

Practical-1

Aim: To implement Basic Linux Commands.

➤ **File Commands**

File Administration

ls [option(s)] [file(s)]

If you run **ls** without any additional parameters, the program will list the contents of the current directory in short form.

-l

detailed list

-a

displays hidden files

cp [option(s)] sourcefile targetfile

Copies sourcefile to targetfile.

-i

Waits for confirmation, if necessary, before an existing targetfile is overwritten

-r

Copies recursively (includes subdirectories)

mv [option(s)] sourcefile targetfile

Copies sourcefile to targetfile then deletes the original sourcefile.

-b

Creates a backup copy of the sourcefile before moving

-i

Waits for confirmation, if necessary, before an existing targetfile is overwritten

rm [option(s)] file(s)

Removes the specified files from the file system. Directories are not removed by **rm** unless the option **-r** is used.

-r

Deletes any existing subdirectories

-i

Waits for confirmation before deleting each file.

ln [option(s)] sourcefile targetfile

Creates an internal link from the sourcefile to the targetfile, under a different name. Normally, such a link points directly to the sourcefile on one and the same file system. However, if **ln** is executed with the **-s** option, it creates a symbolic link that only points to the directory where the sourcefile is located, thus enabling linking across file systems.

-s

Creates a symbolic link

cd [options(s)] [directory]

Changes the current directory. **cd** without any parameters changes to the user's home directory.

mkdir [option(s)] directoryname

Creates a new directory.

rmdir [option(s)] directoryname

Deletes the specified directory, provided it is already empty.

chown [option(s)] username.group file(s)

Transfers the ownership of a file to the user with the specified user name.

-R

Changes files and directories in all subdirectories.

chgrp [option(s)] groupname file(s)

Transfers the group ownership of a given file to the group with the specified group name. The file owner can only change group ownership if a member of both the existing and the new group.

chmod [options] mode file(s)

Changes the access permissions.

The mode parameter has three parts: group, access, and access type. group accepts the following characters:

u

user

g

group

o

others

For access, access is granted by the + symbol and denied by the - symbol.

The access type is controlled by the following options:

r

read

w

write

x

eXecute — executing files or changing to the directory.

s

Set uid bit — the application or program is started as if it were started by the owner of the file.

gzip [parameters] file(s)

This program compresses the contents of files, using complex mathematical algorithms. Files compressed in this way are given the extension .gz and need to be uncompressed before they can be used. To compress several files or even entire directories, use the **tar** command.

-d

decompresses the packed gzip files so they return to their original size and can be processed normally (like the command **gunzip**).

tar options archive file(s)

The **tar** puts one file or (usually) several files into an archive. Compression is optional.

tar is a quite complex command with a number of options available. The most frequently used options are:

-f

Writes the output to a file and not to the screen as is usually the case

-c

Creates a new tar archive

-r

Adds files to an existing archive

-t

Outputs the contents of an archive

-u

Adds files, but only if they are newer than the files already contained in the archive

-x

Unpacks files from an archive (*extraction*)

-Z

Packs the resulting archive with **gzip**

-j

Compresses the resulting archive with **bzip2**

-v

Lists files processed

The archive files created by **tar** end with .tar. If the tar archive was also compressed using **gzip**, the ending is .tgz or .tar.gz. If it was compressed using **bzip2**, .tar.bz2.

Application examples can be found in Section [“Archives and Data Compression”](#).

locate pattern(s)

The **locate** command can find in which directory a specified file is located. If desired, use wild cards to specify file names. The program is very speedy, as it uses a database specifically created for the purpose (rather than searching through the entire file system). This very fact, however, also results in a major drawback: **locate** is unable to find any files created after the latest update of its database.

The database can be generated by root with **updatedb**.

updatedb [options(s)]

This command performs an update of the database used by **locate**. To include files in all existing directories, run the program as root. It also makes sense to place it in the background by appending an ampersand (&), so you can immediately continue working on the same command line (**updatedb &**).

find [option(s)]

The **find** command allows you to search for a file in a given directory. The first argument specifies the directory in which to start the search. The option **-name** must be followed by a search string, which may also include wild cards. Unlike **locate**, which uses a database, **find** scans the actual directory.

➤ **Commands to Access File Contents**

cat [option(s)] file(s)

The **cat** command displays the contents of a file, printing the entire contents to the screen without interruption.

-n

Numbers the output on the left margin

less [option(s)] file(s)

This command can be used to browse the contents of the specified file. Scroll half a screen page up or down with **PgUp** and **PgDn** or a full screen page down with **Space**. Jump to the beginning or end of a file using **Home** and **End**. Press **Q** to exit the program.

grep [option(s)] searchstring filenames

The **grep** command finds a specific searchstring in the specified file(s). If the search string is found, the command displays the line in which the searchstring was found along with the file name.

-i

Ignores case

-l

Only displays the names of the respective files, but not the text lines

-n

Additionally displays the numbers of the lines in which it found a hit

-l

Only lists the files in which searchstring does not occur

diff [option(s)] file1 file2

The **diff** command compares the contents of any two files. The output produced by the program lists all lines that do not match.

This is frequently used by programmers who need only send their program alterations and not the entire source code.

-q

Only reports *whether* the two given files differ

➤ File Systems

mount [option(s)] [<device>] mountpoint

This command can be used to mount any data media, such as hard disks, CD-ROM drives, and other drives, to a directory of the Linux file system.

-r

mount read-only

-t filesystem

Specifies the file system. The most common are ext2 for Linux hard disks, msdos for MS-DOS media, vfat for the Windows file system, and iso9660 for CDs.

For hard disks not defined in the file /etc/fstab, the device type must also be specified. In this case, only root can mount. If the file system should also be mounted by other users, enter the option user in the appropriate line in the /etc/fstab file (separated by commas) and save this change. Further information is available in mount.

umount [option(s)] mountpoint

This command unmounts a mounted drive from the file system. To prevent data loss, run this command before taking a removable data medium from its drive. Normally, only root is allowed to run the commands **mount** and **umount**. To enable other users to run these commands, edit the `/etc/fstab` file to specify the option user for the respective drive.

➤ **System Commands**

- **System Information**

df [option(s)] [directory]

The **df** (disk free) command, when used without any options, displays information about the total disk space, the disk space currently in use, and the free space on all the mounted drives. If a directory is specified, the information is limited to the drive on which that directory is located.

-H

shows the number of occupied blocks in gigabytes, megabytes, or kilobytes — in human-readable format

-t

Type of file system (ext2, nfs, etc.)

du [option(s)] [path]

This command, when executed without any parameters, shows the total disk space occupied by files and subdirectories in the current directory.

-a

Displays the size of each individual file

-h

Output in human-readable form

-s

Displays only the calculated total size

free [option(s)]

The command **free** displays information about RAM and swap space usage, showing the total and the used amount in both categories.

-b

Output in bytes

-k

Output in kilobytes

-m

Output in megabytes

date [option(s)]

This simple program displays the current system time. If run as root, it can also be used to change the system time. Details about the program are available in `date`.

- **Processes**

top [options(s)]

`top` provides a quick overview of the currently running *processes*. Press **H** to access a page that briefly explains the main options to customize the program.

ps [option(s)] [process ID]

If run without any options, this command displays a table of all *your own* programs or processes — those you started. The options for this command are not preceded by hyphen.

`aux`

Displays a detailed list of all processes, independent of the owner.

kill [option(s)] process ID

Unfortunately, sometimes a program cannot be terminated in the normal way. However, in most cases, you should still be able to stop such a runaway program by executing the **kill** command, specifying the respective process ID (see **top** and **ps**).

kill sends a *TERM* signal that instructs the program to shut itself down. If this does not help, the following parameter can be used:

-9

Sends a *KILL* signal instead of a *TERM* signal, with which the process really is *annihilated* by the operating system. This brings the specific processes to an end in almost all cases.

killall [option(s)] processname

This command is similar to **kill**, but uses the process name (instead of the process ID) as an argument, causing all processes with that name to be killed.

- **Network**

ping [option(s)] host name|IP address

The ping command is the standard tool for testing the basic functionality of TCP/IP networks. It sends a small data packet to the destination host, requesting an immediate reply. If this works, ping displays a message to that effect, which indicates that the network link is basically functioning.

-c

number Determines the total number of packages to send and ends after they have been dispatched. By default, there is no limitation set.

-f

flood ping: sends as many data packages as possible. A popular means, reserved to root, to test networks.

-i

value Specifies the interval between two data packages in seconds. Default: one second

nslookup

The Domain Name System resolves domain names to IP addresses. With this tool, send queries to information servers (DNS servers).

telnet [option(s)] host name or IP address

Telnet is actually an Internet protocol that enables you to work on remote hosts across a network. telnet is also the name of a Linux program that uses this protocol to enable operations on remote computers.

➤ **Miscellaneous**

passwd [option(s)] [username]

Users may change their own passwords at any time using this command. Furthermore, the administrator root can use the command to change the password of any user on the system.

su [option(s)] [username]

The **su** command makes it possible to log in under a different user name from a running session. When using the command without specifying a user name, you will be prompted for the root password. Specify a user name and the corresponding password to use the environment of the respective user. The password is not required from root, as root is authorized to assume the identity of any user.

halt [option(s)]

To avoid loss of data, you should always use this program to shut down your system.

reboot [option(s)]

Does the same as **halt** with the difference that the system performs an immediate reboot.

Clear This command cleans up the visible area of the console. It has no options.

Practical-2

Aim: To Implement Hadoop Installation on ubuntu.

Since we know it's the time for parallel computation to tackle large amount of dataset, we will require Apache Hadoop (here the name is derived from Elephant). As Apache Hadoop is the top most contributed Apache project, more and more features are implemented as well as more and more bugs are getting fixed in new coming versions. So, by considering this situation we need to follow slightly different steps than previous version. Here, I am trying to covering full fledged Hadoop installation steps for BigData enthusiasts who wish to install Apache Hadoop on their Ubuntu – Linux machine.

Prerequisites

1. Installing Oracle Java 8

Apache Hadoop is java framework, we need java installed on our machine to get it run over operating system. Hadoop supports all java version greater than 5 (i.e. Java 1.5). So, Here you can also try Java 6, 7 instead of Java 8.

```
sudo apt-get install default-jdk
```

It will install java source in your machine at /usr/lib/jvm/java-8

To verify your java installation, you have to fire the following command like,

```
vignesh@pingax:~$ java -version
```

2. Creating a Hadoop user for accessing HDFS and MapReduce

To avoid security issues, we recommend to setup new Hadoop user group and user account to deal with all Hadoop related activities.

We will create hadoop as system group and hduser as system user by,

```
vignesh@pingax:~$ sudo addgroup hadoop
```

```
vignesh@pingax:~$ sudo adduser --ingroup hadoop hduser
```

3. Installing SSH

SSH ("Secure SHell") is a protocol for securely accessing one machine from another. Hadoop uses SSH for accessing another slaves nodes to start and manage all HDFS and MapReduce daemons.

```
vignesh@pingax:~$ sudo apt-get install openssh-server
```

Now, we have installed SSH over Ubuntu machine so we will be able to connect with this machine as well as from this machine remotely.

Configuring SSH

Once you installed SSH on your machine, you can connect to other machine or allow other machines to connect with this machine. However we have this single machine, we can try connecting with this same machine by SSH. To do this, we need to copy generated RSA key (i.e. id_rsa.pub) pairs to authorized_keys folder of SSH installation of this machine by the following command,

```
# First login with hduser (and from now use only hduser account for further steps)
```

```
vignesh@pingax:~$ sudo su hduser
```

```
# Generate ssh key for hduser account
```

```
hduser@pingax:~$ ssh-keygen -t rsa -P ""
```

```
## Copy id_rsa.pub to authorized keys from hduser
```

```
hduser@pingax:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

In case you are configuring SSH for another machine (i.e. from master node to slave node), you have to update the above command by adding the hostname of slave machine.

Installation Steps

1. Download latest Apache Hadoop source from Apache mirrors

First you need to download [Apache Hadoop 2.6.0](#) (i.e. *hadoop-2.6.0.tar.gz*) or latest version source from Apache download Mirrors. You can also try stable hadoop to get all latest features as well as recent bugs solved with Hadoop source. Choose location where you want to place all your hadoop installation, I have chosen */usr/local/hadoop*

```
## Locate to hadoop installation parent dir
```

```
hduser@pingax:~$ cd /usr/local/
```

Once we have completed this we will need to download Hadoop 2.6 or any version from its official site <http://hadoop.apache.org/> then extract this Hadoop tar.gz manually or through terminal. Now we need to move Hadoop folder to root this step is optional but its recommend that you may move file to root. To move Hadoop folder to its appropriate location use following command (note this command is only use to move folder to root if you are placing to other location you can do it manually).

```
$sudo mv Desktop/hadoop-2.6.0 /usr/local/hadoop
```

```
sudo tar -xzf hadoop-2.6.0.tar.gz
```

```
## Move hadoop-2.6.0 to hadoop folder
```

```
sudo mv hadoop-2.6.0 /usr/local/hadoop
```

```
## Assign ownership of this folder to Hadoop user
```

```
sudo chown hduser:hadoop -R /usr/local/hadoop
```

```
## Create Hadoop temp directories for Namenode and Datanode
```

```
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
```

```
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

Again assign ownership of this Hadoop temp folder to Hadoop user

```
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

Update Hadoop configuration files

Central repository using dynamic data node allocation in multi node HDFS

parameter 1: source Address

parameter 2: destination Address

Note both above address should be complete qualified address

In above command my source address is “Desktop/hadoop-2.6.0” you can change this according to your source location and my destination address is “/usr/local/hadoop”

note- I haven't given '/' after my destination this means that I am renaming my source Folder name from “hadoop-2.6.0” to “hadoop”.

Now we need to set system environment variable so that our system identifies Hadoop. To do this open bashrc file as a root in any text editor.(in my case I am using gedit).

```
$sudo gedit ~/.bashrc
```

Note – some time you get blank file please make sure that this file is ~/.bashrc Append below content to this file.

#Hadoop variables

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

```
export HADOOP_INSTALL=/usr/local/hadoop
```

```
export PATH=$PATH:$HADOOP_INSTALL/bin
```

```
export PATH=$PATH:$HADOOP_INSTALL/sbin
```

```
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
```

```
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
```

```
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
```

```
export YARN_HOME=$HADOOP_INSTALL
```

#end of Hadoop variable declaration

Central repository using dynamic data node allocation in multi node HDFS

Line 1: export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

We are setting Java installation path so that Hadoop can use this path where ever required.

To get your installation path go to /usr/lib search for jvm open folder thee you will get many folders open one without arrow on it (Arrow marked folders are Symbolic links in Linux, similar to shortcuts in windows). That's your installed java.

Line 2: export HADOOP_INSTALL=/usr/local/hadoop

this line is to identify installed location of Java in the system. Note if you have kept this folder in some other location you need to change path accordingly.

Line3 to 8:

these are Hadoop components locations, We are defining these to reduce or work later, I will explain the use of these lines later in depth.

Save and close this ~/.bashrc.

As we have add successfully added the environment variable we need to reflect these to our system for this you can do two things

Central repository using dynamic data node allocation in multi node HDFS

Department of Computer Engineering, MMCOE. Page 48

1.Close all terminals and reopen them as needed.

2.use following command

```
$source ~/.bashrc
```

once you have done this type this command to check we have installed our Hadoop properly or not you can use this command.

```
$hadoop version
```

if you get something like this it means you have successful set up Hadoop in your system.

Now the last thing we need to update JAVA_HOME in Hadoop so open “/hadoop/etc/hadoop/hadoop-env.sh” from you installed Hadoop path and find these line in it

replace this line with your installed java path

Central repository using dynamic data node allocation in multi node HDFS
save it and exit.

In this way we have installed Hadoop 2.6.0 in our Linux.

Note- above all steps are exactly same for installing all 2.x.x versions of Hadoop.

Central repository using dynamic data node allocation in multi node HDFS

Now Hadoop can be use in three different ways

1) Stand Alone Mode

This mode generally does not requires any configuration to be done.

This mode is usually used for Debugging purpose.

All default configuration of Hadoop are done in this mode.

2) Pseudo Distributed Mode (will be Explained in depth Later)

This mode is also called single node mode.

This mode needs little configuration.

This mode is used for Development purpose

3)Distributed Mode (will be Explained in depth Later)

This mode is also called as Multinode node.

This mode needs some changes to be done in Psedudistrbuted mode along with ssh

This mode is generally use for commercial purpose.

Central repository using dynamic data node allocation in multi node HDFS

Department of Computer Engineering, MMCOE. Page 51

10.1.2 Configuring Hadoop 2.6.0 Single Mode/Pseudo Distributed Mode in Linux

Hadoop is by default is configured in Standalone mode. This stand alone mode is used only for debugging purpose but to develops any application we need to configure hadoop in Pseudo Distributed mode.

To configure hadoop in Pseudo Distributed mode we need to edit following files

1)core-site.xml

2)hdfs-site.xml

3)mapred-site.xml

4)yarn-site.xml

Please note that we need to carry out the steps as explained in Previous Document of Setting up hadoop 2.6.0 on Linux.

All mentioned files are present in hadoop installation directory under “/etc/hadoop” in my case as per previous document its address is

“usr/local/hadoop/etc/hadoop”

1) configuring core-site.xml

core site xml is a file containing all core property of hadoop. For example. Namenode url, Temporary storage directory path, etc. Hadoop has predefined configuration which we need to override them if we mention any of the configuration in core-site.xml then during startup of hadoop, hadoop will read these configuration and run hadoop using this. To get more details of default configuration in hadoop you can visit

<https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoopcommon/core-default.xml>

so let us configure some of our requirement.

Open this file in any of the text editor and add these contents in it between

```
<configurations></configurations>
```

```
<property>
```

Central repository using dynamic data node allocation in multi node HDFS

```
<name>fs.defaultFS</name>
```

```
<value>hdfs://localhost:9000</value>
```

```
</property>
```

```
<property>
```

```
<name>hadoop.tmp.dir</name>
```

```
<value>/home/mohamadali/tmp</value>
```

```
</property>
```

Central repository using dynamic data node allocation in multi node HDFS

Department of Computer Engineering, MMCOE. Page 53

Explanation of the above code

property 1: fs.defaultFS

This property overrides the default namenode url its syntax is

hdfs://<ip-address of namenode>:<port number>. This property was named as fs.default.name in hadoop 1.x.x version. Note: Port number can be any number above 255 to 65536

property 2: hadoop.tmp.dir

This property is used to change the temporary storage

directory during execution of any algorithm in hadoop by default its

location is “/tmp/hadoop-\${user.name}” in my case I have created this directory in my home folder name tmp so its “/home/mohamadali/tmp”.

2) Configuring hdfs-site.xml

This file contains all configuration about hadoop distributed file system also called as HDFS such as storage location for namenode, storage location for datanode, replication factor of HDFS, etc.

Similar to core-site.xml we need to place below content between configuration fields to get more information on this you can visit above mentioned link.

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>1</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.namenode.name.dir</name>
```

```
<value>/home/mohamadali/tmp/namenode</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.datanode.data.dir</name>
```

```
<value>/home/mohamadali/tmp/datanode</value>
</property>
```

Central repository using dynamic data node allocation in multi node HDFS

Explanation of above properties in detail.

Property 1: dfs.replication

This property overrides the replication factor in hadoop. By default its value is 3 but in single node cluster it is recommended to be 1.

Property 2: dfs.namenode.name.dir

Central repository using dynamic data node allocation in multi node HDFS

This property overrides storage location of namenode data by default its storage location is inside “/tmp/hadoop-\${user.name}”. To change this you have set value of your folder location in my case it is inside tmp directory created during core-site.xml

Property 3: dfs.datanode.data.dir

This property overrides storage location of datanode data by default its storage location is inside “/tmp/hadoop-\${user.name}”. To change this you have set value of your folder location in my case it is also inside tmp directory created during core-site.xml

Note: for property 1 and property 2

Please make sure if your location of both datanode and namenode is in your root directory then you should change its ownership and read write access using chown and chmod command in Linux. Also you can create these directory manually before this setting them to your path else hadoop will create them for you.

3) Configuring mapred-site.xml

This file contain all configuration about Map Reduce component in hadoop. Please note that this file doesn't exist but you can copy or rename it from mapred-site.xml.template. Configuration for this file is should be as followed.

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

Explanation of above property

As we know that from hadoop 2.x.x hadoop has introduced new layer of technology developed by hadoop to improve performance of map reduce algorithm this layer is called as “yarn” that is Yet Another Resource Negotiator. So here we are configuring that our hadoop framework is yarn if Central repository using dynamic data node allocation in multi node HDFS Department of Computer Engineering, MMCOE. Page 56 we don't specify this property then our hadoop will use Map reduce 1 also called as MR1.

4) Configuring yarn-site.xml

This file contains all information about YARN as we will be using MR2 we need to specify the auxiliary services that need to be used with MR2 so add these lines to yarn-site.xml

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

Now we have successfully configured hadoop

Format Namenode

```
hduser@pingax:hdfs namenode -format
```

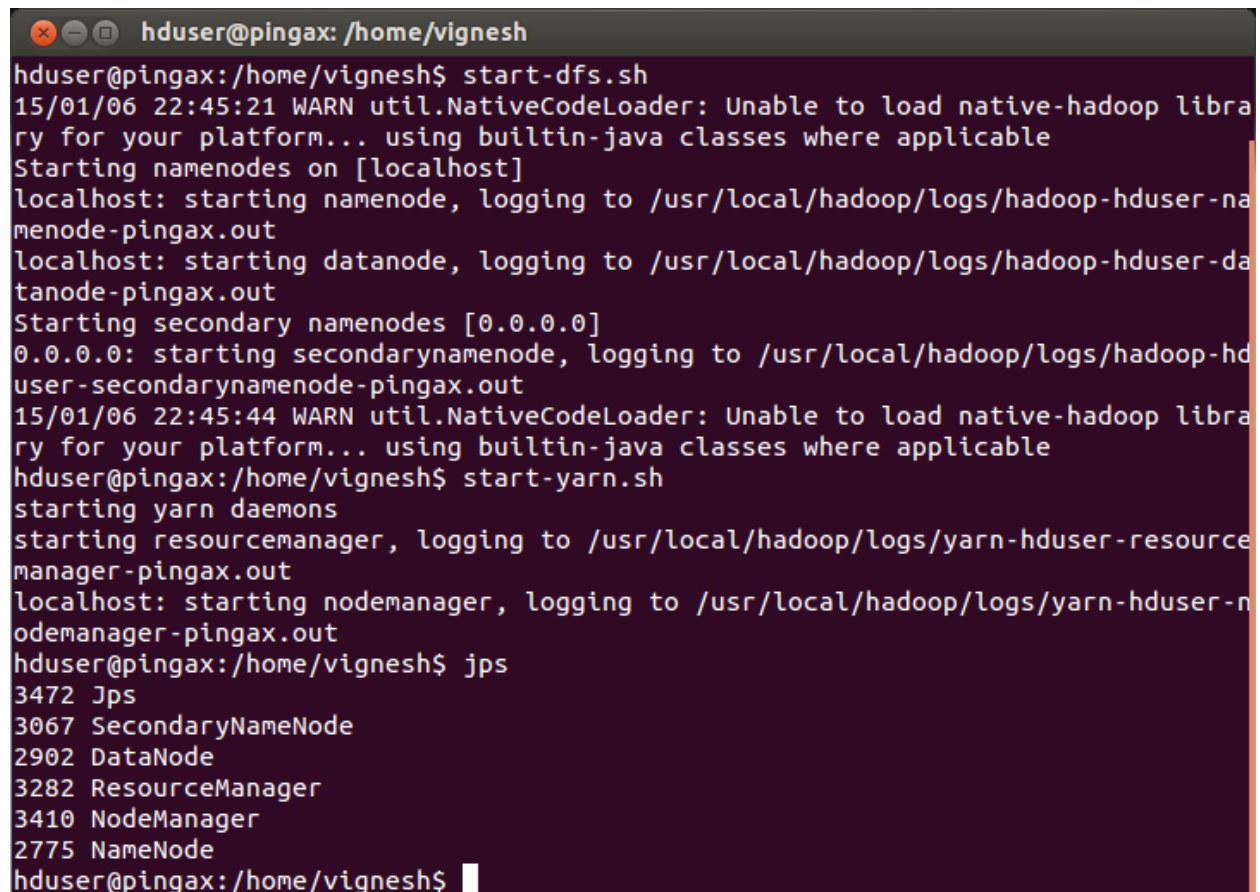
Start all Hadoop daemons

use *start-all.sh*

Track/Monitor/Verify

Verify Hadoop daemons:

```
hduser@pingax: jps
```

A terminal window titled 'hduser@pingax: /home/vignesh' showing the execution of Hadoop startup scripts. The output includes warnings about native code loading, logs for starting namenodes and datanodes, and the output of the 'jps' command listing the running daemons: Jps, SecondaryNameNode, DataNode, ResourceManager, NodeManager, and NameNode.

```
hduser@pingax:/home/vignesh$ start-dfs.sh
15/01/06 22:45:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-pingax.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-pingax.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-pingax.out
15/01/06 22:45:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hduser@pingax:/home/vignesh$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-pingax.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-pingax.out
hduser@pingax:/home/vignesh$ jps
3472 Jps
3067 SecondaryNameNode
2902 DataNode
3282 ResourceManager
3410 NodeManager
2775 NameNode
hduser@pingax:/home/vignesh$
```

Practical-3**Aim: Configuring Hadoop in Single Mode/Pseudo Distributed Mode in Ubuntu.**

Hadoop can be used in three different ways

- 1) Stand Alone Mode - This mode generally does not requires any configuration to be done.This mode is usually used for Debugging purpose.All default configuration of Hadoop are done in this mode.
- 2) Pseudo Distributed Mode - This mode is also called single node mode.This mode needs little configuration.This mode is used for Development purpose.
- 3)Distributed Mode - This mode is also called as Multinode node.This mode needs some changes to be done in Psedudistributed modealong with ssh. This mode is generally used for commercial purpose.

Hadoop is by default configured in Standalone mode. This standalone modeis used only for debugging purpose but to develop any application we need toconfigure hadoop in Pseudo Distributed mode.

To configure hadoop in Pseudo Distributed mode we need to edit followingfiles:

- 1)core-site.xml 2)hdfs-site.xml 3)mapred-site.xml 4)yarn-site.xml

All mentioned files are present in hadoop installation directory under“/etc/hadoop” in this case as per previous document its address is as per below: “/usr/local/hadoop/etc/hadoop”

1) configuring core-site.xml

core site.xml is a file containing all core properties of hadoop. For example, Namenodeurl,Temporary storage directory path, etc. Hadoop has predefined configuration which we need tooverride them if we mention any of the configuration in core-site.xml then during startup of hadoop, hadoop will read these configuration an run hadoop using this. To get more details of defaultconfiguration in hadoop you can visit [https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoopcommon/](https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoopcommon/core-default.xml) core-default.xml. So configure as per requirement.

Open this file in any of the text editor and add these contents in it between<configurations></configurations>

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/usr/local/hadoop/tmp</value>
</property>
```

property 1: fs.defaultFS

This property overrides the default namenodeurl its syntax is hdfs://<ip-address of namenode>:<port number> .This property was named as fs.default.name in hadoop 1.x.x version. Note: Port number can be any number above 255 to 65536

property 2: hadoop.tmp.dir

This property is used to change the temporary storage directory during execution of any algorithm in hadoop by default its location is “/tmp/hadoop-\${user.name}” in my case it is“/usr/local/hadoop/tmp”.

2) Configuringhdfs-site.xml

This file contains all configuration about hadoop distributed filesystem also called as HDFS such as storage location for namenode, storage location for datanode, replication factor of HDFS, etc. Similar to core-site.xml we need to place below content between configuration fields to get more information on this you can visit above mentioned link.

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/hadoop/tmp/namenode</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>/usr/local/hadoop/tmp/datanode</value>
</property>
```

Property 1:dfs.replication

This property overrides the replication factor in hadoop. By default its value is 3 but in single node cluster it is recommended to be 1.

Property 2:dfs.namenode.name.dir

This property overrides storage location of namenode data by default its storage location is inside “/tmp/hadoop-\${user.name}”. To change this you have set value of your folder location in my case it is inside tmp directory created during core-site.xml

Property 3:dfs.datanode.data.dir

This property overrides storage location of datanode data by default its storage location is inside “/tmp/hadoop-\${user.name}”. To change this you have set value of your folder location in my case it is also inside tmp directory created during core-site.xml.

Note: for property 1 and property 2, Please make sure if your location of both datanode and namenode is in your root directory then you should change its ownership and read write access using chown and chmod command in Linux. Also you can create these directory manually before this setting them to your path ,else hadoop will create them for you.

3) Configuringmapred-site.xml

This file contains all configuration about Map Reduce component in hadoop. Please note that this file doesn't exist but you can copy or rename it from mapred-site.xml.template. Configuration for this file should be as followed.

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
```

As we know that from hadoop 2.x.x hadoop has introduced new layer of technology developed by hadoop to improve performance of mapreduce algorithm this layer is called as “yarn” that is Yet Another Resource Negotiator. So here we are configuring that our hadoop framework is yarn if we don't specify this property then our hadoop will use Map reduce 1 also called as MR1.

4) Configuring yarn-site.xml

This file contains all information about YARN as we will be using MR2 we need to specify the auxiliary services that need to be used with MR2 so add these lines to yarn-site.xml.

```
<property>  
    <name>yarn.nodemanager.aux-services</name>  
    <value>mapreduce_shuffle</value>  
</property>
```

Now we have successfully configured Hadoop 2.7.3 in Pseudo distributed mode. Before starting Hadoop we need to format our namenode. Execute below command to format namenode.

```
$hdfs namenode-format
```

Now to start Hadoop you can use two commands.

```
$start-dfs.sh
```

```
$start-yarn.sh
```

or you can also use deprecated commands

```
$start-all.sh
```

To check which components are working you can use below command

```
$ jps
```

you will get output as

```
hduser@VIE300236:~$ jps  
4563 NameNode  
5189 NodeManager  
4710 DataNode  
4902 SecondaryNameNode  
5062 ResourceManager  
5499 Jps  
hduser@VIE300236:~$
```

Practical 4

Aim: Implement basic Hadoop Commands.

mkdir

Usage: `hadoopfs -mkdir [-p] <paths>`

Takes path uri's as argument and creates directories.

Options:

- The -p option behavior is much like Unix `mkdir -p`, creating parent directories along the path.

Example:

- `hadoopfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2`
- `hadoopfs -mkdir hdfs://nn1.example.com/user/hadoop/dir`
`hdfs://nn2.example.com/user/hadoop/dir`

Exit Code: Returns 0 on success and -1 on error.

ls

Usage: `hadoopfs -ls [-d] [-h] [-R] <args>`

Options:

- -d: Directories are listed as plain files.
- -h: Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).
- -R: Recursively list subdirectories encountered.

For a file ls returns stat on the file with the following format:

permissions number_of_replicas user id group id file size modification_date modification_time filename

For a directory it returns list of its direct children as in Unix. A directory is listed as:

permissions user id group id modification_date modification_time dir name

Files within a directory are order by filename by default.

Example:

- `hadoopfs -ls /user/hadoop/file1`

Exit Code: Returns 0 on success and -1 on error.

put

Usage: `hadoopfs -put <localsrc> ... <dst>`

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

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

Exit Code: Returns 0 on success and -1 on error.

get

Usage: `hadoopfs -get [-ignorecrc] [-crc] <src><localdst>`

Copy files to the local file system. Files that fail the CRC check may be copied with the -ignorecrc option.

Files and CRCs may be copied using the -crc option.

Example:

- `hadoopfs -get /user/hadoop/file localfile`
- `hadoopfs -get hdfs://nn.example.com/user/hadoop/file localfile`

Exit Code: Returns 0 on success and -1 on error.

copyFromLocal

Usage: `hadoopfs -copyFromLocal <localsrc> URI`

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

Options:

- The -f option will overwrite the destination if it already exists.

copyToLocal

Usage: `hadoopfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>`

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

count

Usage: `hadoopfs -count [-q] [-h] [-v] <paths>`

Count the number of directories, files and bytes under the paths that match the specified file pattern. The output columns with -count are: DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME

The output columns with -count -q are: QUOTA, REMAINING_QUOTA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME

The -h option shows sizes in human readable format.

The -v option displays a header line.

Example:

- `hadoopfs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2`
- `hadoopfs -count -q hdfs://nn1.example.com/file1`
- `hadoopfs -count -q -h hdfs://nn1.example.com/file1`
- `hdfsdfs -count -q -h -v hdfs://nn1.example.com/file1`

Exit Code: Returns 0 on success and -1 on error.

cp

Usage: `hadoopfs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>`

Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

'raw.*' namespace extended attributes are preserved if (1) the source and destination filesystems support them (HDFS only), and (2) all source and destination pathnames are in the /.reserved/raw hierarchy.

Determination of whether raw.* namespace xattrs are preserved is independent of the -p (preserve) flag.

Options:

- The -f option will overwrite the destination if it already exists.
- The -p option will preserve file attributes [topx] (timestamps, ownership, permission, ACL, XAttr). If -p is specified with no *arg*, then preserves timestamps, ownership, permission. If -pa is specified, then preserves permission also because ACL is a super-set of permission. Determination of whether raw namespace extended attributes are preserved is independent of the -p flag.

Example:

- `hadoopfs -cp /user/hadoop/file1 /user/hadoop/file2`
- `hadoopfs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir`

Exit Code: Returns 0 on success and -1 on error.

moveFromLocal

Usage: `hadoopfs -moveFromLocal<localsrc><dst>`

Similar to put command, except that the source localsrc is deleted after it's copied.

moveToLocal

Usage: `hadoopfs -moveToLocal [-crc] <src><dst>`

Displays a "Not implemented yet" message.

mv

Usage: `hadoopfs -mv URI [URI ...] <dest>`

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

Example:

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

Exit Code: Returns 0 on success and -1 on error.

rm

Usage: `hadoopfs -rm [-f] [-r|-R] [-skipTrash] URI [URI ...]`

Delete files specified as args.

If trash is enabled, file system instead moves the deleted file to a trash directory (given by `FileSystem#getTrashRoot`).

Currently, the trash feature is disabled by default. User can enable trash by setting a value greater than zero for parameter `fs.trash.interval` (in `core-site.xml`).

Options:

- The `-f` option will not display a diagnostic message or modify the exit status to reflect an error if the file does not exist.
- The `-R` option deletes the directory and any content under it recursively.
- The `-r` option is equivalent to `-R`.
- The `-skipTrash` option will bypass trash, if enabled, and delete the specified file(s) immediately. This can be useful when it is necessary to delete files from an over-quota directory.

Example:

- `hadoopfs -rm hdfs://nn.example.com/file /user/hadoop/emptydir`

Exit Code: Returns 0 on success and -1 on error.

rmdir

Usage: `hadoopfs -rmdir [--ignore-fail-on-non-empty] URI [URI ...]`

Delete a directory.

Options:

- `--ignore-fail-on-non-empty`: When using wildcards, do not fail if a directory still contains files.

Example:

- `hadoopfs -rmdir /user/hadoop/emptydir`

setrep

Usage: `hadoopfs -setrep [-R] [-w] <numReplicas><path>`

Changes the replication factor of a file. If *path* is a directory then the command recursively changes the replication factor of all files under the directory tree rooted at *path*.

Options:

- The `-w` flag requests that the command wait for the replication to complete. This can potentially take a very long time.
- The `-R` flag is accepted for backwards compatibility. It has no effect.

Example:

- `hadoopfs -setrep -w 3 /user/hadoop/dir1`

Exit Code: Returns 0 on success and -1 on error.

stat

Usage: `hadoopfs -stat [format] <path> ...`

Print statistics about the file/directory at *<path>* in the specified format. Format accepts filesize in blocks (%b), type (%F), group name of owner (%g), name (%n), block size (%o), replication (%r), user name of

owner(%u), and modification date (%y, %Y). %y shows UTC date as “yyyy-MM-ddHH:mm:ss” and %Y shows milliseconds since January 1, 1970 UTC. If the format is not specified, %y is used by default.

Example:

- `hadoopfs -stat "%F %u:%g %b %y %n" /file`

Exit Code: Returns 0 on success and -1 on error.

text

Usage: `hadoopfs -text <src>`

Takes a source file and outputs the file in text format. The allowed formats are zip and TextRecordInputStream.

touchz

Usage: `hadoopfs -touchz URI [URI ...]`

Create a file of zero length.

Example:

- `hadoopfs -touchz pathname`

Exit Code: Returns 0 on success and -1 on error.

usage

Usage: `hadoopfs -usage command`

Return the help for an individual command.

expunge

Usage: `hadoopfs -expunge`

Permanently delete files in checkpoints older than the retention threshold from trash directory, and create new checkpoint.

When checkpoint is created, recently deleted files in trash are moved under the checkpoint. Files in checkpoints older than `fs.trash.checkpoint.interval` will be permanently deleted on the next invocation of `-expunge` command.

If the file system supports the feature, users can configure to create and delete checkpoints periodically by the parameter stored as `fs.trash.checkpoint.interval` (in `core-site.xml`). This value should be smaller or equal to `fs.trash.interval`.

find

Usage: `hadoopfs -find <path> ... <expression> ...`

Finds all files that match the specified expression and applies selected actions to them. If no *path* is specified then defaults to the current working directory. If no expression is specified then defaults to `-print`.

The following primary expressions are recognised:

- `-name pattern`
`-iname pattern`
Evaluates as true if the basename of the file matches the pattern using standard file system globbing. If `-iname` is used then the match is case insensitive.
- `-print`
`-print0Always`
evaluates to true. Causes the current pathname to be written to standard output. If the `-print0` expression is used then an ASCII NULL character is appended.

The following operators are recognised:

- `expression -a expression`
`expression -and expression`
`expressionexpression`

Logical AND operator for joining two expressions. Returns true if both child expressions return true. Implied by the juxtaposition of two expressions and so does not need to be explicitly specified. The second expression will not be applied if the first fails.

Example:

`hadoopfs -find / -name test -print`

Exit Code:

Returns 0 on success and -1 on error.

Practical-5

Aim: To implement word count using Mapreduce.

```
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 {

    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);
            }
        }
    }

    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);
        }
    }
}
```

```
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);  
}  
}
```

Commands to execute the program:

```
hadoop com.sun.tools.javac.Main WordCount.java
```

```
cd /home/hduser/Desktop/
```

```
hadoop com.sun.tools.javac.Main WordCount.java
```

```
jar cf wc.jar WordCount*.class
```

```
hadoop fs -mkdir /vie
```

```
hadoop fs -mkdir /vie/input /vie/output
```

```
hadoop fs -ls /vie
```

```
hadoop fs -put 'sample.txt' /vie/input
```

```
hadoop fs -ls /vie
```

```
hadoop fs -ls /vie/input
```

```
hadoop jar wc.jar WordCount /vie/input /vie/output
```

```
hadoop jar wc.jar WordCount /vie/input /vie/output1
```

```
hadoop fs -ls /vie/output1
```

```
hadoop fs -cat /vie/output1/part-r-00000
```

Practical 6

Aim: Implement a Mapreduce Application Program for the solution of Sudoku Puzzle.

```
//package org.apache.hadoop.examples.dancing;

import java.io.*;
import java.util.*;

import com.google.common.base.Charsets;

/**
 * This class uses the dancing links algorithm from Knuth to solve sudoku
 * puzzles. It has solved 42x42 puzzles in 1.02 seconds.
 */
public class Sudoku {

    /**
     * The preset values in the board
     * board[y][x] is the value at x,y with -1 = any
     */
    private int[][] board;

    /**
     * The size of the board
     */
    private int size;

    /**
     * The size of the sub-squares in cells across
     */
    private int squareXSize;

    /**
     * The size of the sub-squares in cells up and down
     */
    private int squareYSize;

    /**
     * This interface is a marker class for the columns created for the
     * Sudoku solver.
     */
    protected static interface ColumnName {
        // NOTHING
    }

    /**
     * A string containing a representation of the solution.
     * @param size the size of the board
     * @param solution a list of list of column names
     * @return a string of the solution matrix
     */
}
```

```

    */
    static String stringifySolution(int size, List<List<ColumnName>> solution) {
        int[][] picture = new int[size][size];
        StringBuffer result = new StringBuffer();
        // go through the rows selected in the model and build a picture of the
        // solution.
        for(List<ColumnName> row: solution) {
            int x = -1;
            int y = -1;
            intnum = -1;
            for(ColumnName item: row) {
                if (item instanceof ColumnConstraint) {
                    x = ((ColumnConstraint) item).column;
                    num = ((ColumnConstraint) item).num;
                } else if (item instanceof RowConstraint) {
                    y = ((RowConstraint) item).row;
                }
            }
            picture[y][x] = num;
        }
        // build the string
        for(int y=0; y < size; ++y) {
            for (int x=0; x < size; ++x) {
                result.append(picture[y][x]);
                result.append(" ");
            }
            result.append("\n");
        }
        return result.toString();
    }

    /**
     * An acceptor to get the solutions to the puzzle as they are generated and
     * print them to the console.
     */
    private static class SolutionPrinter
        implements DancingLinks.SolutionAcceptor<ColumnName> {
        int size;

        public SolutionPrinter(int size) {
            this.size = size;
        }

        /**
         * A debugging aid that just prints the raw information about the
         * dancing link columns that were selected for each row.
         * @param solution a list of list of column names
         */
        void drawWrite(List solution) {
            for (Iterator itr=solution.iterator(); itr.hasNext(); ) {
                Iterator subitr = ((List) itr.next()).iterator();

```

```
while (subitr.hasNext()) {
    System.out.print(subitr.next().toString() + " ");
}
System.out.println();
}
}

public void solution(List<List<ColumnName>> names) {
    System.out.println(stringifySolution(size, names));
}
}

/**
 * Set up a puzzle board to the given size.
 * Boards may be asymmetric, but the squares will always be divided to be
 * more cells wide than they are tall. For example, a 6x6 puzzle will make
 * sub-squares that are 3x2 (3 cells wide, 2 cells tall). Clearly that means
 * the board is made up of 2x3 sub-squares.
 * @param stream The input stream to read the data from
 */
public Sudoku(InputStream stream) throws IOException {
    BufferedReader file = new BufferedReader(
        new InputStreamReader(stream, Charsets.UTF_8));
    String line = file.readLine();
    List<int[]> result = new ArrayList<int[]>();
    while (line != null) {
        StringTokenizer tokenizer = new StringTokenizer(line);
        int size = tokenizer.countTokens();
        int[] col = new int[size];
        int y = 0;
        while(tokenizer.hasMoreElements()) {
            String word = tokenizer.nextToken();
            if ("?".equals(word)) {
                col[y] = - 1;
            } else {
                col[y] = Integer.parseInt(word);
            }
            y += 1;
        }
        result.add(col);
        line = file.readLine();
    }
    size = result.size();
    board = result.toArray(new int [size][]);
    squareYSize = (int) Math.sqrt(size);
    squareXSize = size / squareYSize;
    file.close();
}

/**
 * A constraint that each number can appear just once in a column.
```

```
*/
static private class ColumnConstraint implements ColumnName {
    ColumnConstraint(intnum, int column) {
        this.num = num;
        this.column = column;
    }
    intnum;
    int column;
    public String toString() {
        return num + " in column " + column;
    }
}

/**
 * A constraint that each number can appear just once in a row.
 */
static private class RowConstraint implements ColumnName {
    RowConstraint(intnum, int row) {
        this.num = num;
        this.row = row;
    }
    intnum;
    int row;
    public String toString() {
        return num + " in row " + row;
    }
}

/**
 * A constraint that each number can appear just once in a square.
 */
static private class SquareConstraint implements ColumnName {
    SquareConstraint(intnum, int x, int y) {
        this.num = num;
        this.x = x;
        this.y = y;
    }
    intnum;
    int x;
    int y;
    public String toString() {
        return num + " in square " + x + "," + y;
    }
}

/**
 * A constraint that each cell can only be used once.
 */
static private class CellConstraint implements ColumnName {
    CellConstraint(int x, int y) {
        this.x = x;
```

```

this.y = y;
    }
int x;
int y;
public String toString() {
return "cell " + x + "," + y;
    }
}

/**
 * Create a row that places num in cell x, y.
 * @param rowValues a scratch pad to mark the bits needed
 * @param x the horizontal offset of the cell
 * @param y the vertical offset of the cell
 * @param num the number to place
 * @return a bitvector of the columns selected
 */
private boolean[] generateRow(boolean[] rowValues, int x, int y, int num) {
    // clear the scratch array
    for(int i=0; i < rowValues.length; ++i) {
        rowValues[i] = false;
    }
    // find the square coordinates
    int xBox = x / squareXSize;
    int yBox = y / squareYSize;
    // mark the column
    rowValues[x*size + num - 1] = true;
    // mark the row
    rowValues[size*size + y*size + num - 1] = true;
    // mark the square
    rowValues[2*size*size + (xBox*squareXSize + yBox)*size + num - 1] = true;
    // mark the cell
    rowValues[3*size*size + size*x + y] = true;
    return rowValues;
}

private DancingLinks<ColumnName> makeModel() {
    DancingLinks<ColumnName> model = new DancingLinks<ColumnName>();
    // create all of the columns constraints
    for(int x=0; x < size; ++x) {
        for(int num=1; num <= size; ++num) {
            model.addColumn(new ColumnConstraint(num, x));
        }
    }
    // create all of the row constraints
    for(int y=0; y < size; ++y) {
        for(int num=1; num <= size; ++num) {
            model.addColumn(new RowConstraint(num, y));
        }
    }
    // create the square constraints

```



```

for(int x=0; x < squareYSize; ++x) {
for(int y=0; y < squareXSize; ++y) {
for(int num=1; num <= size; ++num) {
model.addColumn(new SquareConstraint(num, x, y));
    }
    }
    }
    // create the cell constraints
for(int x=0; x < size; ++x) {
for(int y=0; y < size; ++y) {
model.addColumn(new CellConstraint(x, y));
    }
    }
boolean[] rowValues = new boolean[size*size*4];
for(int x=0; x < size; ++x) {
for(int y=0; y < size; ++y) {
if (board[y][x] == -1) {
    // try each possible value in the cell
for(int num=1; num <= size; ++num) {
model.addRow(generateRow(rowValues, x, y, num));
    }
    } else {
    // put the given cell in place
model.addRow(generateRow(rowValues, x, y, board[y][x]));
    }
    }
    }
return model;
}

public void solve() {
DancingLinks<ColumnName> model = makeModel();
int results = model.solve(new SolutionPrinter(size));
System.out.println("Found " + results + " solutions");
}

/**
 * Solves a set of sudoku puzzles.
 * @param args a list of puzzle filenames to solve
 */
public static void main(String[] args) throws IOException {
if (args.length == 0) {
System.out.println("Include a puzzle on the command line.");
}
for(int i=0; i < args.length; ++i) {
    Sudoku problem = new Sudoku(new FileInputStream(args[i]));
System.out.println("Solving " + args[i]);
problem.solve();
}
}

```

}

Explanation:

Go to the directory where mapreduce examples reside using below command:

```
hduser@VIE300212:/$ cd /usr/local/hadoop/share/hadoop/mapreduce
```

Puzzle1.dta

9 lines (9 sloc) 161 Bytes

```
? 5 ? 3 9 ? ? ? ?
? ? 2 ? ? ? ? ? ?
? ? 6 ? 1 ? ? ? 2
? ? 4 ? ? 3 ? 5 9
? ? 8 9 ? 1 4 ? ?
3 2 ? 4 ? ? 8 ? ?
9 ? ? ? 8 ? 5 ? ?
? ? ? ? ? ? 2 ? ?
? ? ? ? 4 5 ? 7 8
```

Use the following command to list the samples:

```
hduser@VIE300212:/usr/local/hadoop/share/hadoop/mapreduce$ yarn jar hadoop-mapreduce-examples-2.7.3.jar
```

Use the following command to get help on a specific sample. In this case, the sudoku sample:

```
hduser@VIE300212:/usr/local/hadoop/share/hadoop/mapreduce$ yarn jar hadoop-mapreduce-examples-2.7.3.jar sudoku sources/puzzle1.dta
```

```
hduser@VIE300212: /usr/local/hadoop/share/hadoop/mapreduce
hduser@VIE300212:/usr/local/hadoop/share/hadoop/mapreduce$ yarn jar hadoop-mapreduce-examples-2.7.3.jar sudoku sources/puzzle1.dta
Solving sources/puzzle1.dta
4 5 1 3 9 2 6 8 7
8 3 2 6 7 4 1 9 5
7 9 6 5 1 8 3 4 2
6 1 4 8 2 3 7 5 9
5 7 8 9 6 1 4 2 3
3 2 9 4 5 7 8 1 6
9 4 7 2 8 6 5 3 1
1 8 5 7 3 9 2 6 4
2 6 3 1 4 5 9 7 8

8 5 1 3 9 2 6 4 7
4 3 2 6 7 8 1 9 5
7 9 6 5 1 4 3 8 2
6 1 4 8 2 3 7 5 9
5 7 8 9 6 1 4 2 3
3 2 9 4 5 7 8 1 6
9 4 7 2 8 6 5 3 1
1 8 5 7 3 9 2 6 4
2 6 3 1 4 5 9 7 8

Found 2 solutions
```

Practical 7

Aim: Implement installation and configuration of Hive package on Hadoop framework

Hive-Introduction

Hive can be defined as follows:

Hive is a data warehouse system for Hadoop that facilitates ad-hoc queries and the analysis of large data sets stored in Hadoop.

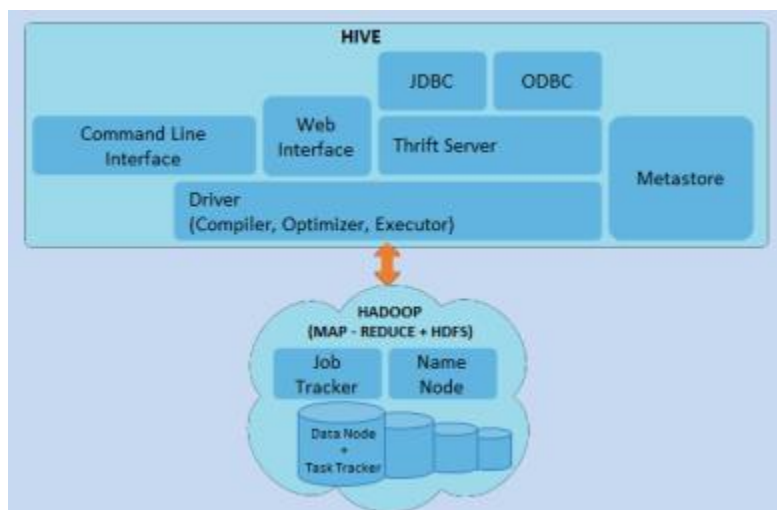
Following are the facts related to Hive:

- It provides a SQL-like language called HiveQL (HQL). Due to its SQL-like interface, Hive is a popular choice for Hadoop analytics.
- It provides massive scale-out and fault tolerance capabilities for data storage and processing of commodity hardware.

Relying on MapReduce for execution, Hive is batch-oriented and has high latency for query execution.

System Architecture and Components of Hive

The image illustrates the architecture of the Hive system. It also displays the role of Hive and Hadoop in the development process.

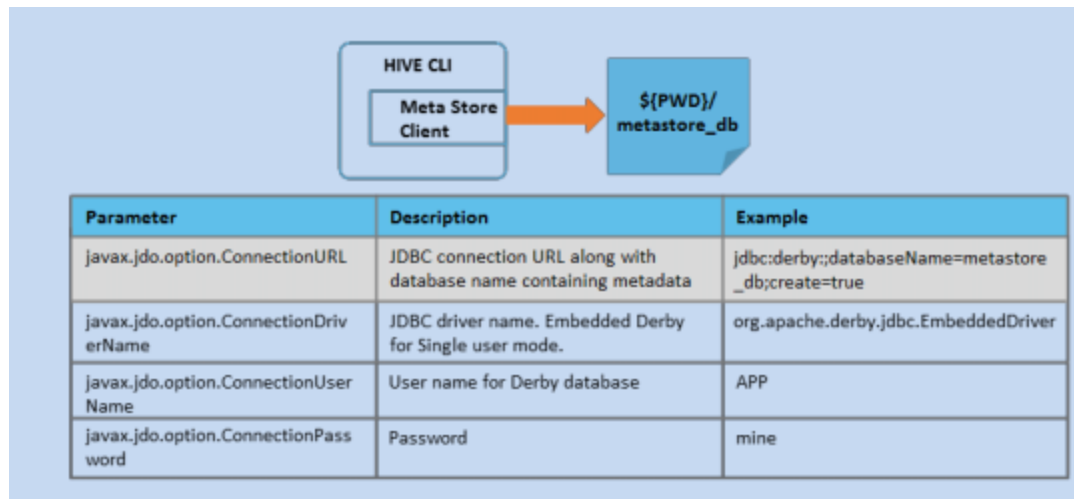


Meta-store

Metastore is the component that stores the system catalog and metadata about tables, columns, partitions, and so on. Metadata is stored in a traditional RDBMS format. Apache Hive uses Derby database by default. Any JDBC compliant database like MySQL can be used for metastore.

Metastore Configuration

The key attributes that should be configured for Hive metastore are given below:



Metastore Configuration—Template

The hive-site.xml file is used to configure the metastore. A template for the file is displayed here.

Driver

Driver is the component that:

- manages the lifecycle of a Hive Query Language (HiveQL) statement as it moves through Hive;
- maintains a session handle and any session statistics.

```
<configuration>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://db1.mydomain.pvt/hive_db?createDatabaseIfNotExist=true</value>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>database_user</value>
</property>
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>database_pass</value>
</property>
</configuration>
```

HiveInstallation—Step1

Locate the Hive tar file for the latest version. You will be using Hive 0.13.1 version.



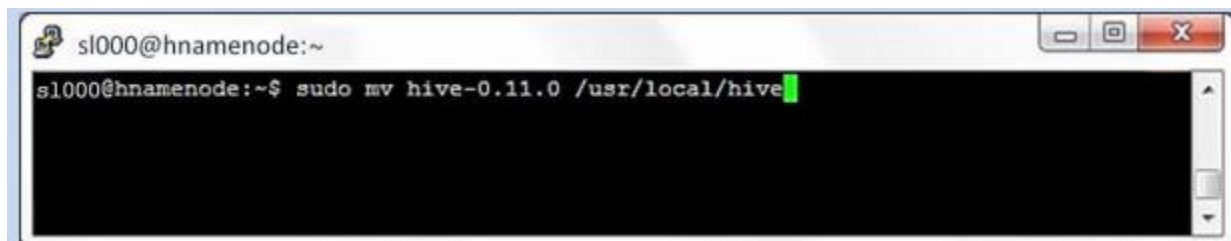
Hive Installation—Step 2

Download the Hive tar file in Ubuntu system using the wget command.



Hive Installation—Step 3

Move the extracted tar file to /usr/local/hive.



Hive Installation—Step 4

Move the extracted folder and set the path for Hive.

Running Hive

To start Hive, type 'hive' in developer machine at shell prompt



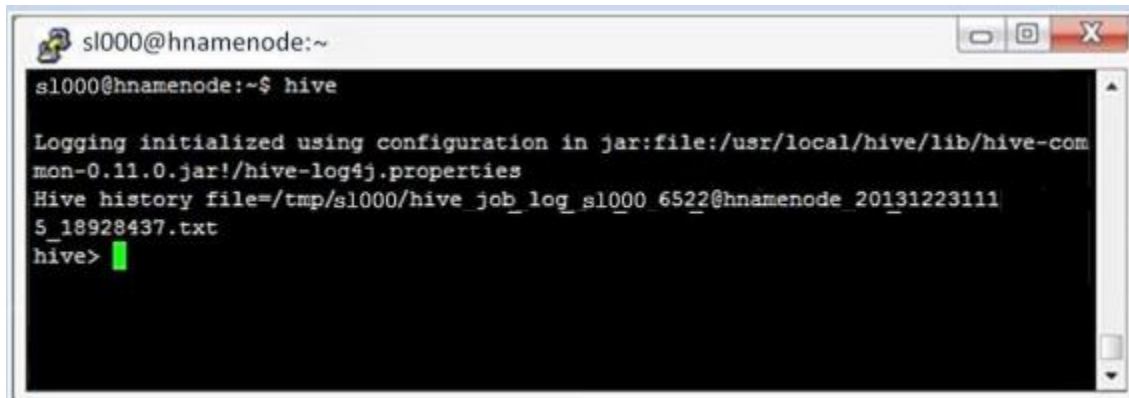
```
sl000@hnamenode:~  
sl000@hnamenode:~$ export HIVE_PREFIX=/usr/local/hive
```



```
sl000@hnamenode:~  
sl000@hnamenode:~$ export PATH=$PATH:$HIVE_PREFIX/bin
```

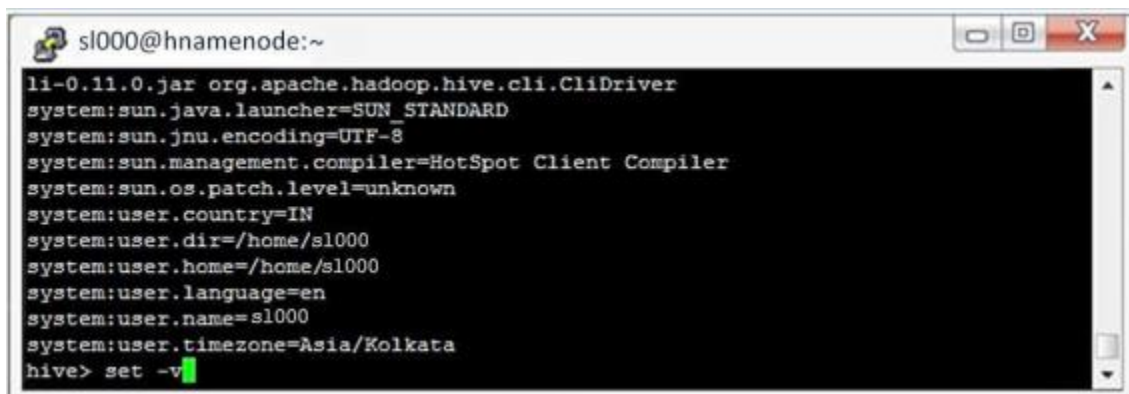
Running Hive

To start Hive, type 'hive' in developer machine at shell prompt.



```
sl000@hnamenode:~  
sl000@hnamenode:~$ hive  
  
Logging initialized using configuration in jar:file:/usr/local/hive/lib/hive-com  
mon-0.11.0.jar!/hive-log4j.properties  
Hive history file=/tmp/sl000/hive_job_log_sl000_6522@hnamenode_20131223111  
5_18928437.txt  
hive>
```

To know the Hive properties, set the 'set -v' command in your system.



```
sl000@hnamenode:~  
li-0.11.0.jar org.apache.hadoop.hive.cli.CliDriver  
system:sun.java.launcher=SUN_STANDARD  
system:sun.jnu.encoding=UTF-8  
system:sun.management.compiler=HotSpot Client Compiler  
system:sun.os.patch.level=unknown  
system:user.country=IN  
system:user.dir=/home/sl000  
system:user.home=/home/sl000  
system:user.language=en  
system:user.name=sl000  
system:user.timezone=Asia/Kolkata  
hive> set -v
```

Practical 8

Aim: To store the basic information about students such as roll no, name, date of birth , subjects and corresponding marks using various collection type such as Map.

There are three types of collections that Cassandra supports.

1. Set

A Set stores group of elements that returns sorted elements when querying.

Syntax

Here is the syntax of the Set collection that store multiple email addresses for the teacher.


Create table University.Teacher

```
(  
id int,  
Name text,  
Email set<text>,  
Primary key(id)  
);
```

Execution


Here is the snapshot where table "Teacher" is created with "Email" column as a collection.

```
cqlsh> Create table University.Teacher  
...  
... id int,  
... Name text,  
... Email set<text>,  
... Primary key(id)  
... );  
cqlsh>
```



Here is the snapshot where data is being inserted in the collection.

```
cqlsh> insert into University.Teacher(id,Name,Email) values(1,'Guru99',{'abc@gma  
il.com','xyz@hotmail.com'});  
cqlsh>
```



2. List

When the order of elements matters, the list is used.

Here is the snapshot where column courses of list type id added in table "Teacher."

```
cqlsh> alter table University.Teacher add coursename list<text>;  
cqlsh>
```

Here is the snapshot where data is being inserted in column "coursename".

```
cqlsh> insert into University.Teacher(id,Name,Email,coursename) values(2,'Hamilton',  
{'hamilton@yahoo.com'},['Data Science']);  
cqlsh>
```

Here is the snapshot that shows the current database state after insertion.

```
id | coursename | email | name  
---+-----+-----+---  
2 | ['Data Science'] | {'hamilton@yahoo.com'} | Hamilton  
(1 rows)
```

3. Map

The map is a collection type that is used to store key value pairs. As its name implies that it maps one thing to another.

For example, if you want to save course name with its prerequisite course name, map collection can be used.

Here is the snapshot where map type is created for course name and its prerequisite course name.

```
cqlsh> Create table University.Course  
... (id int,  
... prereq map<text,text>,  
... primary key(id)  
... );  
cqlsh>
```

map collection
type

Here is the snapshot where data is being inserted in map collection type.

```
cqlsh> insert into University.Course(id,prereq) values(1,{'DataScience':'Database',  
'Neural Network':'Artificial Intelligence'});  
cqlsh>
```

course name mapped
to its prereq course

Practical 9

Aim: To create HDFS tables and loading them in hive and learn joining of tables in hive.

a)Load the Data in Table

Data can be loaded in 2 ways in Hive either from local file or from HDFS to Hive.

1)To load the data from local to Hive use the following command in NEW terminal:

```
hadoop fs -copyFromLocal /home/user/data/weather/2012.txt hdfs://hname:10001/hive/data/weather
```

Here the hdfs path was initially made in the create statement using LOCATION ‘ /hive/data/weather’

2)To load data is to load it from HDFS to hive using the following command:

```
LOAD DATA INPATH ‘hdfs:/data/2012.txt’ INTO TABLE weather;
```

b)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. It is more or less similar to SQL JOIN.

Syntax

join_table:

```
table_reference JOIN table_factor [join_condition]
```

```

| table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference
join_condition
| table_reference LEFT SEMI JOIN table_reference join_condition
| table_reference CROSS JOIN table_reference [join_condition]

```

Example

Consider the following table named CUSTOMERS..

```

+----+-----+----+-----+-----+
| ID | NAME   | AGE | ADDRESS | SALARY |
+----+-----+----+-----+-----+
| 1  | Ramesh | 32  | Ahmedabad | 2000.00 |
| 2  | Khilan | 25  | Delhi    | 1500.00 |
| 3  | kaushik | 23  | Kota     | 2000.00 |
| 4  | Chaitali | 25  | Mumbai   | 6500.00 |
| 5  | Hardik | 27  | Bhopal   | 8500.00 |
| 6  | Komal  | 22  | MP       | 4500.00 |
| 7  | Muffy  | 24  | Indore   | 10000.00 |
+----+-----+----+-----+-----+

```

Consider another table ORDERS as follows:

```

+----+-----+-----+-----+
| OID | DATE           | CUSTOMER_ID | AMOUNT |
+----+-----+-----+-----+
| 102 | 2009-10-08 00:00:00 | 3 | 3000 |
| 100 | 2009-10-08 00:00:00 | 3 | 1500 |
| 101 | 2009-11-20 00:00:00 | 2 | 1560 |
| 103 | 2008-05-20 00:00:00 | 4 | 2060 |
+----+-----+-----+-----+

```

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.

The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

```

hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
FROM CUSTOMERS c JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);

```

On successful execution of the query, you get to see the following response:

ID	NAME	AGE	AMOUNT
3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

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.

The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
LEFT OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

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.

The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c RIGHT
OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME | AMOUNT | DATE |
+-----+-----+-----+-----+
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
+-----+-----+-----+-----+
```

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.

The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
FULL OUTER JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME | AMOUNT | DATE |
+-----+-----+-----+-----+
| 1 | Ramesh | NULL | NULL |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
| 5 | Hardik | NULL | NULL |
| 6 | Komal | NULL | NULL |
| 7 | Muffy | NULL | NULL |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
```

Practical 10

Aim: Implement installation and configuration pig package.

Step 1: Download Apache Pig tar file.

```
hduser@VIE300211: ~  
hduser@VIE300211:~$ wget http://www-us.apache.org/dist/pig/pig-0.16.0/pig-0.16.0  
.tar.gz  
--2017-02-20 08:37:59-- http://www-us.apache.org/dist/pig/pig-0.16.0/pig-0.16.0  
.tar.gz  
Resolving www-us.apache.org (www-us.apache.org)... 140.211.11.105  
Connecting to www-us.apache.org (www-us.apache.org)[140.211.11.105]:80... connec  
ted.  
HTTP request sent, awaiting response... 200 OK  
Length: 177279333 (169M) [application/x-gzip]  
Saving to: 'pig-0.16.0.tar.gz'  
  
pig-0.16.0.tar.gz 56%[=====> ] 94.76M 247KB/s eta 7m 18s
```

Step 2: Copy file on Desktop and Extract the file (Right Click - Extract Here).

```
hduser@VIE300211: ~  
hduser@VIE300211:~$ wget http://www-us.apache.org/dist/pig/pig-0.16.0/pig-0.16.0  
.tar.gz  
--2017-02-20 08:37:59-- http://www-us.apache.org/dist/pig/pig-0.16.0/pig-0.16.0  
.tar.gz  
Resolving www-us.apache.org (www-us.apache.org)... 140.211.11.105  
Connecting to www-us.apache.org (www-us.apache.org)[140.211.11.105]:80... connec  
ted.  
HTTP request sent, awaiting response... 200 OK  
Length: 177279333 (169M) [application/x-gzip]  
Saving to: 'pig-0.16.0.tar.gz'  
  
pig-0.16.0.tar.gz 100%[=====] 169.07M 298KB/s in 13m 50s  
2017-02-20 08:51:50 (209 KB/s) - 'pig-0.16.0.tar.gz' saved [177279333/177279333]  
  
hduser@VIE300211:~$ tar -xzf pig-0.16.0.tar.gz  
hduser@VIE300211:~$
```

Step 3: Move Setup file to pig's home.

Command: `sudo mv /home/hduser/Desktop/pig-0.16.0 /usr/local/pig`

Step 4: Set environment variables into .bashrc file.

```

[;@]{$*alerts//'^(){}*}
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

#Hadoop variables
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-and64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export PATH=$JAVA_HOME/bin:$PATH
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
#Hive Variables
export HIVE_HOME=/usr/local/hive
export PATH=$PATH:/usr/local/hive/bin
#Pig Variables
export PIG_HOME=/usr/local/pig
export PATH=$PATH:/usr/local/pig/bin
export PIG_CLASSPATH=$HADOOP_INSTALL/etc

```

Command: `sudo gedit ~/.bashrc`

```

export PIG_HOME=/usr/local/pig
export PATH=$PATH:/usr/local/pig/bin
export PIG_CLASSPATH=$HADOOP_INSTALL/etc

```

Pig is installed and configured now.

Verifying the Installation:

Verify the installation of Apache Pig by typing the version command. If the installation is successful, you will get the version of Apache Pig as shown below.

`$ pig -version`

```

Apache Pig version 0.16.0 (r1682971)
compiled Feb 20 2017, 11:44:35

```

Apache Pig Execution Modes

You can run Apache Pig in two modes, namely, Local Mode and HDFS mode.

Local Mode : In this mode, all the files are installed and run from your local host and local file system. There is no need of Hadoop or HDFS. This mode is generally used for testing purpose.

MapReduce Mode : MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig Latin statements to process the data, a MapReduce job is invoked in the back-end to perform a particular operation on the data that exists in the HDFS.

Apache Pig Execution Mechanisms : Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

Interactive Mode (Grunt shell) – You can run Apache Pig in interactive mode using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).

Batch Mode (Script) – You can run Apache Pig in Batch mode by writing the Pig Latin script in a single file with .pig extension.

Embedded Mode (UDF) – Apache Pig provides the provision of defining our own functions (User Defined Functions) in programming languages such as Java, and using them in our script.

Invoking the Grunt Shell : You can invoke the Grunt shell in a desired mode (local/MapReduce) using the -x option as shown below.

Local mode

Command – \$./pig -x local

Output – Local Mode Output

MapReduce mode

Command – \$./pig -x mapreduce

Output – MapReduce Mode Output

Either of these commands gives you the Grunt shell prompt as shown below.

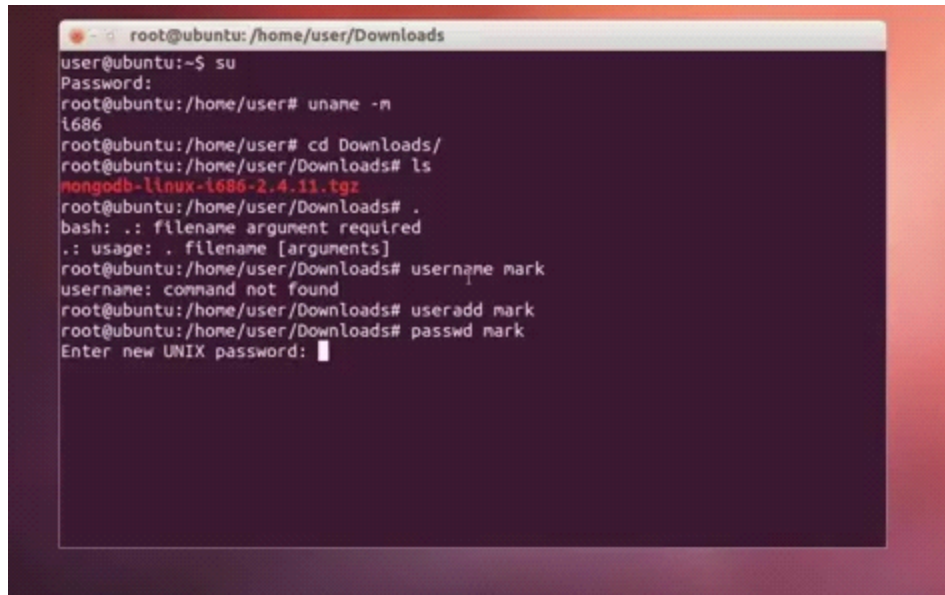
grunt>

Practical 11

Aim: Installation and configuration of MongoDB.

Method 1:

Step 1: Log in to a root user. For that there is a command and it will ask for the password and you can log in.

A terminal window titled 'root@ubuntu: /home/user/Downloads' with a dark purple background. The terminal shows a sequence of commands and their outputs. The user switches to root using 'su', then runs 'uname -m' to check the system architecture, which returns 'i686'. The user navigates to the 'Downloads' directory and lists its contents, showing a file named 'mongodb-linux-i686-2.4.11.tgz'. The user then attempts to run a command starting with a dot, which results in a 'bash: .: filename argument required' error. The user then tries 'username mark', which results in 'username: command not found'. Finally, the user runs 'useradd mark' and 'passwd mark', which prompts for a new UNIX password.

```
root@ubuntu: /home/user/Downloads
user@ubuntu:~$ su
Password:
root@ubuntu:/home/user# uname -m
i686
root@ubuntu:/home/user# cd Downloads/
root@ubuntu:/home/user/Downloads# ls
mongodb-linux-i686-2.4.11.tgz
root@ubuntu:/home/user/Downloads# .
bash: .: filename argument required
.: usage: . filename [arguments]
root@ubuntu:/home/user/Downloads# username mark
username: command not found
root@ubuntu:/home/user/Downloads# useradd mark
root@ubuntu:/home/user/Downloads# passwd mark
Enter new UNIX password: 
```

Step 2: Check the OS bit whether it is a 64-bit or 32-bit. Download a MongoDB for a Linux.

```
root@ubuntu: /opt/mongo
.: usage: . filename [arguments]
root@ubuntu:/home/user/Downloads# username mark
username: command not found
root@ubuntu:/home/user/Downloads# useradd mark
root@ubuntu:/home/user/Downloads# passwd mark
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@ubuntu:/home/user/Downloads# mkdir /opt/mongo
root@ubuntu:/home/user/Downloads# cp mongodb-linux-i686-2.4.11.tgz /opt/mongo
root@ubuntu:/home/user/Downloads# cd /opt/mongo
bash: cd: /opt/mongo: No such file or directory
root@ubuntu:/home/user/Downloads# cd /opt/mongo
root@ubuntu:/opt/mongo# ls
mongodb-linux-i686-2.4.11.tgz
root@ubuntu:/opt/mongo# tar -xvf mongodb-linux-i686-2.4.11.tgz
mongodb-linux-i686-2.4.11/README
mongodb-linux-i686-2.4.11/THIRD-PARTY-NOTICES
mongodb-linux-i686-2.4.11/GNU-AGPL-3.0
mongodb-linux-i686-2.4.11/bin/mongodump
mongodb-linux-i686-2.4.11/bin/mongorestore
mongodb-linux-i686-2.4.11/bin/mongoexport
mongodb-linux-i686-2.4.11/bin/mongoimport
```

Step 3: Extract files to /opt/mongo directory.

```
root@ubuntu: /opt/mongo/mongo/bin
total 234176
drwxr-xr-x 2 mark mark 4096 Aug 24 22:20 ./
drwxr-xr-x 3 root root 4096 Aug 24 22:20 ../
-rwxr-xr-x 1 mark mark 18017604 Aug 21 23:30 bsondump*
-rwxr-xr-x 1 mark mark 9285020 Aug 21 23:32 mongo*
-rwxr-xr-x 1 mark mark 18079484 Aug 21 23:31 mongod*
-rwxr-xr-x 1 mark mark 18076068 Aug 21 23:25 mongodump*
-rwxr-xr-x 1 mark mark 18031076 Aug 21 23:26 mongoexport*
-rwxr-xr-x 1 mark mark 18078288 Aug 21 23:29 mongofiles*
-rwxr-xr-x 1 mark mark 18042372 Aug 21 23:27 mongoimport*
-rwxr-xr-x 1 mark mark 18022052 Aug 21 23:29 mongooplog*
-rwxr-xr-x 1 mark mark 18024964 Aug 21 23:30 mongoperf*
-rwxr-xr-x 1 mark mark 18082500 Aug 21 23:26 mongorestore*
-rwxr-xr-x 1 mark mark 13621796 Aug 21 23:32 mongos*
-rwxr-xr-x 1 mark mark 17984852 Aug 21 23:31 mongosniff*
-rwxr-xr-x 1 mark mark 18067108 Aug 21 23:27 mongostat*
-rwxr-xr-x 1 mark mark 18022436 Aug 21 23:28 mongotop*
root@ubuntu:/opt/mongo/mongo/bin# mkdir /var/lib/mongo
root@ubuntu:/opt/mongo/mongo/bin# chown -R mongo:mongo /var/lib/mongo/
chown: invalid user: 'mongo:mongo'
root@ubuntu:/opt/mongo/mongo/bin# chown -R mark:mark /var/lib/mongo/
root@ubuntu:/opt/mongo/mongo/bin# mkdir /var/log/mongo
root@ubuntu:/opt/mongo/mongo/bin# chown -R mark:mark /var/lib/mongo/
root@ubuntu:/opt/mongo/mongo/bin#
```

Step 4: Open a new terminal to start the service.

```

root@ubuntu: /opt/mongo/mongo/bin
Sun Aug 24 22:33:25.777 [initandlisten] git version: fa13d1ee8da0f112f588570b407
0f
root@ubuntu: /opt/mongo/mongo/bin
6_user@ubuntu:~$ su
SuPassword:
Suroot@ubuntu:/home/user# cd /opt/mongo/mongo/bin/
Suroot@ubuntu:/opt/mongo/mongo/bin# ./mongo
ocMongoDB shell version: 2.4.11
Suconnecting to: test
SuWelcome to the MongoDB shell.
lgFor interactive help, type "help".
SuFor more comprehensive documentation, see
oc http://docs.mongodb.org/
SuQuestions? Try the support group
lg http://groups.google.com/group/mongodb-user
SuServer has startup warnings:
tgSun Aug 24 22:33:25.776 [initandlisten]
1gSun Aug 24 22:33:25.776 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary
Su.
SuSun Aug 24 22:33:25.776 [initandlisten] ** 32 bit builds are limited to less
rtss than 2GB of data (or less with --journal).
SuSun Aug 24 22:33:25.776 [initandlisten] ** Note that journaling defaults to
#o off for 32 bit and is currently off.
[ ] Sun Aug 24 22:33:25.776 [initandlisten] ** See http://dochub.mongodb.org/c
ore/32bit
Sun Aug 24 22:33:25.776 [initandlisten]
>

```

Method 2:

sudo apt-get install mongodb

```

hduser@VIE300211: ~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.4.2/bin
hduser@VIE300211:~$ apt-get install mongodb
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
hduser@VIE300211:~$ sudo apt-get install mongodb
[sudo] password for hduser:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libboost-program-options1.58.0 libboost-thread1.58.0 libgoogle-perftools4
libpcrecpp0v5 libsnappy1v5 libtcmalloc-minimal4 libv8-3.14.5
libyaml-cpp0.5v5 mongodb-clients mongodb-server
The following NEW packages will be installed:
libboost-program-options1.58.0 libboost-thread1.58.0 libgoogle-perftools4
libpcrecpp0v5 libsnappy1v5 libtcmalloc-minimal4 libv8-3.14.5
libyaml-cpp0.5v5 mongodb mongodb-clients mongodb-server
0 upgraded, 11 newly installed, 0 to remove and 349 not upgraded.
Need to get 57.9 MB of archives.
After this operation, 196 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libpcrecpp0v5 amd64 2:8.38-3.1 [15.2 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libboost-program-options1.58.0 amd64 1.58.0+dfsg-Subuntu3.1 [138 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libboost-thread1.58.0 amd64 1.58.0+dfsg-Subuntu3.1 [47.0 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libtcmalloc-minimal4 amd64 2.4-0ubuntu5 [105 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libgoogle-perftools4 amd64 2.4-0ubuntu5 [187 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 libv8-3.14.5 amd64 3.14.5-8-Subuntu2 [1,189 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 libyaml-cpp0.5v5 amd64 0.5.2-3 [158 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libsnappy1v5 amd64 1.1.3-2 [16.0 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 mongodb-clients amd64 1:2.6.10-0ubuntu1 [48.6 MB]
Get:10 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 mongodb-server amd64 1:2.6.10-0ubuntu1 [7,425 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 mongodb amd64 1:2.6.10-0ubuntu1 [5,112 B]
Fetched 57.9 MB in 2min 10s (442 kB/s)
Selecting previously unselected package libpcrecpp0v5:amd64.
(Reading database ... 213076 files and directories currently installed.)
Preparing to unpack .../libpcrecpp0v5_2%3a8.38-3.1_amd64.deb ...
Unpacking libpcrecpp0v5:amd64 (2:8.38-3.1) ...
Selecting previously unselected package libboost-program-options1.58.0:amd64.
Preparing to unpack .../libboost-program-options1.58.0_1.58.0+dfsg-Subuntu3.1_amd64.deb ...
Unpacking libboost-program-options1.58.0:amd64 (1.58.0+dfsg-Subuntu3.1) ...
Selecting previously unselected package libboost-thread1.58.0:amd64.
Preparing to unpack .../libboost-thread1.58.0_1.58.0+dfsg-Subuntu3.1_amd64.deb ...
Unpacking libboost-thread1.58.0:amd64 (1.58.0+dfsg-Subuntu3.1) ...
Selecting previously unselected package libtcmalloc-minimal4.

```

Starting the mongod shell

```
hduser@VIE300211: ~/Desktop/mongodb-linux-x86_64-ubuntu1604-3.4.2/bin
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for systemd (229-4ubuntu10) ...
Processing triggers for ureadahead (0.100.0-19) ...
hduser@VIE300211:~$ ./mongo
bash: ./mongo: No such file or directory
hduser@VIE300211:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
Questions? Try the support group
http://groups.google.com/group/mongodb-user

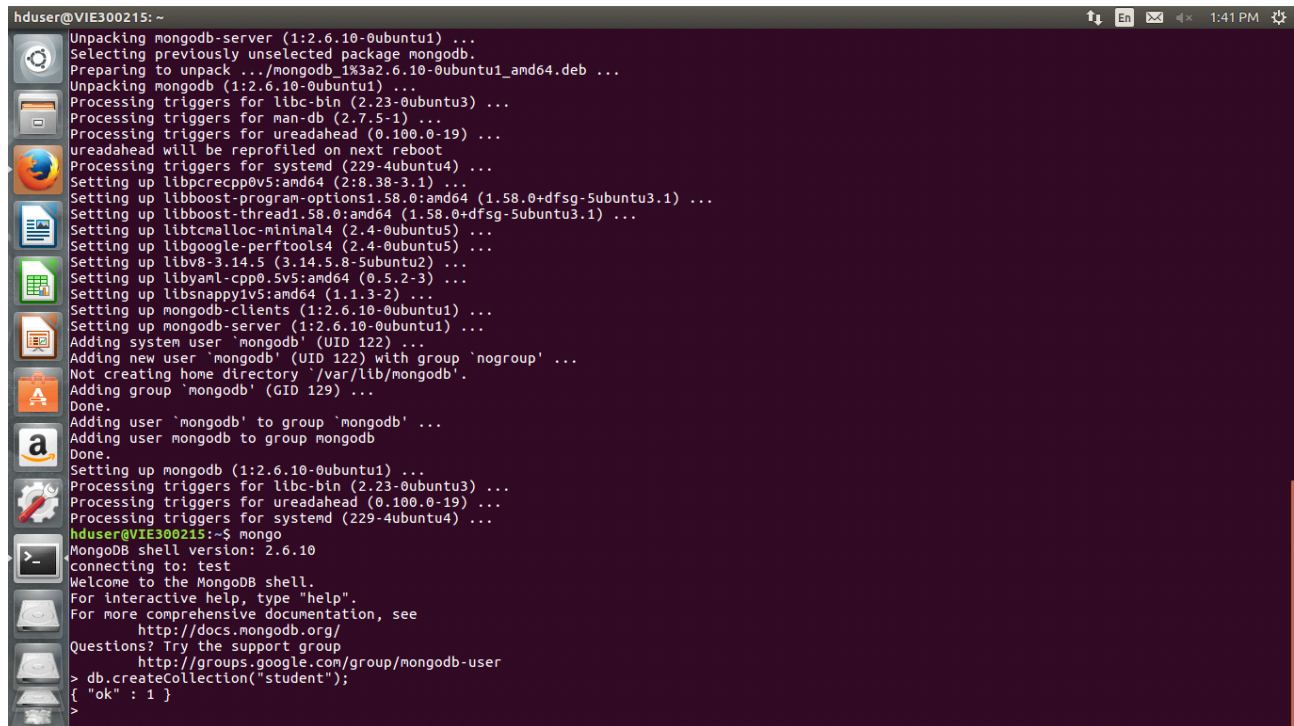
> use myDB
switched to db myDB
> db;
myDB
> show collections
db.createCollection("Students");
{ "ok" : 1 }
> db.Students.insert({_id:1,Studname:"MALAV",Grade:"V",Hobbies:"Singing"})
WriteResult({ "nInserted" : 1 })
> db.Students.insert({_id:1,Studname:"MALAV",Grade:"V",Hobbies:"Singing"})
WriteResult({
  "nInserted" : 0,
  "writeError" : {
    "code" : 11000,
    "errmsg" : "insertDocument :: caused by :: 11000 E11000 duplicate key error index: myDB.Students.$_id_ dup key: { : 1.0 }"
  }
})
> db.Students.find();
{ "_id" : 1, "Studname" : "MALAV", "Grade" : "V", "Hobbies" : "Singing" }
> db.Students.find().pretty();
{ "_id" : 1, "Studname" : "MALAV", "Grade" : "V", "Hobbies" : "Singing" }
> db.Students.update({_id:1,Studname:"MALAV",Grade:"V",Hobbies:"Singing"},{$set:{Hobbies:"gaming"}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Students.find().pretty();
{ "_id" : 1, "Studname" : "MALAV", "Grade" : "V", "Hobbies" : "gaming" }
> db;
myDB
> exit
bye
```

Practical 12

Aim : Using Mongodb create a collection named “student”, Insert fields _id, Name, Grade and Hobbies . Implement updating of one field and retrieve various documents from collection student.

Step 1 : Create collection

```
> db.createCollection("student");
```

A terminal window showing the installation of MongoDB on Ubuntu. The output of the 'dpkg --get-selections mongodb' command is visible, showing that the package is installed. The terminal then shows the user entering the MongoDB shell and running the 'db.createCollection("student");' command, which returns '{ "ok" : 1 }'.

```
hduser@VIE300215:~$ dpkg --get-selections mongodb
Unpacking mongodb-server (1:2.6.10-0ubuntu1) ...
Selecting previously unselected package mongodb.
Preparing to unpack .../mongodb_1%3a2.6.10-0ubuntu1_amd64.deb ...
Unpacking mongodb (1:2.6.10-0ubuntu1) ...
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for ureadahead (0.100.0-19) ...
ureadahead will be reprofiled on next reboot
Processing triggers for systemd (229-4ubuntu4) ...
Setting up libpcrcpp0v5:amd64 (2:8.38-3.1) ...
Setting up libboost-program-options1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...
Setting up libboost-thread1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...
Setting up libtcmalloc-minimal4 (2.4-0ubuntu5) ...
Setting up libgoogle-perftools4 (2.4-0ubuntu5) ...
Setting up libv8-3.14.5 (3.14.5.8-5ubuntu2) ...
Setting up libyaml-cpp0.5v5:amd64 (0.5.2-3) ...
Setting up libsnappy1v5:amd64 (1.1.3-2) ...
Setting up mongodb-clients (1:2.6.10-0ubuntu1) ...
Setting up mongodb-server (1:2.6.10-0ubuntu1) ...
Adding system user 'mongodb' (UID 122) ...
Adding new user 'mongodb' (UID 122) with group 'nogroup' ...
Not creating home directory '/var/lib/mongodb'.
Adding group 'mongodb' (GID 129) ...
Done.
Adding user 'mongodb' to group 'mongodb' ...
Adding user mongodb to group mongodb
Done.
Setting up mongodb (1:2.6.10-0ubuntu1) ...
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (229-4ubuntu4) ...
hduser@VIE300215:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
> db.createCollection("student");
{ "ok" : 1 }
>
```

Step 2 : Insert fields

```
> db.student.insert({_id:1,stdname:"xyz",Grade:"VII",Hobbies:"Dancing"});
> db.student.insert({_id:2,stdname:"abc",Grade:"VII",Hobbies:"Music"});
> db.student.insert({_id:3,stdname:"pqr",Grade:"VI",Hobbies:"Swimming"});
```

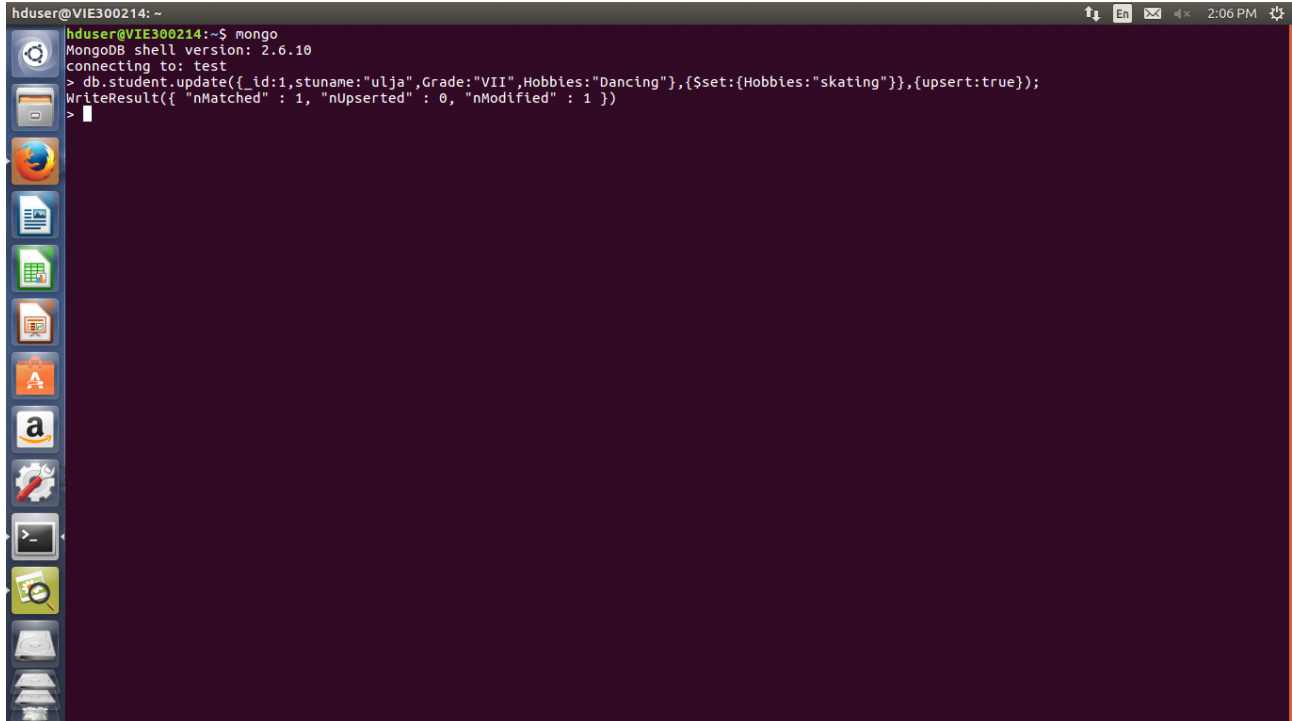


```
hduser@VIE300214: ~  
Processing triggers for libc-bin (2.23-0ubuntu3) ...  
Processing triggers for man-db (2.7.5-1) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
ureadahead will be reprofiled on next reboot  
Processing triggers for systemd (229-4ubuntu4) ...  
Setting up libpcrecpp0v5:amd64 (2:8.38-3.1) ...  
Setting up libboost-program-options1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...  
Setting up libboost-thread1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...  
Setting up libtcmalloc-minimal4 (2.4-0ubuntu5) ...  
Setting up libgoogle-perftools4 (2.4-0ubuntu5) ...  
Setting up libv8-3.14.5 (3.14.5.8-5ubuntu2) ...  
Setting up libyaml-cpp0.5v5:amd64 (0.5.2-3) ...  
Setting up libsnappy1v5:amd64 (1.1.3-2) ...  
Setting up mongodb-clients (1:2.6.10-0ubuntu1) ...  
Setting up mongodb-server (1:2.6.10-0ubuntu1) ...  
Adding system user 'mongodb' (UID 122) ...  
Adding new user 'mongodb' (UID 122) with group 'nogroup' ...  
Not creating home directory '/var/lib/mongodb'.  
Adding group 'mongodb' (GID 129) ...  
Done.  
Adding user 'mongodb' to group 'mongodb' ...  
Adding user mongodb to group mongodb  
Done.  
Setting up mongodb (1:2.6.10-0ubuntu1) ...  
Processing triggers for libc-bin (2.23-0ubuntu3) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
Processing triggers for systemd (229-4ubuntu4) ...  
hduser@VIE300214:~$ mongo  
MongoDB shell version: 2.6.10  
connecting to: test  
Welcome to the MongoDB shell.  
For interactive help, type "help".  
For more comprehensive documentation, see  
http://docs.mongodb.org/  
Questions? Try the support group  
http://groups.google.com/group/mongodb-user  
> db.createCollection("student");  
{ "ok" : 1 }  
> db.student.insert({_id:1,stuname:"  
2017-03-15T13:41:44.671+0530 SyntaxError: Unexpected token ILLEGAL  
> db.student.insert({_id:1,stuname:"ulja",Grade:"VII",Hobbies:"Dancing"});  
WriteResult({ "nInserted" : 1 })  
>
```

```
hduser@VIE300214: ~  
Processing triggers for systemd (229-4ubuntu4) ...  
Setting up libpcrecpp0v5:amd64 (2:8.38-3.1) ...  
Setting up libboost-program-options1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...  
Setting up libboost-thread1.58.0:amd64 (1.58.0+dfsg-5ubuntu3.1) ...  
Setting up libtcmalloc-minimal4 (2.4-0ubuntu5) ...  
Setting up libgoogle-perftools4 (2.4-0ubuntu5) ...  
Setting up libv8-3.14.5 (3.14.5.8-5ubuntu2) ...  
Setting up libyaml-cpp0.5v5:amd64 (0.5.2-3) ...  
Setting up libsnappy1v5:amd64 (1.1.3-2) ...  
Setting up mongodb-clients (1:2.6.10-0ubuntu1) ...  
Setting up mongodb-server (1:2.6.10-0ubuntu1) ...  
Adding system user 'mongodb' (UID 122) ...  
Adding new user 'mongodb' (UID 122) with group 'nogroup' ...  
Not creating home directory '/var/lib/mongodb'.  
Adding group 'mongodb' (GID 129) ...  
Done.  
Adding user 'mongodb' to group 'mongodb' ...  
Adding user mongodb to group mongodb  
Done.  
Setting up mongodb (1:2.6.10-0ubuntu1) ...  
Processing triggers for libc-bin (2.23-0ubuntu3) ...  
Processing triggers for ureadahead (0.100.0-19) ...  
Processing triggers for systemd (229-4ubuntu4) ...  
hduser@VIE300214:~$ mongo  
MongoDB shell version: 2.6.10  
connecting to: test  
Welcome to the MongoDB shell.  
For interactive help, type "help".  
For more comprehensive documentation, see  
http://docs.mongodb.org/  
Questions? Try the support group  
http://groups.google.com/group/mongodb-user  
> db.createCollection("student");  
{ "ok" : 1 }  
> db.student.insert({_id:1,stuname:"  
2017-03-15T13:41:44.671+0530 SyntaxError: Unexpected token ILLEGAL  
> db.student.insert({_id:1,stuname:"ulja",Grade:"VII",Hobbies:"Dancing"});  
WriteResult({ "nInserted" : 1 })  
> db.student.insert({_id:2,stuname:"Tithi",Grade:"VII",Hobbies:"Music"});  
WriteResult({ "nInserted" : 1 })  
> db.student.insert({_id:3,stuname:"Khushi",Grade:"VI",Hobbies:"Swimming"});  
WriteResult({ "nInserted" : 1 })  
>
```

Step 3 : update one field

```
>  
db.student.update({_id:1,stdname:"xyz",Grade:"VII",Hobbies:"Dancing"},{$set:{Hobbies:"skating"}},{  
upsert:true});
```

A screenshot of a terminal window titled 'hduser@VIE300214: ~'. The terminal shows the execution of the 'mongo' command, which opens the MongoDB shell. The shell version is 2.6.10. The user connects to a 'test' database. The command executed is 'db.student.update({_id:1,stdname:"ulja",Grade:"VII",Hobbies:"Dancing"},{\$set:{Hobbies:"skating"}},{upsert:true});'. The output is 'WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })'. The terminal window has a dark background and a vertical toolbar on the left side with various application icons.

```
hduser@VIE300214: ~  
hduser@VIE300214:~$ mongo  
MongoDB shell version: 2.6.10  
connecting to: test  
> db.student.update({_id:1,stdname:"ulja",Grade:"VII",Hobbies:"Dancing"},{$set:{Hobbies:"skating"}},{upsert:true});  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
>
```

Practical 13

AIM:- Implement CRUD operations.**C(Create/Insert):-**

1. Create DataBase:
syntax: use database_name;
Ex: use mydb;
2. create Collection:
syntax: db.createCollection("Collection_name");
Ex: db.createCollection("std");
3. Insert Document:
syntax: db.collection_name.insert({_Field_name:Value, Field_name:Value});
Ex: db.std.insert({_id:32,name:"jeet",clg:"VIT"});
4. Save Document:
syntax: db.collection_name.save({record});
Ex: db.std.save({_id:32,name:"jeet",clg:"VIT"});

R(Read):-

List of all databases:

Syntax: show dbs;

Ex: show dbs;

Display all collections in Particular database:

syntax: show collection;

Ex: show collection;

Display document of Collection:

syntax: db.collection_name.find();

Ex: db.std.find();

Display particular document:

syntax: db.collection_name.find({}, {field_name});

Ex: db.std.find({}, {"id:32"});

Display last updated Document:

syntax: db.collection_name.find().pretty();

Ex: db.std.find().pretty();

Display document with condition:

conditions:

eq: for equal

ne: for not equal

gt: for greater than

lt: for less than

gte: for greater or equal

lte: for less than or equal

or: for or condition

syntax: db.collection_name.find({field_name:{\$condition:"value"}}).pretty();

Ex: db.std.find({clg:{\$eq:"clg"}}).pretty();

U(Update):-

syntax: db.Collection_name.update(field_name,{\$set:{change field_name:"value"}},
{upsert:true};

Ex: db.std.update({_id:32},{\$set:{clg:"VIER"}},{upsert:true};

D(Delete):-

drop DataBase:

syntax: db.dropDatabase(database_name);

Ex: db.dropDatabase(mydb);

Delete Collection:

syntax: db.Collection_name.drop();

Ex: db.std.drop();

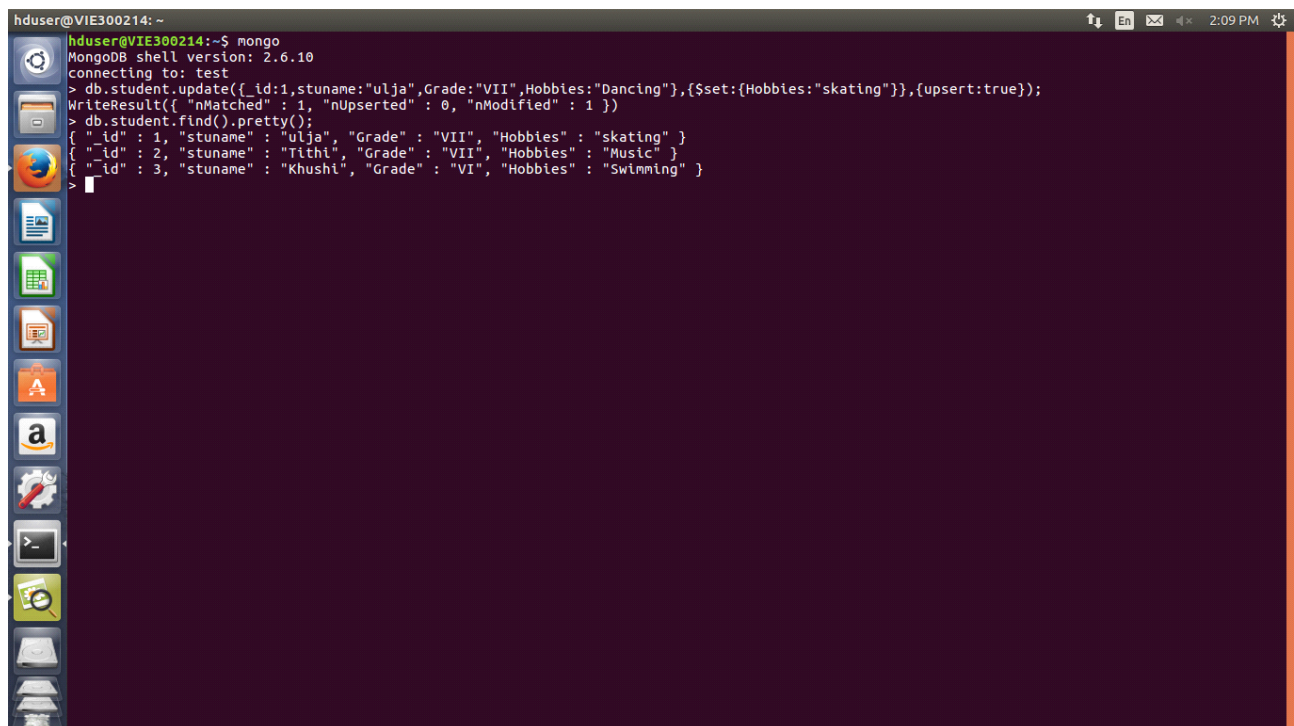
Delete Document:

syntax: db.Collection_name.remove({field_name});

Ex: db.std.remove({_id:32});

Step 4 : Retrive documents from collection

```
> db.student.find().pretty();
```



The screenshot shows a terminal window with the following content:

```
hduser@VIE300214: ~  
hduser@VIE300214:~$ mongo  
MongoDB shell version: 2.6.10  
connecting to: test  
> db.student.update({_id:1,stuname:"ulja",Grade:"VII",Hobbies:"Dancing"},{$set:{Hobbies:"skating"}},{upsert:true});  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
> db.student.find().pretty();  
{ "_id" : 1, "stuname" : "ulja", "Grade" : "VII", "Hobbies" : "skating" }  
{ "_id" : 2, "stuname" : "Tithi", "Grade" : "VII", "Hobbies" : "Music" }  
{ "_id" : 3, "stuname" : "Khushi", "Grade" : "VI", "Hobbies" : "Swimming" }
```

Practical 14

Aim : To implement mapreduce using mongodb.

Basic operations of mongodb like CRUD operations can be implemented using mapreduce.

Return the Total Price Per Customer

Perform the map-reduce operation on the orders collection to group by the cust_id, and calculate the sum of the price for each cust_id:

Define the map function to process each input document:

In the function, this refers to the document that the map-reduce operation is processing.

The function maps the price to the cust_id for each document and emits the cust_id and price pair.

```
var mapFunction1 = function() {
  emit(this.cust_id, this.price);
};
```

Define the corresponding reduce function with two arguments keyCustId and valuesPrices:

The valuesPrices is an array whose elements are the price values emitted by the map function and grouped by keyCustId.

The function reduces the valuesPrice array to the sum of its elements.

```
var reduceFunction1 = function(keyCustId, valuesPrices) {
  return Array.sum(valuesPrices);
};
```

Perform the map-reduce on all documents in the orders collection using the mapFunction1 map function and the reduceFunction1 reduce function.

```
db.orders.mapReduce(
  mapFunction1,
  reduceFunction1,
  { out: "map_reduce_example" }
)
```

This operation outputs the results to a collection named map_reduce_example. If the map_reduce_example collection already exists, the operation will replace the contents with the results of this map-reduce operation:

Calculate Order and Total Quantity with Average Quantity Per Item

The operation groups by the item.sku field, and calculates the number of orders and the total quantity ordered for each sku. The operation concludes by calculating the average quantity per order for each sku value:

VIER Page No.:

Date: Roll No.:

Define the map function to process each input document:

In the function, this refers to the document that the map-reduce operation is processing.

For each item, the function associates the sku with a new object value that contains the count of 1 and the item qty for the order and emits the sku and value pair.

```
var mapFunction2 = function() {
  for (var idx = 0; idx < this.items.length; idx++) {
    var key = this.items[idx].sku;
    var value = {
      count: 1,
      qty: this.items[idx].qty
    };
    emit(key, value);
  }
};
```

Define the corresponding reduce function with two arguments keySKU and countObjVals:

countObjVals is an array whose elements are the objects mapped to the grouped keySKU values passed by map function to the reducer function.

The function reduces the countObjVals array to a single object reducedValue that contains the count and the qty fields.

In reducedVal, the count field contains the sum of the count fields from the individual array elements, and the qty field contains the sum of the qty fields from the individual array elements.

```
var reduceFunction2 = function(keySKU, countObjVals) {  
  reducedVal = { count: 0, qty: 0 };  
  for (var idx = 0; idx < countObjVals.length; idx++) {  
    reducedVal.count += countObjVals[idx].count;  
    reducedVal.qty += countObjVals[idx].qty;  
  }  
  return reducedVal;  
};
```

Define a finalize function with two arguments key and reducedVal. The function modifies the reducedVal object to add a computed field named avg and returns the modified object:

```
var finalizeFunction2 = function (key, reducedVal) {  
  reducedVal.avg = reducedVal.qty/reducedVal.count;  
  return reducedVal;  
};
```

Perform the map-reduce operation on the orders collection using the mapFunction2, reduceFunction2, and finalizeFunction2 functions.

```
db.orders.mapReduce( mapFunction2,reduceFunction2,  
{  
  out: { merge: "map_reduce_example" },  
  query: { ord_date:  
    { $gt: new Date('01/03/2017') }  
  },  
  finalize: finalizeFunction2  
})
```

This operation uses the query field to select only those documents with ord_date greater than new. Then it output the results to a collection map_reduce_example. If the map_reduce_example collection already exists, the operation will merge the existing contents with the results of this map-reduce operation.