```

cloudera@quickstart:~$ groupedWords = group wordsInEachLine by word;
2022-03-25 07:44:23,576 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapReduceUtil
- Total input paths to process : 1
(We)
(are)
(studying)
(Big)
(Data)
(Technology)
(We)
(have)
(covered)
(the)
(Hadoop)
(ecosystem)
(We)
(have)
(executed)
(Wordcount)
(using)
(MapReduce)
(We)
(have)
(also)
(implemented)
(CRUD)
(operation)
(using)
(MongoDB)
grunt> groupedWords = group wordsInEachLine by word;
grunt>

```

13. DUMP groupedWords;

```

cloudera@quickstart:~$ groupedWords = group wordsInEachLine by word;
cloudera@quickstart:~$ DUMP groupedWords;
2022-03-25 07:47:36,142 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in
the script: GROUP BY
2022-03-25 07:47:36,146 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer -
[RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSet
ter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeForEach, NewPartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, PartitionFilterOptimizer]]
2022-03-25 07:47:36,275 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRC
ompiler - File concatenation threshold: 100 optimistic? false
2022-03-25 07:47:36,312 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Mul
tiQueryOptimizer - MR plan size before optimization: 1
2022-03-25 07:47:36,312 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Mul
tiQueryOptimizer - MR plan size after optimization: 1
2022-03-25 07:47:36,448 [main] INFO org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceMa
nager at quickstart.cloudera/10.0.2.15:8032
2022-03-25 07:47:36,452 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settings
are added to the job
2022-03-25 07:47:36,500 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Job
ControlCompiler - mapred.job.reduce.marksreset.buffer.percent is not set, set to default 6.3
2022-03-25 07:47:36,502 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Job
ControlCompiler - Reduce phase detected, estimating # of required reducers.
2022-03-25 07:47:36,511 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Job
ControlCompiler - Using reducer estimator: org.apache.pig.backend.hadoop.executionengine.mapReduceLay
er.InputSizeReducerEstimator
2022-03-25 07:47:36,548 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Inp
utSizeReducerEstimator - BytesPerReducer=1000000000 maxReducers=999 totalInputFileSize=169
2022-03-25 07:47:36,548 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Job
ControlCompiler - Setting Parallelism to 1

```

A screenshot of a Linux desktop environment titled "cloudera@quickstart:~". The desktop has a blue theme with a large white 'a' logo. A terminal window is open with the following command and output:

```
File Edit View Search Terminal Help  
ReduceLauncher - Success!  
2022-03-25 07:52:04,472 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS.  
2022-03-25 07:52:04,472 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set, so will not generate code.  
2022-03-25 07:52:04,509 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1  
2022-03-25 07:52:04,509 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapReduceUtil - Total input paths to process : 1  
grunt> (using,{{(using)}},{{(using)}},{{(using)}},{{(using)}},{{(using)}})  
(Big,(Big))  
(are,{{(are)}})  
(CRUD,{{(CRUD)}})  
(Data,{{(Data)}})  
(at,{{(at)})}  
(have,{{(have)}},{{(have)}},{{(have)}},{{(have)}},{{(have)}})  
(using,{{(using)}},{{(using)}},{{(using)}},{{(using)}},{{(using)}})  
(Hadoop,{{(Hadoop)}},{{(Hadoop)}},{{(Hadoop)}},{{(Hadoop)}},{{(Hadoop)}})  
(Map,{{(Map)}},{{(Map)}},{{(Map)}},{{(Map)}},{{(Map)}})  
(Covered,{{(Covered)}},{{(Covered)}},{{(Covered)}},{{(Covered)}},{{(Covered)}})  
(executed,{{(executed)}},{{(executed)}},{{(executed)}},{{(executed)}},{{(executed)}})  
(is,{{(is)}},{{(is)}},{{(is)}},{{(is)}},{{(is)}})  
(MapReduce,{{(MapReduce)}},{{(MapReduce)}},{{(MapReduce)}},{{(MapReduce)}},{{(MapReduce)}})  
(Wordcount,{{(Wordcount)}},{{(Wordcount)}},{{(Wordcount)}},{{(Wordcount)}},{{(Wordcount)}})  
(recyclable,{{(recyclable)}},{{(recyclable)}},{{(recyclable)}},{{(recyclable)}},{{(recyclable)}})  
(operation,{{(operation)}},{{(operation)}},{{(operation)}},{{(operation)}},{{(operation)}})  
(technology,{{(technology)}},{{(technology)}},{{(technology)}},{{(technology)}},{{(technology)}})  
(implemented,{{(implemented)}},{{(implemented)}},{{(implemented)}},{{(implemented)}},{{(implemented)}})  
grunt>
```

14. Now we count those words. For each group we count words in each line.

```
countedWords = FOREACH groupedWords GENERATE group, COUNT(wordsInEachLine);
```

```
Migrate to Parcels <line 4, column 45> invalid field projection. Projected Field [group] does not exist in schema: group :chararray,wordsInEachLine:bag(:tuple(word:chararray)).  
Details at logfile: /home/cloudera/pig_1648218365125.log  
grunt> countedWords = FOREACH groupedWords GENERATE group, COUNT(wordsInEachLine);  
grunt>
```

15. DUMP countedWords;

Now the Final Output we are getting as word count for every word.

A screenshot of a terminal window titled "clouders@quickstart~". The window displays a log from the Apache Pig system. The log includes various INFO messages about the execution environment, such as the number of cores (4), memory (6.0GB), and the use of MapReduce as the execution engine. It also shows the configuration of various optimization rules like Rule 45 and Rule 10. The log ends with a message from the ControlCompiler.

A screenshot of a Cloudera QuickStart VM desktop environment. The desktop has a blue theme with various icons for applications like Computer, Trash, Eclipse, and Cloudera Home. A terminal window titled "cloudera@quickstart:" is open, displaying log output from an Apache Hue run. The log shows successful execution of a ReduceLauncher job and various INFO messages related to Hadoop configuration, schema tuple generation, and mapreduce execution engine util. The terminal window is positioned over a large, semi-transparent watermark of the letter 'a'.

As we can see from above image the Word “We” start with capital W occurred four times, word “Big” occurred once, and so on.

Now Exit from the grunt shell using quit command.

16. quit

```
Migrate to Fancier  
  (Technology,1)  
  (implemented,1)  
  grunt> quit  
 [cloudera@quickstart ~]$
```

Aim: Demonstrate the use of Sqoop tool to transfer data between Hadoop & relational database servers

Sqoop:

The traditional application management system, that is, the interaction of applications with relational database using RDBMS, is one of the sources that generate Big Data. Such Big Data, generated by RDBMS, is stored in Relational Database Servers in the relational database structure.

When Big Data storages and analyzers such as MapReduce, Hive, HBase, Cassandra, Pig, etc. of the Hadoop ecosystem came into picture, they required a tool to interact with the relational database servers for importing and exporting the Big Data residing in them. Here, Sqoop occupies a place in the Hadoop ecosystem to provide feasible interaction between relational database server and Hadoop's HDFS.

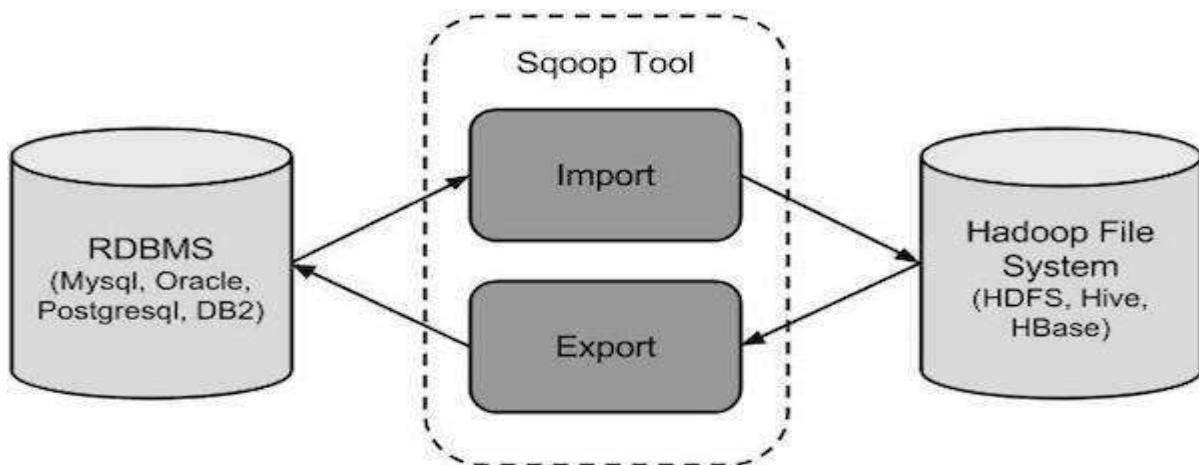
Sqoop – “SQL to Hadoop and Hadoop to SQL”

Sqoop is a tool designed to transfer data between Hadoop and external datastores such as relational databases and enterprise data warehouses.

It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

It imports data from external datastores into HDFS, Hive, and HBase.

Working of Sqoop:



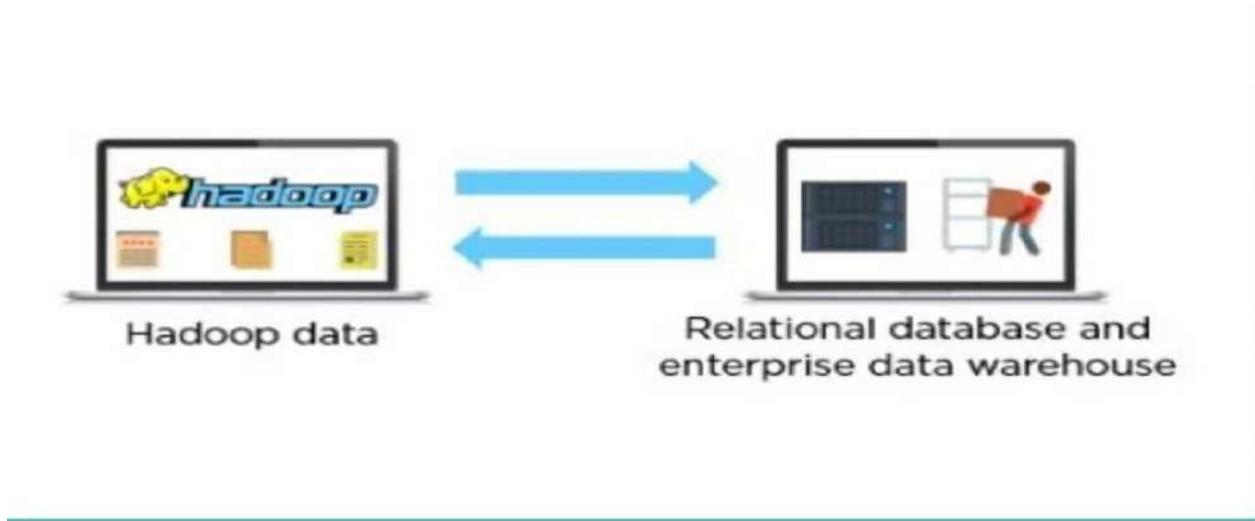
There are mainly two functions associated with Apache Sqoop tool which are Sqoop Import and Sqoop export.

1. Sqoop Import

Sqoop Import This is an important function which executes the task of data importing from external sources (RDBMS) to HDFS. In HDFS, each row of a table is considered as a record. The entire records are stored in a text format in the text files or as binary data in Sequence files.

2. Sqoop Export

Sqoop Export This function performs bulk data exportation tasks from the HDFS to RDBMS. Once the modifications are done to the imported records you will get a result set and the next process is to send back the data to the relational database (RDBMS). Sqoop export function reads a group of delimited text files from HDFS in parallel, divides the files into records, and stores them as new rows in a targeted database table.



Before performing the practical first perform 3 steps of the given following

1. sudo /home/cloudera/cloudera-manager --force --express

```
cloudera@quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Fri Mar 25, 1:15 AM cloudera
cloudera@quickstart:~
```

File Edit View Search Terminal Help

[QuickStart] Waiting for Cloudera Manager API...

[QuickStart] Starting Cloudera Manager agent...

[QuickStart] Configuring deployment...

Submitted jobs: 41

[QuickStart] Deploying client configuration...

Submitted jobs: 42

[QuickStart] Starting Cloudera Management Service...

Submitted jobs: 50

[QuickStart] Enabling Cloudera Manager daemons on boot...

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:

http://quickstart.cloudera:7180

Username: cloudera
Password: cloudera

[cloudera@quickstart ~]\$ sudo -u hdfs hadoop dfsadmin -safemode leave

DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

22/03/25 01:04:22 WARN ipc.Client: Failed to connect to server: quickstart.cloudera/10.0.2.15:8020: try once and fail.

java.net.ConnectException: Connection refused

at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)

at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:739)

at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:266)

at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)

at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)

at org.apache.hadoop.ipc.Client\$Connection.setupConnection(Client.java:648)

at org.apache.hadoop.ipc.Client\$Connection.access\$3000(Client.java:744)

at org.apache.hadoop.ipc.Client\$Connection.access\$3000(Client.java:396)

2. Start all services

3. sudo -u hdfs hadoop dfsadmin -safemode leave

```
cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ Fri Mar 25, 1:16 AM cloudera
File Edit View Search Terminal Help
at org.apache.hadoop.ipc.ClientsConnection.setupConnection(Client.java:648)
at org.apache.hadoop.ipc.ClientsConnection.setupOutputStreams(Client.java:744)
at org.apache.hadoop.ipc.ClientsConnection.access$3009(Client.java:396)
at org.apache.hadoop.ipc.ClientsConnection$InvocationHandler.invoke(Ava:1557)
at org.apache.hadoop.ipc.Client$Call.call(Client.java:1480)
at org.apache.hadoop.ipc.Client.call(Client.java:1442)
at org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:236)
at com.sun.proxy.$Proxy17.setSafeMode(Unknown Source)
at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolTranslatorPB.setSafeMode(ClientNamenodeProtocolTranslatorPB.java:681)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at org.apache.hadoop.ipc.RetryInvocationHandler.invoke(RetryInvocationHandler.java:268)
at org.apache.hadoop.ipc.RetryInvocationHandler.invoke(RetryInvocationHandler.java:104)
at com.sun.proxy.$Proxy17.setSafeMode(Unknown Source)
at org.apache.hadoop.hdfs.DFSClient.setSafeMode(DFSClient.java:2048)
at org.apache.hadoop.hdfs.DistributedFileSystem.setSafeMode(DistributedFileSystem.java:1182)
at org.apache.hadoop.hdfs.DistributedFileSystem.setSafeMode(DistributedFileSystem.java:1166)
at org.apache.hadoop.hdfs.tools.DFSAdmin.setSafeMode(DFSAdmin.java:570)
at org.apache.hadoop.hdfs.tools.DFSAdmin.run(DFSAdmin.java:1856)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
at org.apache.hadoop.hdfs.tools.DFSAdmin.main(DFSAdmin.java:2832)
safemode: Call From quickstart.cloudera:10.0.2.15 to quickstart.cloudera:8020 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/DFSAdmin#safemode_leaves
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$
```

Steps: Demonstrate the use of Sqoop tool to transfer data between Hadoop & relational database servers

- Starting the mysql by giving username as root and password as cloudera.

mysql -u root -pcloudera

```
cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ Fri Mar 25, 8:10 PM cloudera
File Edit View Search Terminal Help
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> Ctrl-C -- exit!
Aborted
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$ mysql -u root -pcloudera
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 432
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

- Now using below command it will displaying or give the list of databases which are already present or exist in mysql.

show databases;

```
cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera@quickstart:~$ Fri Mar 25, 8:11 PM cloudera
File Edit View Search Terminal Help
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cm |
| firehose |
| hue |
| metastore |
| mysql |
| nav |
| navms |
| oozie |
| retail_db |
| rman |
| sentry |
+-----+
12 rows in set (0.03 sec)

mysql>
```

3. Now we are using the existing database i.e. retail_db which are already present in mysql.

`use retail_db;`

```
cloudera@quickstart:~$ use retail_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
```

So right now we are under **retail_db** database.

4. Now to see the tables under a specific database so we will be using the same command which is used to display the databases.

`show tables;`

```
cloudera@quickstart:~$ show tables;
+-----+
| Tables in retail_db |
+-----+
| categories          |
| customers           |
| departments          |
| order_items          |
| orders               |
| products             |
+-----+
6 rows in set (0.00 sec)
```

5. Here we are displaying all the records present in customers tables using below command.

`select * from customers;`

ID	First Name	Last Name	Address	City	State	Zip Code
12430	Hannah	Brown	Caguas	XXXXXX	PR	
8316	Pleasant Bend					
12431	Mary	Rios	XXXXXX	XXXXXX	HI	
1221	Cinder Pines		Kaneohe			
96744						
12432	Angela	Smith	XXXXXX	XXXXXX	PR	
1525	Jagged Barn Highlands		Caguas			
86725						
12433	Benjamin	Garcia	Levittown	XXXXXX	NY	
5459	Noble Brook Landing					
11756						
12434	Mary	Mills	XXXXXX	XXXXXX	PR	
9720	Colonial Parade		Caguas			
68725						
12435	Laura	Horton	Summerville	XXXXXX	SC	
5796	Honey Downs					
29483						
12435						

As it is a huge table. It contains total 12435 rows or records.

select * from customers limit 10;

```
cloudera@quickstart:~$ select * from customers limit 10;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'from customers' at line 1
cloudera@quickstart:~$ select top(10) * from customers;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'from customers' at line 1
cloudera@quickstart:~$ select * from customers limit 10;
+-----+-----+-----+-----+
| customer_id | customer_fname | customer_lname | customer_email | customer_password |
| customer_street | customer_city | customer_state | customer_zipcode |
+-----+-----+-----+-----+
| 1 | Richard | Hernandez | XXXXXXXXX | XXXXXXXXXX |
| 6303 Heather Plaza | Brownsville | TX | 78521 |
| 2 | Mary | Barrett | XXXXXXXXX | XXXXXXXXXX |
| 9526 Noble Embers Ridge | Littleton | CO | 80126 |
+-----+-----+-----+-----+
```

6. Let see any other table. Here we are displaying department table it has 6 rows in it.

select * from departments;

```
cloudera@quickstart:~$ select * from departments;
+-----+-----+
| department_id | department_name |
+-----+-----+
| 2 | Fitness |
| 3 | Footwear |
| 4 | Apparel |
| 5 | Gift |
| 6 | Outdoors |
| 7 | Fan Shop |
+-----+-----+
6 rows in set (0.00 sec)
```

These are the different departments i.e. department_name with their respective department_id.

7. Open the new terminal for running command for sqoop.

8. We will require hostname for this sqoop .

hostname -f

```
cloudera@quickstart:~$ hostname -f
quickstart.cloudera
cloudera@quickstart:~$
```

```
cloudera@quickstart:~$ select * from departments;
+-----+-----+
| department_id | department_name |
+-----+-----+
| 2 | Fitness |
| 3 | Footwear |
| 4 | Apparel |
| 5 | Gift |
| 6 | Outdoors |
| 7 | Fan Shop |
+-----+-----+
```

So it is giving as **quickstart.cloudera** as the name of the host that we are already connected to.

9. Then if we want to list down all the databases we will use below sqoop command.

sqoop list-databases --connect jdbc:mysql://quickstart:3306 --password cloudera --username root;

So we have studied in this sqoop that we can make use of jdbc or the odbc type driver. So the applications that support the jdbc will be connecting them with the jdbc driver. So here we are using mysql which we are going to connect this with the jdbc . mysql running on **quickstart:3306** then we have to mention password which is **cloudera** and username i.e. **root**.

```
[cloudera@quickstart ~]$ sqoop list-databases --connect jdbc:mysql://quickstart:3306 --password cloudera --username root;
Warning: /usr/lib/sqoop/./.accumulo does not exist. Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
22/03/25 20:45:11 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
22/03/25 20:45:11 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/03/25 20:45:12 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
information_schema
firehose
hue
metastore
mysql
Performance_schema
navms
oozie
retail_db
rman
security
[cloudera@quickstart ~]$
```

These are the lists of same databases which are present in mysql and the information also present here. So we are connecting the sqoop with mysql with the help of jdbc.

10. Now we will list out all the tables using below command.

sqoop list-tables --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root;

```
[cloudera@quickstart ~]$ sqoop list-tables --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root;
Warning: /usr/lib/sqoop/./.accumulo does not exist. Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
22/03/25 20:59:39 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
22/03/25 20:59:39 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/03/25 20:59:39 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
categories
customers
departments
order_items
orders
products
[cloudera@quickstart ~]$
```

11. Now we will start with the import and export tools of the Hadoop. We want to Import table "departments" from retail_db database which are present inside in mysql.

sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments;

```
[cloudera@quickstart-vms-5.13.0-0-virtualbox ~]$ sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments
HDFS: Number of large read operations=8
HDFS: Number of write operations=8
Job Counters
Launched map tasks=4
   CPU time spent (ms)=480
   Total time spent by all maps in occupied slots (ms)=56487424
   Total time spent by all reduces in occupied slots (ms)=0
   Total time spent by all map tasks (ms)=110327
   Total vcore-milliseconds taken by all map tasks=110327
   Total megabyte-milliseconds taken by all map tasks=56487424
Map-Reduce Framework
   Map input records=6
   Map output records=6
   Input split bytes=481
   Spilled Records=0
   Failed Shuffles=0
   Merged Map outputs=1
   CPU time spent (ms)=9100
   Physical memory (bytes) snapshot=545624964
   Virtual memory (bytes) snapshot=290861696
   Total committed heap usage (bytes)=197132288
File Input Format Counters
   Bytes Read=0
   File Output Format Counters
   Bytes Written=60
22/03/25 20:54:38 INFO mapreduce.ImportJobBase: Transferred 60 bytes in 100.9359 seconds (0.5944 bytes/sec)
22/03/25 20:54:38 INFO mapreduce.ImportJobBase: Retrieved 6 records.
[cloudera@quickstart ~]$
```

12. Now we will see whether all departments table successfully imported from mysql in Hadoop (hdfs) or not using below command.

hadoop fs -ls

```
[cloudera@quickstart ~]$ hadoop fs -ls
Found 3 items
drwx-----  cloudera cloudera  0 2022-03-25 20:54 .staging
drwxr-xr-x  - cloudera cloudera  0 2022-03-25 02:22 Training
drwxr-xr-x  - cloudera cloudera  0 2022-03-25 20:54 [departments]
```

Departments table is now successfully imported in hdfs.

13. Now we will see what inside this department using below command.

hadoop fs -ls departments;

```
[cloudera@quickstart ~]$ hadoop fs -ls departments;
Found 5 items
-rw-r--r--  1 cloudera cloudera  0 2022-03-25 20:54 departments/_SUCCESS
-rw-r--r--  1 cloudera cloudera 21 2022-03-25 20:54 departments/part-m-00000
-rw-r--r--  1 cloudera cloudera 10 2022-03-25 20:54 departments/part-m-00001
-rw-r--r--  1 cloudera cloudera  7 2022-03-25 20:54 departments/part-m-00002
-rw-r--r--  1 cloudera cloudera 22 2022-03-25 20:54 departments/part-m-00003
```

As we can see there are once SUCCESS file and four part-m files which has got the output present inside the departments.

14. Now we will see what there inside this some of the part m files so that can be done with the help of below commands.

hadoop fs -cat /user/cloudera/departments/part-m-00000

hadoop fs -cat /user/cloudera/departments/part-m-00001

hadoop fs -cat /user/cloudera/departments/part-m-00002

hadoop fs -cat /user/cloudera/departments/part-m-00003

```
[cloudera@quickstart ~]$ hadoop fs -ls departments;
Found 5 items
-rw-r--r--  1 cloudera cloudera  0 2022-03-25 20:54 departments/_SUCCESS
-rw-r--r--  1 cloudera cloudera 21 2022-03-25 20:54 departments/part-m-00000
-rw-r--r--  1 cloudera cloudera 10 2022-03-25 20:54 departments/part-m-00001
-rw-r--r--  1 cloudera cloudera  7 2022-03-25 20:54 departments/part-m-00002
-rw-r--r--  1 cloudera cloudera 22 2022-03-25 20:54 departments/part-m-00003
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part-m-00000
cat: '/user/cloudera/departments/part-m-00000': No such file or directory
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part-m-00000
2.Fitness
3.Footwear
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part-m-00001
4.Apparel
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part-m-00002
5.Golf
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part-m-00003
6.Outdoors
7.Fan Shop
[cloudera@quickstart ~]$
```

15. If want to display output of all part-m files together we will use below command.

hadoop fs -cat /user/cloudera/departments/part*

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/departments/part*
2.Fitness
3.Footwear
4.Apparel
5.Golf
6.Outdoors
7.Fan Shop
[cloudera@quickstart ~]$
```

16. If we want to mention that where should we have our this output in the hdfs so for that we have to mention the target directory.

We will want to import my department table in –target directory as department1.

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments --target-dir /user/cloudera/department1
```

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments --target-dir /user/cloudera/department1
HDFS: Number of bytes read=481
HDFS: Number of bytes written=60
HDFS: Number of read operations=16
HDFS: Number of large read operations=0
HDFS: Number of write operations=8
Job Counters:
    Launched map tasks=4
    Other local map tasks=4
    Total time spent by all maps in occupied slots (ms)=52659712
    Total time spent by all reducers in occupied slots (ms)=8
    Total time spent by all map tasks (ms)=182851
    Total vcore-milliseconds taken by all map tasks=162851
    Total megabyte-milliseconds taken by all map tasks=52659712
Map-Reduce Framework
    Map input records=6
    Map output records=6
    Input split bytes=401
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=839
    CPU time spent (ms)=9500
    Physical memory (bytes) snapshot=538230784
    Virtual memory (bytes) snapshot=2969560822
    Total committed heap usage (bytes)=197132288
File Input Format Counters
    Bytes Read=9
    File Output Format Counters
    Bytes Written=60
22/03/25 21:28:15 INFO mapreduce.ImportJobBase: Transferred 60 bytes in 85.597 seconds (0.7011 bytes/sec)
22/03/25 21:28:15 INFO mapreduce.ImportJobBase: Retrieved 6 records.
[cloudera@quickstart ~]$
```

As we can see 6 records are retrieved successfully.

17. Now let's check it using below command.

```
hadoop fs -ls
```

```
[cloudera@quickstart ~]$ hadoop fs -ls
Found 4 items
drwx-----  - cloudera cloudera          0 2022-03-25 21:28 .staging
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 02:22 Training
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 21:28 department1
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 20:54 departments
[cloudera@quickstart ~]$
```

So here we have department1 directory.

18. Now we will check what is present inside this department1 directory using below command.

```
hadoop fs -ls /user/cloudera/department1
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/department1
Found 4 items
drwx-----  - cloudera cloudera          0 2022-03-25 21:28 .staging
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 02:22 Training
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 21:28 department1
drwxr-xr-x  - cloudera cloudera          0 2022-03-25 20:54 departments
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/department1
Found 5 items
-rw-r--r--  1 cloudera cloudera          0 2022-03-25 21:28 /user/cloudera/department1/_SUCCESS
-rw-r--r--  1 cloudera cloudera        21 2022-03-25 21:27 /user/cloudera/department1/part-m-00001
-rw-r--r--  1 cloudera cloudera       10 2022-03-25 21:27 /user/cloudera/department1/part-m-00002
-rw-r--r--  1 cloudera cloudera        7 2022-03-25 21:28 /user/cloudera/department1/part-m-00003
-rw-r--r--  1 cloudera cloudera       22 2022-03-25 21:28 /user/cloudera/department1/part-m-00004
[cloudera@quickstart ~]$
```

So it has different part-m files.

19. Now we will read the content of these part-m files using below command.

```
hadoop fs -cat /user/cloudera/department1/part*
```

```
[cloudera@quickstart ~]$ hadoop fs -cat /user/cloudera/department1/part*
2,Fitness
3,Footwear
4,Apparel
5,Golf
6,Outdoors
7,Fan Shop
[cloudera@quickstart ~]$
```

So we have got the output.

20. Now we will filter out some or specific rows only from the departments table and have it in hdfs but

before we will apply some conditions on the rows of the departments table and whichever rows will satisfy the condition only those rows are would be stored in the hdfs.

We want to fetch only those departments where department_id is greater than 4.

```
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments --where "department_id>4" --target-dir /user/cloudera/department2
```

```
cloudera@quickstart:~$ sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table departments --where "department_id>4" --target-dir /user/cloudera/department2
22/03/25 21:38:29 INFO mapreduce.ImportJobBase: Transferred 29 bytes in 73.7504 seconds (0.4042 bytes/sec)
22/03/25 21:38:29 INFO mapreduce.ImportJobBase: Retrieved 3 records.
[cloudera@quickstart ~]$
```

As we can see 3 records are retrieved successfully.

21. Now we will check it using below command.

```
hadoop fs -ls /user/cloudera/department2
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/department2
Found 4 items
-rw-r--r-- 1 cloudera cloudera 0 2022-03-25 21:38 /user/cloudera/department2/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 7 2022-03-25 21:38 /user/cloudera/department2/part-m-00000
-rw-r--r-- 1 cloudera cloudera 11 2022-03-25 21:38 /user/cloudera/department2/part-m-00001
-rw-r--r-- 1 cloudera cloudera 11 2022-03-25 21:38 /user/cloudera/department2/part-m-00002
[cloudera@quickstart ~]$
```

22. Now will read the content of these part-m files using cat command.

```
hadoop fs -ls /user/cloudera/department2/part*
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/department2/part*
-rw-r--r-- 1 cloudera cloudera 7 2022-03-25 21:38 /user/cloudera/department2/part-m-00000
-rw-r--r-- 1 cloudera cloudera 11 2022-03-25 21:38 /user/cloudera/department2/part-m-00001
-rw-r--r-- 1 cloudera cloudera 11 2022-03-25 21:38 /user/cloudera/department2/part-m-00002
[cloudera@quickstart ~]$
```

23. Now we will see the Export command. So what the export tool does is it willexport the data from our hdfs to the RDBMS. So for that we need to have some table in mysql with some records so for that we will now move to mysql.

24. So here we will create the table “dpt” and it will be having two attributes as

“department_id” and “ department_name”.

```
create table dpt(department_id int not null auto_increment, department_name varchar(50) not null, primary key(department_id));
```

```
cloudera@quickstart:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.7.24-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from departments;
+-----+-----+
| department_id | department_name |
+-----+-----+
| 1 | Marketing |
| 2 | Fitness |
| 3 | Consumer Packaged Goods |
| 4 | Apparel |
| 5 | Golf |
| 6 | Outdoors |
| 7 | Fan Shop |
+-----+-----+
7 rows in set (0.00 sec)

mysql> create table dpt(department_id int not null auto increment, department_name varchar(50) not null, primary key(department_id));
Query OK, 0 rows affected (0.05 sec)

mysql> \q
cloudera@quickstart:~$ hadoop fs -cat /user/cloudera/department2/part* | mysql -u root -p
Enter password:
Query OK, 0 rows affected (0.05 sec)

mysql> \q
cloudera@quickstart:~$
```

25. Now we want to check what we have inside this dpt table.

select * from dpt;

As we have not inserted any records inside the dpt table so that's why it is showing as Empty set.

26. Now we will exporting the data from the hdf5 to dpt table of mysql. Now we will move to the sqoop terminal.

27. So now we are performing export operation using below command.

we are trying to export department2 which are present in cloudera to inside our dpt tablewhich are present inside mysql.

```
sqoop export --connect jdbc:mysql://quickstart:3306/retail_db --password cloudera --username root --table dpt --export-dir /user/cloudera/department2
```

```
cloudera@quickstart:~$ hdfs dfs -report /tmp
File Edit View Search Terminal Help
File Applications Places System cloudera@quickstart:~
HDFS: Number of bytes read=0
HDFS: Number of bytes written=0
HDFS: Number of read operations=18
HDFS: Number of large read operations=0
HDFS: Number of write operations=0
Job Counters
Launched map tasks=3
Data-local map tasks=3
Total time spent by all maps in occupied slots (ms)=32715776
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=32715776
Total vcore-milliseconds taken by all map tasks=62898
Total megabyte-milliseconds taken by all map tasks=32715776
Map-Reduce Framework
  Map input records=3
  Map output records=3
  Input split bytes=1027
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=558
  CPU time spent (ms)=5940
  Physical memory (bytes) snapshot=432672800
  Virtual memory (bytes) snapshot=2189742980
  Virtual memory (bytes) snapshot=144703488
  Input Format counters
    Bytes Read=0
  File Output Counters
    Bytes Written=0
22/03/25 21:57:21 INFO mapreduce.ExportJobBase: Transferred 0.03 bytes in 70.2615 seconds (9.7268 bytes/sec)
22/03/25 21:57:21 INFO mapreduce.ExportJobBase: Exported 3 records.
[cloudera@quickstart ~]$
```

Now we have successfully exported 3 records.

28. Now we will see whether the records are successfully exported in dpt table which are present inside mysql using below command.

select * from dpt;

```
cloudera@quickstart:~$ hdfs dfs -ls /user/cloudera
Job Counters
  Job Submissions=1
  Data Locality=100.00%
  Total M.R. Input=0 B
  Total M.R. Output=0 B
  Total Map=0
  Total Map=0
Map-Reduce Framework
  Map input records=0
  Map output records=0
  Spilled Records=0
  Failed Shuffles=0
  Bytes Read=0
  Bytes Written=0
  File Input Format=HDFS
  File Output Format=Text
22/03/25 21:57:21 INFO mapred.ExportJobBase: Transferred 0.00 bytes in 70.2615 seconds (0.7208 bytes/sec)
22/03/25 21:57:21 INFO mapred.ExportJobBase: Exported 0 records.
[cloudera@quickstart ~]$
```

As we can see these 3 records which are present in department2 table are successfully exported inside the dpt table of mysql.

quit

```
VIRTUAL Total cmd: mysql> quit  
File Input Form: Bye  
Bytes Read [cloudera@quickstart ~]$  
File Output Form: counters  
Bytes Written=0  
22/03/25 21:57:21 INFO mapredUCE.ExportJobBase: Transferred 683 bytes in 70.2615 seconds (9.7208 bytes/sec)  
22/03/25 21:57:21 INFO mapredUCE.ExportJobBase: Exported 3 records.  
[cloudera@quickstart ~]$ cloudera@quickstart:~
```

Aim - Schema design using HBaseWhat is HBase?

HBase is a column-oriented non-relational database management system that runs on top of Hadoop Distributed File System (HDFS). HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases. It is well suited for real-time data processing or random read/write access to large volumes of data.

Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java™ much like a typical Apache MapReduce application. HBase does support writing applications in Apache Avro, REST and Thrift.

An HBase system is designed to scale linearly. It comprises a set of standard tables with rows and columns, much like a traditional database. Each table must have an element defined as a primary key, and all access attempts to HBase tables must use this primary key.

Avro, as a component, supports a rich set of primitive data types including: numeric, binary data and strings; and a number of complex types including arrays, maps, enumerations and records. A sort order can also be defined for the data.

HBase relies on ZooKeeper for high-performance coordination. ZooKeeper is built into HBase, but if you're running a production cluster, it's suggested that you have a dedicated ZooKeeper cluster that's integrated with your HBase cluster.

HBase works well with Hive, a query engine for batch processing of big data, to enable fault-tolerant big data applications.

An example of HBase

An HBase column represents an attribute of an object; if the table is storing diagnostic logs from servers in your environment, each row might be a log record, and a typical column could be the timestamp of when the log record was written, or the server name where the record originated.

HBase allows for many attributes to be grouped together into column families, such that the elements of a column family are all stored together. This is different from a row-oriented relational database, where all the columns of a given row are stored together. With HBase you must predefine the table schema and specify the column families. However, new columns can be added to families at any time, making the schema flexible and able to adapt to changing application requirements.

Just as HDFS has a NameNode and slave nodes, and MapReduce has JobTracker and TaskTracker slaves, HBase is built on similar concepts. In HBase a master node manages the cluster and region servers store portions of the tables and perform the work on the data. In the same way HDFS has some enterprise concerns due to the availability of the NameNode HBase is also sensitive to the loss of its master node.

HBase Shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

The master server manages these region servers and all these tasks take place on HDFS. Given below are some of the commands supported by HBase Shell.

Before performing the practical first perform 3 steps of the given following

1. sudo /home/cloudera/cloudera-manager --force --express

```
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --force --express
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Starting Cloudera Manager agent...
[QuickStart] Configuring deployment...
Submitted jobs: 41
[QuickStart] Deploying client configuration...
Submitted jobs: 42
[QuickStart] Starting Cloudera Management Service...
Submitted jobs: 50
[QuickStart] Enabling Cloudera Manager daemons on boot...

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:
http://quickstart.cloudera:7180

Username: cloudera
Password: cloudera

[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

22/03/25 01:04:22 WARN ipc.Client: Failed to connect to server: quickstart.cloudera/10.0.2.15:8020: try once and fail.
java.net.ConnectException: Connection refused
    at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
    at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:739)
    at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:206)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:530)
    at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:494)
    at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:648)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:744)
    at org.apache.hadoop.ipc.Client$Connection.access$3000(Client.java:396)
```

2. Start all services

The left screenshot shows the Cloudera Manager interface with three clouds (HDFS, MapReduce, YARN) each having a green 'Running' status indicator. The right screenshot shows a detailed log for the HDFS service, specifically for the 'Restart Command' section. It lists several log entries, including 'All services successfully started.' and 'Successfully completed 1 stage.'

```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$
```

3. sudo -u hdfs hadoop dfsadmin -safemode leave

We can start the HBase interactive shell using “hbase shell” command as shown below.

hbase shell

```

cloudera@quickstart:~$ hbase shell
22/02/25 01:19:20 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help' or 'RETURN' for list of supported commands.
Type "exit" or "RETURN" to leave the HBase Shell.
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0>
  
```

Check the shell functioning before proceeding further. Use the list command for this purpose. List is a command used to get the list of all the tables in HBase. It lists all the tables in HBase.

List

```

cloudera@quickstart:~$ hbase shell
22/02/25 01:19:20 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help' or 'RETURN' for list of supported commands.
Type "exit" or "RETURN" to leave the HBase Shell.
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017

hbase(main):001:0> list
TABLE
2 row(s) in 0.8268 seconds
=> []
hbase(main):002:0>
  
```

Restart HBase services if this is not running on terminal

\$ sudo su – This command is to become super user

\$ service hbase-master restart – This command is to restart hbase-master services

\$ service hbase-regionserver restart – This command is to restart hbase-regionserver services

Once these commands run successfully then Open the browser and refresh the page and see all the HBase servers will be restarted.

Schema Design using HBase

The HBase schema design is very different compared to the relation database schema design. Below are some of general concept that should be followed while designing schema in Hbase:

Row key: Each table in HBase table is indexed on row key. Data is sorted lexicographically by this row key. There are no secondary indices available on HBase table.

Automaticity: Avoid designing table that requires atomacity across all rows. All operations on HBase rows are atomic at row level.

Even distribution: Read and write should uniformly distributed across all nodes available in cluster. Design row key in such a way that, related entities should be stored in adjacent rows to increase read efficacy.

HBase Schema Row key, Column family, Column qualifier, individual and Row value SizeLimit

1. Creating a Table using HBase Shell

We can create a table using the create command, here you must specify the table name and the Column Family name. The syntax to create a table in HBase shell is shown below.

create '<table name>','<column family>'

create 'customer','address','order'

customer -> table ; address & order -> column_name

```

cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System cloudera@quickstart:~ Fri Mar 25, 1:24 AM
File Edit View Search Terminal Help
at com.sun.proxy.$Proxy17.setSafeMode(Unknown Source)
at org.apache.hadoop.hdfs.DFSClient.setSafeMode(DFSClient.java:2648)
at org.apache.hadoop.hdfs.DistributedFileSystem.setSafeMode(DistributedFileSystem.java:1182)
at org.apache.hadoop.hdfs.DistributedFileSystem.setSafeMode(DistributedFileSystem.java:1166)
at org.apache.hadoop.hdfs.DistributedFileSystem.setSafeMode(DistributedFileSystem.java:576)
at org.apache.hadoop.hdfs.tools.DFSAdmin.run(DFSAdmin.java:1856)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
at org.apache.hadoop.hdfs.tools.DFSAdmin.main(DFSAdmin.java:2032)
safemode: Call From quickstart.cloudera:10.0.2.15 to quickstart.cloudera:8020 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$ hbase shell
22/03/25 01:19:20 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help|RETURN' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017
hbase(main):001:0> list
TABLE
0 row(s) in 0.8200 seconds
=> []
hbase(main):002:0> create 'customer','address','order'
0 row(s) in 2.6180 seconds
=> Hbase::Table - customer
hbase(main):003:0> []
cloudera@quickstart:~$ Home - Cloudera Man...

```

list

```

cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System cloudera@quickstart:~ Fri Mar 25, 1:27 AM
File Edit View Search Terminal Help
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
at org.apache.hadoop.hdfs.tools.DFSAdmin.main(DFSAdmin.java:2032)
safemode: Call From quickstart.cloudera:10.0.2.15 to quickstart.cloudera:8020 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
[cloudera@quickstart ~]$ sudo -u hdfs hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Safe mode is OFF
[cloudera@quickstart ~]$ hbase shell
22/03/25 01:19:20 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help|RETURN' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct 4 11:16:18 PDT 2017
hbase(main):001:0> list
TABLE
0 row(s) in 0.8200 seconds
=> []
hbase(main):002:0> create 'customer','address','order'
0 row(s) in 2.6180 seconds
=> Hbase::Table - customer
hbase(main):003:0> list
TABLE
customer
1 row(s) in 0.0450 seconds
=> ['customer']
hbase(main):004:0> []
cloudera@quickstart:~$ Home - Cloudera Man...

```

2. Put: Inserts a new record into the table with row identified by 'row.'

This command is used for following things

- It will put a cell 'value' at defined or specified table or row or column.
- It will optionally coordinate time stamp.

Syntax: put <tablename>,<rowname>,<columnvalue>,<value>

Example – with the help of put commands we have inserted new records in “customer” table for address and order family. Here name “Nick” is Row key

```
put 'customer','Nick','address:city','Mumbai'
put 'customer','Nick','address:state','Maharashtra'
put 'customer','Nick','address:street','Street1'
put 'customer','Nick','order:number','ORD-15'
put 'customer','Nick','order:amount','50'
put 'customer','Nick','address:state','Maharashtra'
```

```
cloudera@quickstart:~$ hbase(main):001:0> put 'customer','Nick','address:city','Mumbai'
0 row(s) in 0.0200 seconds
cloudera@quickstart:~$ put 'customer','Nick','address:state','Maharashtra'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Nick','address:street','Street1'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Nick','order:number','ORD-15'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Nick','order:amount','50'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Nick','address:state','Maharashtra'
0 row(s) in 0.0100 seconds
```

Adding one more record of customer name “Justin”. Here name “Justin” is Row key

```
put 'customer','Justin','address:city','Pune'
put 'customer','Justin','address:state','Maharashtra'
put 'customer','Justin','order:number','ORD-16'
put 'customer','Justin','order:amount','60'
```

```
cloudera@quickstart:~$ put 'customer','Nick','order:amount','50'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Nick','order:number','ORD-15'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Justin','address:city','Pune'
0 row(s) in 0.0300 seconds
cloudera@quickstart:~$ put 'customer','Justin','address:state','Maharashtra'
0 row(s) in 0.0000 seconds
cloudera@quickstart:~$ put 'customer','Justin','address:street','Street2'
0 row(s) in 0.0000 seconds
cloudera@quickstart:~$ put 'customer','Justin','order:number','ORD-16'
0 row(s) in 0.0100 seconds
cloudera@quickstart:~$ put 'customer','Justin','order:amount','60'
0 row(s) in 0.0100 seconds
```

3. Get: Returns the records matching the row identifier provided in the table

By using this command, you will get a row or cell contents present in the table. In addition to that you can also add additional parameters to it like TIMESTAMP, TIMERANGE, VERSIONS, FILTERS, etc. to get a particular row or cell content.

Syntax: get <tablename>,<rowname>,{<Additional parameters>}

a) get 'customer', 'Nick'

cloudera@quickstart:~\$ hbase(main):011:0> put 'customer','Justin','address:state','Maharashtra'
8 row(s) in 0.0880 seconds

cloudera@quickstart:~\$ hbase(main):012:0> put 'customer','Justin','address:street','street2'
8 row(s) in 0.0890 seconds

cloudera@quickstart:~\$ hbase(main):013:0> put 'customer','Justin','order:number','ORD-16'
8 row(s) in 0.0100 seconds

cloudera@quickstart:~\$ hbase(main):014:0> put 'customer','Justin','order:amount','60'
8 row(s) in 0.0130 seconds

cloudera@quickstart:~\$ hbase(main):015:0> get 'customer', 'Nick'
COLUMN CELL
address:city timestamp=1648197017898, value=Mumbai
address:number timestamp=1648197149436, value=ORD-15
address:state timestamp=1648197068315, value=Maharashtra
address:street timestamp=1648197098129, value=street1
order:amount timestamp=1648197193433, value=50
order:number timestamp=1648197213674, value=ORD-15
6 row(s) in 0.0490 seconds

cloudera@quickstart:~\$ hbase(main):016:0>

b) Additional parameters to get only address details**get 'customer', 'Nick','address'**

cloudera@quickstart:~\$ hbase(main):016:0> get 'customer', 'Nick', 'address'
COLUMN CELL
address:city timestamp=1648197260721, value=Pune
address:street timestamp=1648197275891, value=street1
address:state timestamp=1648197293721, value=street2
order:amount timestamp=1648197354548, value=60
order:number timestamp=1648197336614, value=ORD-16
5 row(s) in 0.0490 seconds

cloudera@quickstart:~\$ hbase(main):017:0> get 'customer', 'Justin', 'address'
COLUMN CELL
address:city timestamp=1648197017898, value=Mumbai
address:number timestamp=1648197149436, value=ORD-15
address:state timestamp=1648197068315, value=Maharashtra
address:street timestamp=1648197098129, value=street1
4 row(s) in 0.0220 seconds

cloudera@quickstart:~\$ hbase(main):018:0>

c) Additional parameters to get only city details**get 'customer', 'Nick','address:city'**

cloudera@quickstart:~\$ hbase(main):018:0> get 'customer', 'Nick', 'address:city'
COLUMN CELL
address:city timestamp=1648197017898, value=Mumbai
1 row(s) in 0.0480 seconds

cloudera@quickstart:~\$ hbase(main):019:0>

4. Scan: The scan command is used to view the data in HTable. Using the scan

command, you can get the table data.

- This command scans entire table and displays the table contents.
- We can pass several optional specifications to this scan command to get more information about the tables present in the system.
- Scanner specifications may include one or more of the following attributes.
 - These are TIMERANGE, FILTER, TIMESTAMP, LIMIT, MAXLENGTH, COLUMNS, CACHE, STARTROW and STOPROW.

Its syntax is as follows:

Syntax: **scan <'tablename'>, {Optional parameters}**

scan 'customer'

When we execute above commands in HBase then we will be getting all the table “customer” contents along with additional parameters like timestamp as show in below screenshot.

```

cloudera-quickstart-vm-5.15.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System Terminal cloudera@quickstart:~ Fri Mar 25, 1:42 AM
File Edit View Search Terminal Help
address:city timestamp=1648197917890, value=Mumbai
1 row(s) in 0.0480 seconds
hbase(main):020:0> scan 'customer'
ROW COLUMN+CELL
Justin    column=address:city, timestamp=1648197250721, value=Pune
Justin    column=address:state, timestamp=1648197275891, value=Maharashtra
Justin    column=address:street, timestamp=1648197293721, value=street 123
Justin    column=order:amount, timestamp=1648197254540, value=60
Justin    column=order:number, timestamp=1648197336014, value=ORD-16
Nick     column=address:city, timestamp=1648197817890, value=Mumbai
Nick     column=address:number, timestamp=1648197149436, value=ORD-15
Nick     column=address:state, timestamp=1648197868315, value=Maharashtra
Nick     column=address:street, timestamp=1648197098129, value=street 456
Nick     column=order:amount, timestamp=1648197193433, value=50
Nick     column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.1280 seconds
hbase(main):021:0>

```

5. Delete -Using the delete command, you can delete a specific cell in a table.

- This command will delete cell value at defined table of row or column.
- Delete must and should match the deleted cells coordinates exactly.
- When scanning, delete cell suppresses older versions of values. The syntax of delete command is as follows:

Syntax:**delete <'tablename'>,<'row name'>,<'column name'>**

delete 'customer','Nick','address:street'

The above command below delete street from address family for row key “Nick” from “customer” table.

Use scan command to see if street is deleted for customer “Nick”. As we can see “street” information is deleted from customer “Nick” from below screenshot.

```

cloudera@quickstart:~$ hbase(main):021:0> delete 'customer','Nick','address:street'
0 row(s) in 0.0770+ seconds

hbase(main):022:0> scan 'customer'
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Justin                                column=address:state, timestamp=1648197275891, value=Maharashtra
Justin                                column=address:street, timestamp=1648197293721, value=street
Justin                                column=order:amount, timestamp=16481972936014, value=ORD-16
Nick                                 column=address:city, timestamp=1648197017890, value=Mumbai
Nick                                 column=address:number, timestamp=1648197149436, value=ORD-15
Nick                                 column=address:state, timestamp=1648197068315, value=Maharashtra
Nick                                 et1
Nick                                 column=order:amount, timestamp=1648197193433, value=50
Nick                                 column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.1280 seconds

hbase(main):023:0> ■

```

6. Alter - This command alters the column family schema. To understand what exactly it does, we have explained it here with an example.

Alter commands are useful for below cases -

- Altering single, multiple column family names
- Deleting column family names from table
- Several other operations using scope attributes with table

Syntax: **alter <tablename>, NAME=><column family name>, VERSIONS=>5**

We can delete specific column family by using alter commands

alter 'customer','delete' => 'address'

After deleting "address" family from "customer" table. Let's again check customer table using "scan" commands as follow

scan 'customer'

As you can see from below screenshot we only now have order:amount and order:number.

```

hbase(main):023:0> alter 'customer','delete' => 'address'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.4640 seconds

hbase(main):024:0> scan 'customer'
ROW                                     COLUMN+CELL
Justin                                column=order:amount, timestamp=1648107354540, value=60
Justin                                column=order:number, timestamp=1648107336014, value=ORD-16
Nick                                 column=order:amount, timestamp=1648197193433, value=50
Nick                                 column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0340 seconds

hbase(main):025:0> ■

```

7. Describe - This command describes the named table.

- It will give more information about column families present in the mentioned table
- In our case, it gives the description about table "customer."
- It will give information about table name with column families, associated filters, versions and some more details.

Syntax:**describe <table name>**

desc 'customer'

```
hbase(main):025:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL => 'FO
REVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0849 seconds
hbase(main):026:0> ■
```

8. Versions –

A {row, column, version} tuple exactly specifies a cell in HBase. In the Apache HBase you can have many cells where row and columns are same but differs only in version values. A version is a timestamp values is written alongside each value. By default, the timestamp values represent the time on the RegionServer when the data was written, but you can change the default HBase setting and specify a different timestamp value when you put data into the cell.

In HBase, rows and column keys are expressed as bytes, the version is specified using a longinteger. The HBase version dimension is stored in decreasing order, so that when reading from a store file, the most recent values are found first.

create 'customer1',{NAME => 'address', VERSIONS => 3}

With the help of above commands we are creating 3 versions

```
cloudera-quickstart:~ 5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System cloudera@quickstart:~ Fri Mar 25, 1:51 AM cloudera
File Edit View Search Terminal Help
Nick column=order:amount, timestamp=1648197193433, value=59
Nick column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0590 seconds
hbase(main):023:0> alter 'customer','delete' => 'address'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.4640 seconds
hbase(main):024:0> scan 'customer'
ROW
Justin column=order:amount, timestamp=1648197354548, value=60
Justin column=order:number, timestamp=1648197336014, value=ORD-16
Nick column=order:amount, timestamp=1648197193433, value=59
Nick column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0340 seconds
hbase(main):025:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL => 'FO
REVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0840 seconds
hbase(main):026:0> create 'customer1',{NAME => 'address', VERSIONS => 3}
0 row(s) in 1.3020 seconds
=> Hbase::Table - customer1
hbase(main):027:0> ■
```

Verifying if “customer1” is created after executing above commands with the help of list commands as shown in screenshot below.

```
cloudera-quickstart:~ 5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System cloudera@quickstart:~ Fri Mar 25, 1:52 AM cloudera
File Edit View Search Terminal Help
1/1 regions updated.
Done.
0 row(s) in 3.4640 seconds
hbase(main):024:0> scan 'customer'
ROW
Justin column=order:amount, timestamp=1648197354548, value=60
Justin column=order:number, timestamp=1648197336014, value=ORD-16
Nick column=order:amount, timestamp=1648197193433, value=59
Nick column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0340 seconds
hbase(main):025:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSIONS => '0', TTL => 'FO
REVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0840 seconds
hbase(main):026:0> create 'customer1',{NAME => 'address', VERSIONS => 3}
0 row(s) in 1.3020 seconds
=> Hbase::Table - customer1
hbase(main):027:0> list
TABLE
customer
customer1
2 row(s) in 0.0000 seconds
=> ["customer", "customer1"]
hbase(main):028:0> ■
```

9. Count - You can count the number of rows of a table using the count command. Its syntax is as follows:

count 'customer'

```
hbase(main):027:0> list
TABLE
customer
customer1
2 row(s) in 0.0060 seconds
=> ["customer", "customer1"]
hbase(main):028:0> count 'customer'
2 row(s) in 0.2910 seconds
=> 2
hbase(main):029:0> ■
```

10. Alter -Update the version number for already existing columns family

alter 'customer', NAME => 'address', VERSIONS => 5

```
hbase(main):029:0> alter 'customer', NAME => 'address', VERSIONS => 5
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.3090 seconds
hbase(main):030:0> ■
```

Again using describe command to see version is updated

desc 'customer'

```
hbase(main):030:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', COMPRESSION => 'NONE', VERSIONS => '5', TTL => 'FOREVER', MIN_VERSION => '0', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0710 seconds
hbase(main):031:0> ■
```

scan 'customer'

```
hbase(main):031:0> scan 'customer'
ROW                                     COLUMN+CELL
Justin                                  column=address:city, timestamp=1648197260721, value=Pune
Justin                                  column=address:state, timestamp=1648197275891, value=Maharashtra
Justin                                  column=address:street, timestamp=1648197293721, value=street2
Justin                                  column=order:amount, timestamp=1648197354540, value=60
Justin                                  column=order:number, timestamp=1648197336014, value=ORD-16
Nick                                    column=address:city, timestamp=1648197017890, value=Mumbai
Nick                                    column=address:number, timestamp=1648197149436, value=ORD-15
Nick                                    column=address:state, timestamp=1648197068315, value=Maharashtra
Nick                                    column=order:amount, timestamp=1648197193433, value=50
Nick                                    column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0910 seconds
hbase(main):032:0> ■
```

put 'customer', 'Nick' , 'address:city', 'Pune'

put 'customer', 'Nick' , 'address:city', 'Bangalore'

put 'customer', 'Nick' , 'address:city', 'Delhi'

```
hbase(main):032:0> put 'customer', 'Nick', 'address:city', 'Pune'
0 row(s) in 0.0130 seconds
hbase(main):033:0> put 'customer', 'Nick', 'address:city', 'Bangalore'
0 row(s) in 0.0140 seconds
hbase(main):034:0> put 'customer', 'Nick', 'address:city', 'Delhi'
0 row(s) in 0.0190 seconds
hbase(main):035:0> scan 'customer'
ROW                                     COLUMN+CELL
Justin                                  column=address:city, timestamp=1648197260721, value=Pune
Justin                                  column=address:state, timestamp=1648197275891, value=Maharashtra
Justin                                  column=address:street, timestamp=1648197293721, value=street2
Justin                                  column=order:amount, timestamp=1648197354540, value=60
Justin                                  column=order:number, timestamp=1648197336014, value=ORD-16
Nick                                    column=address:city, timestamp=1648197017890, value=Delhi
Nick                                    column=address:number, timestamp=1648197149436, value=ORD-15
Nick                                    column=address:state, timestamp=1648197068315, value=Maharashtra
Nick                                    column=order:amount, timestamp=1648197193433, value=50
Nick                                    column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0288 seconds
hbase(main):036:0> ■
```

Below are example of adding version of address:city

Currently below is the data in 'customer'. We will be adding more information in it one by one.

scan 'customer', {COLUMN => 'address:city', VERSIONS => 2}

```
hbase(main):036:0> scan 'customer', {COLUMN => 'address:city', VERSIONS => 2}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
2 row(s) in 0.0260 seconds
```

```
hbase(main):037:0> 
```

scan 'customer', {COLUMN=> 'address:city', VERSIONS => 3}

scan 'customer', {COLUMN=> 'address:city', VERSIONS => 4}

After executing above commands its giving us 3 and 4 version of address:city records as shown in below screenshot.

```
hbase(main):038:0> scan 'customer', {COLUMN => 'address:city', VERSIONS => 3}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
2 row(s) in 0.0220 seconds

hbase(main):039:0> scan 'customer', {COLUMN => 'address:city', VERSIONS => 4}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
Nick                                  column=address:city, timestamp=1648198734145, value=Pune
Nick                                  column=address:city, timestamp=1648197017890, value=Mumbai
2 row(s) in 0.0540 seconds

hbase(main):040:0> scan 'customer', {COLUMN => 'address:city', VERSIONS => 4}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
Nick                                  column=address:city, timestamp=1648198734145, value=Pune
Nick                                  column=address:city, timestamp=1648197017890, value=Mumbai
2 row(s) in 0.0540 seconds

hbase(main):041:0> scan 'customer', {COLUMN => 'address:city', VERSIONS => 5}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
Nick                                  column=address:city, timestamp=1648198734145, value=Pune
Nick                                  column=address:city, timestamp=1648197017890, value=Mumbai
2 row(s) in 0.0540 seconds
```

scan 'customer', {VERSIONS => 5}

```
hbase(main):042:0> scan 'customer', {VERSIONS => 5}
ROW                                     COLUMN+CELL
Justin                                column=address:city, timestamp=1648197260721, value=Pune
Justin                                column=address:state, timestamp=164819725891, value=Maharashtra
Justin                                column=address:street, timestamp=1648197293721, value=street2
Justin                                column=order:amount, timestamp=1648197354548, value=60
Justin                                column=order:number, timestamp=1648197336014, value=ORD-16
Nick                                  column=address:city, timestamp=1648198762680, value=Delhi
Nick                                  column=address:city, timestamp=1648198755614, value=Bangalore
Nick                                  column=address:city, timestamp=1648198734145, value=Pune
Nick                                  column=address:city, timestamp=1648197017890, value=Mumbai
Nick                                  column=address:number, timestamp=1648197149436, value=ORD-15
Nick                                  column=address:state, timestamp=1648197068315, value=Maharashtra
Nick                                  column=order:amount, timestamp=1648197193433, value=50
Nick                                  column=order:number, timestamp=1648197213674, value=ORD-15
2 row(s) in 0.0760 seconds
```

```
hbase(main):043:0> 
```

11. Disable -This command will start disabling the named table

If table needs to be deleted or dropped, it has to disable first

Syntax: **disable <table_name>**

disable 'customer'

```
=> ["customer", "customer1"]
hbase(main):047:0> disable 'customer'
0 row(s) in 0.0720 seconds
```

12. Drop– It drops a table from HBase. Drop means complete deletion of table. For this first disable the table then drop it.

- a. To delete the table present in HBase, first we have to disable it
- b. To drop the table present in HBase, first we have to disable it
- c. So either table to drop or delete first the table should be disable using disable command

Syntax:**drop <table_name>**

drop 'customer'

```
hbase(main):047:0> disable 'customer'  
0 row(s) in 0.0720 seconds  
  
hbase(main):048:0> drop 'customer'  
0 row(s) in 1.3830 seconds  
  
hbase(main):049:0> list  
TABLE  
customer1  
1 row(s) in 0.0220 seconds  
  
=> ["customer1"]  
hbase(main):050:0> ■
```

