Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss Parth Sandeep Dabholkar of Computer Department, Semester VII with Roll No. 2103032 has completed a course of the necessary experiments in the subject Big Data Analytics under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2024 - 2025

15/10/24

Teacher In- Charge

Head of the Department

Date 15/10/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

EXPERIMENT 1

INSTALLATION GUIDELINE FOR HADOOP

THEORY

What is Hadoop?

Apache Hadoop is an open-source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key-value pairs. The output of the map task is consumed by reducing tasks to aggregate output and provide the desired result.
- Hadoop Common – Provides common Java libraries that can be used across all modules.

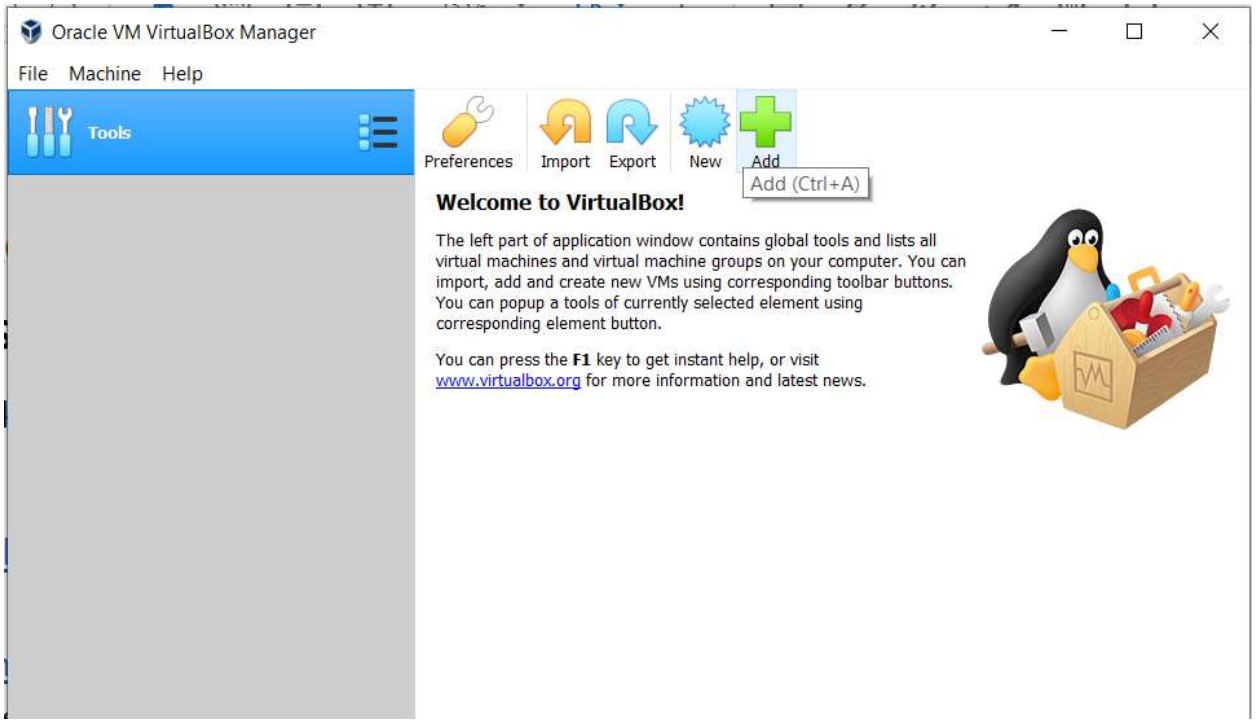
STEP 1

Use the following links to download above mentioned software successfully:
1. Link for Cloudera:
https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.12.0-0-virtualbox.zip

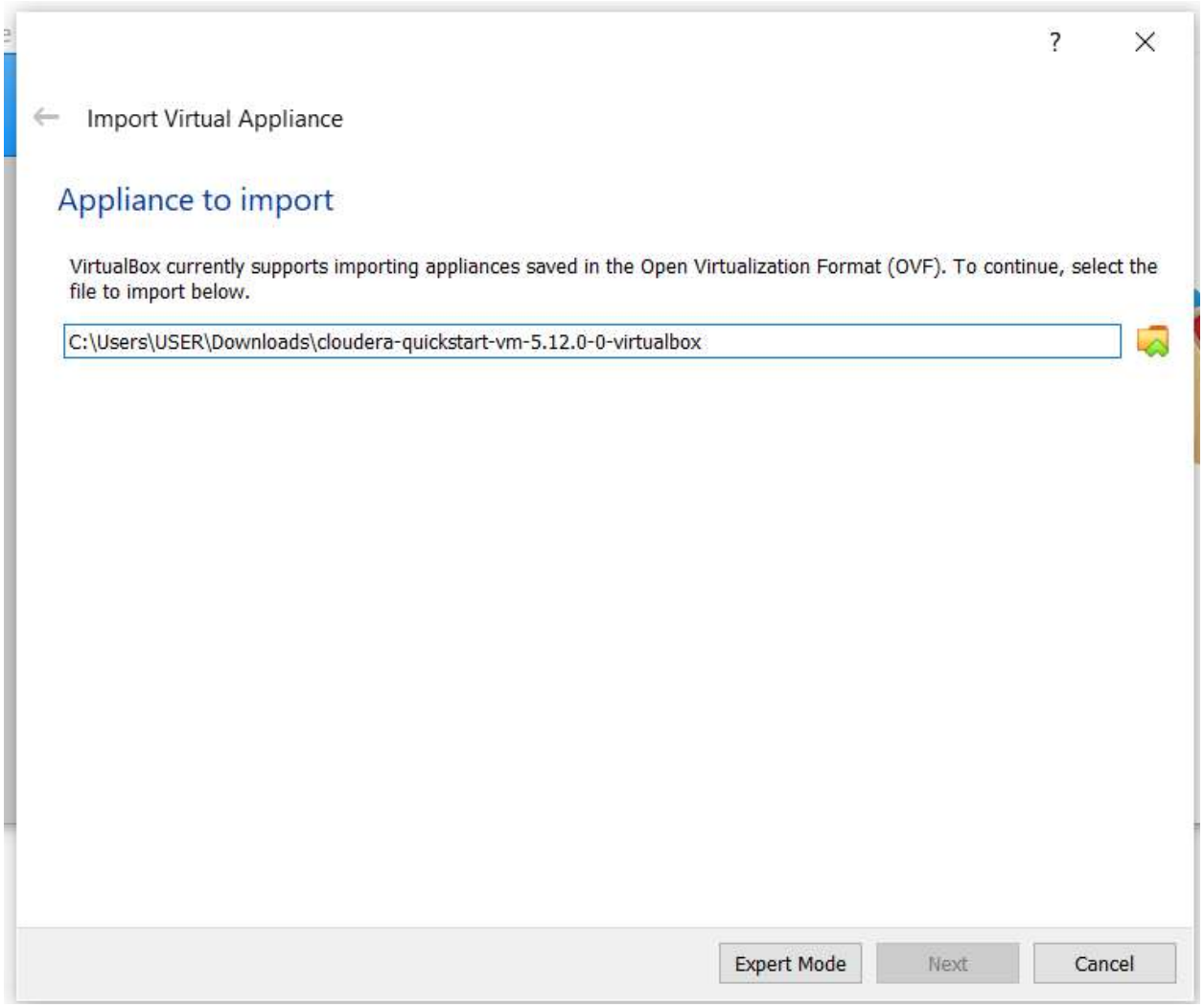
2. Link for VirtualBox: https://www.virtualbox.org/wiki/Download_Old_Builds_6_0 3
In case of any error, check the following link to enable Virtualization on your device (please look for the company whose machine(laptop/PC) you are using):
https://2nwiki.2n.cz/pages/viewpage.action?pageId=75202968

STEP 2

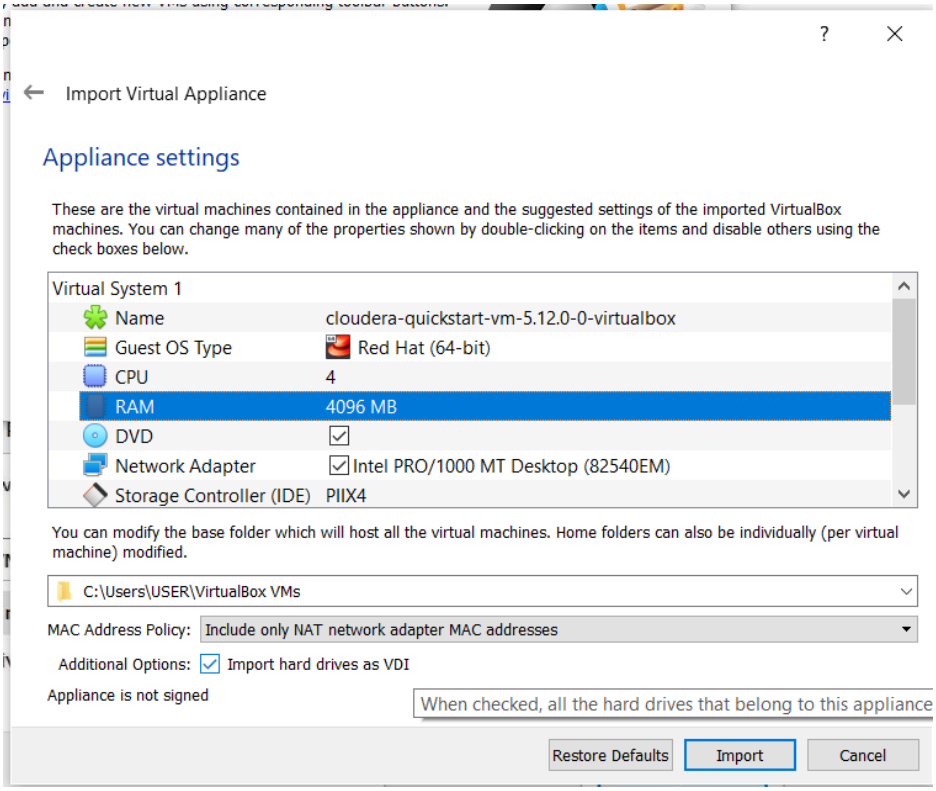
After downloading Cloudera, unzip it using a zip extractor and extract the files. Upon completion, open the virtual box software and select the Import option.

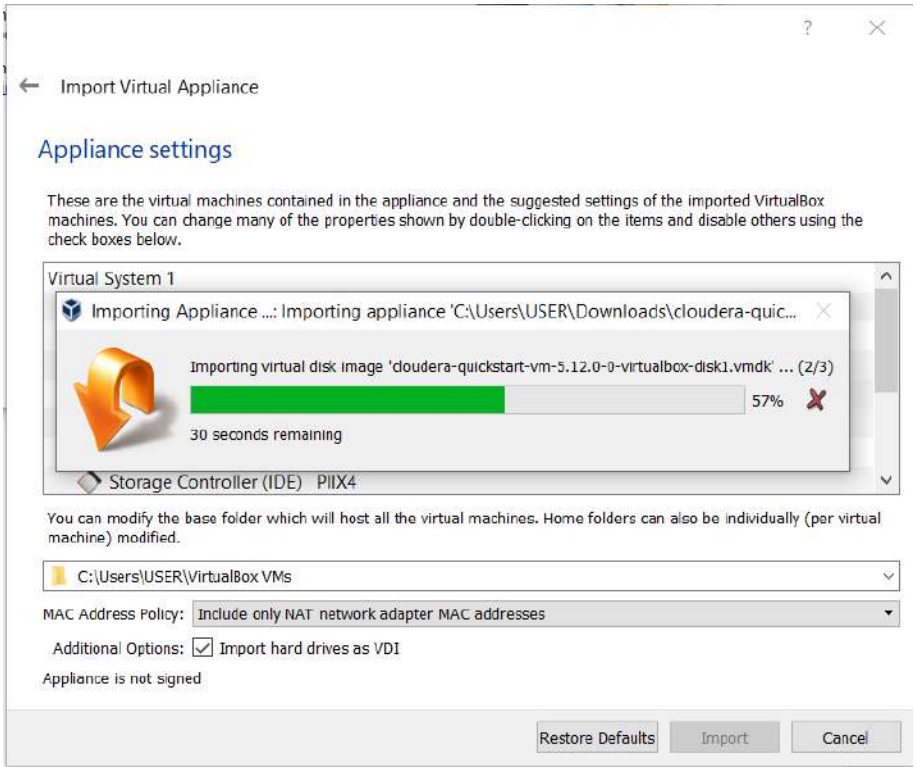


After selecting import, include the path of the previously downloaded Cloudera software.

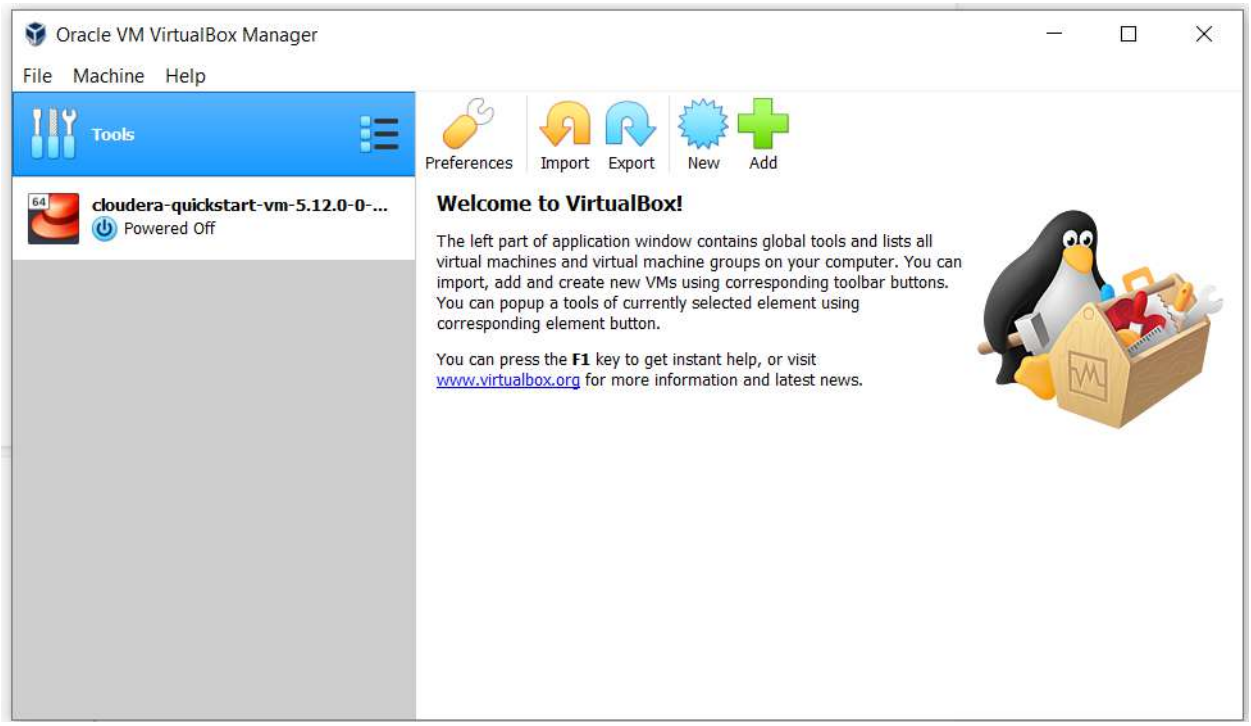


STEP 3 In the appliance settings, change the CPU section value from ‘1’ to ‘4’.

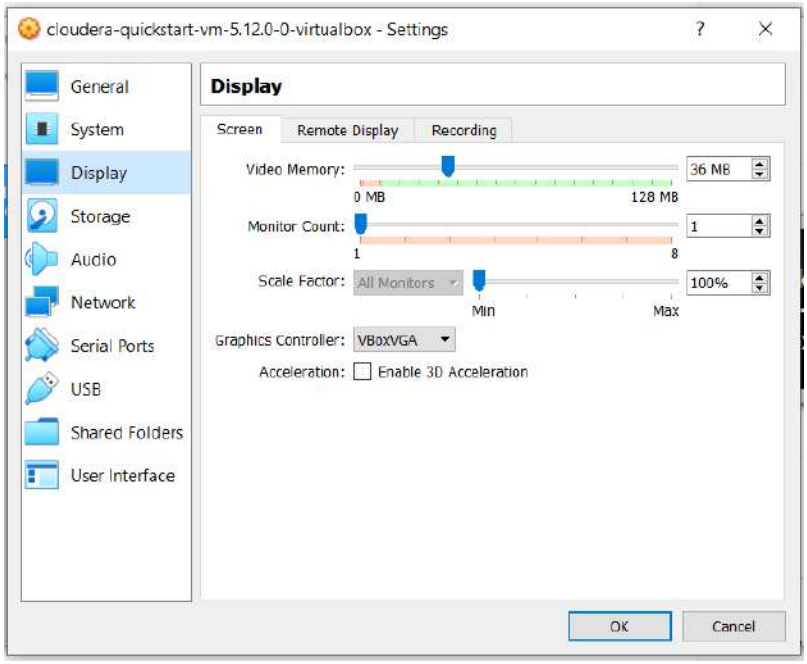




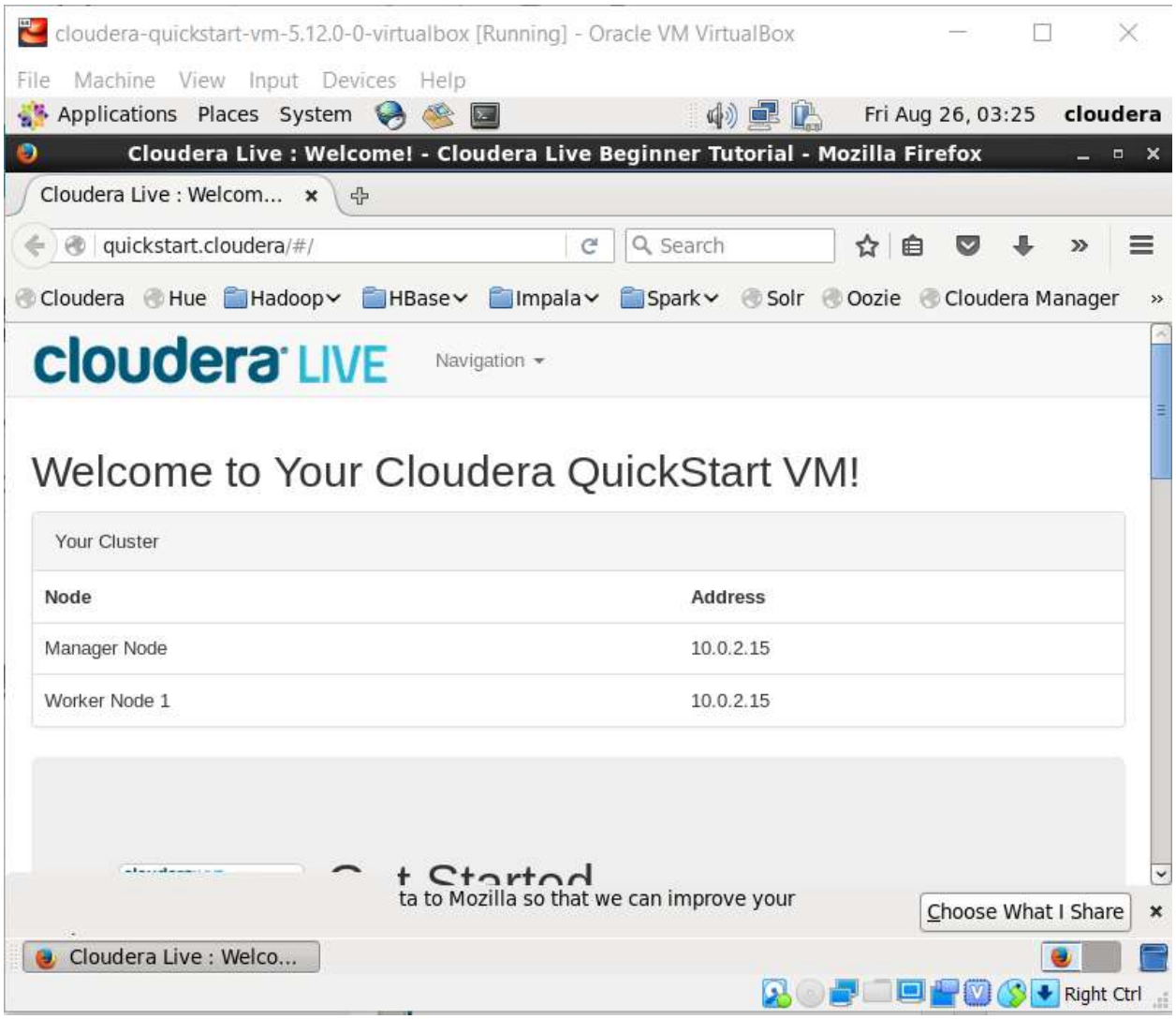
STEP 4 Proceed further if your VirtualBox homepage looks like this



Now click on the cloudera-quickstart-vm file which was initially showing powered off. Once you click on it, change the display settings and keep the video memory value between 0-40MB.



STEP 5 Now click on the Start button and wait for a few minutes. Initially, your window will look like this.



Once loading is completed, this window will appear.

Experiment 02

Aim: Use of Sqoop tool to transfer data between Hadoop and relational database servers.

Theory:

Introduction to Sqoop

Sqoop, short for SQL-to-Hadoop, is an open-source tool specifically designed to facilitate the efficient transfer of bulk data between Hadoop and structured data sources such as relational databases. The primary purpose of Sqoop is to allow seamless data import from traditional databases like MySQL, Oracle, or PostgreSQL into the Hadoop Distributed File System (HDFS), as well as data export back to these databases. This functionality is crucial in big data environments, where the integration of traditional and modern data storage systems is essential for comprehensive data processing and analysis.

Key Components and Workflow

Sqoop operates using database-specific connectors, which ensure compatibility and optimize the data transfer process. The tool offers two main functionalities: the Import Tool and the Export Tool. The Import Tool is used to bring data from relational databases into Hadoop, while the Export Tool enables data transfer from Hadoop back to relational databases. Additionally, Sqoop jobs, which are saved commands, can be executed repeatedly for consistent data transfers. The workflow typically starts with importing data by defining a database connection using JDBC, selecting the relevant table or query, and then transferring the data into HDFS or Hive. Sqoop supports parallel processing, which significantly speeds up data transfer by splitting tasks into multiple parallel operations. Conversely, during data export, Sqoop prepares the data in Hadoop, selects the target table in the relational database, and executes the transfer back to the database.

Sqoop Commands and Options

Common Sqoop commands include the import command, which is used to bring data into Hadoop, and the export command, which sends data back to the relational database. For example, a typical import command might look like: sqoop import --connect jdbc:mysql://hostname/dbname --username user --password pass --table tablename --target-dir /user/hadoop/tablename. Similarly, the export command would resemble: sqoop export --connect jdbc:mysql://hostname/dbname --username user

--password pass --table tablename --export-dir /user/hadoop/tablename. Sqoop offers a range of options to customize these commands, including general options like --connect, --username, and --password, as well as specific options for import (--target-dir, --split-by) and export (--export-dir, --input-fields-terminated-by) operations.

Import Modes and Data Transformation

Sqoop supports different import modes, including full table import, where entire tables are transferred, and incremental import, which only imports new or updated rows. The incremental import mode can be further fine-tuned with options like --incremental append or --incremental lastmodified, and checkpointing can be used to maintain data integrity during these operations. Additionally, Sqoop can integrate with MapReduce jobs to allow data transformation during import/export, and supports compression techniques such as Gzip or Snappy to optimize storage and transfer.

Error Handling, Optimization, and Applications

To ensure data accuracy, Sqoop provides error handling options like --validate, which checks for data integrity after transfer. Performance can be optimized by adjusting the number of mappers (--num-mappers) for parallel execution, and by monitoring and tuning based on network and database throughput. Real-world applications of Sqoop include data migration from legacy systems to Hadoop, integration with ETL processes for streamlined data ingestion, and populating data warehouses with processed data from Hadoop.

In a practical context, the number of mappers used during parallel import can be calculated using the formula: Number of Mappers=Total RowsRows per Mapper\text{Number of Mappers} = \frac{\text{Total Rows}}{\text{Rows per Mapper}}Number of Mappers=Rows per MapperTotal Rows Additionally, data throughput, an important performance metric, is calculated as: Throughput=Total Data SizeTotal Transfer Time\text{Throughput} = \frac{\text{Total Data Size}}{\text{Total Transfer Time}}Throughput=Total Transfer TimeTotal Data Size

Several challenges can arise during the use of Sqoop, including network bandwidth limitations, database locking issues during heavy imports, and the need for secure authentication. Best practices include ensuring sufficient network capacity, optimizing query execution to avoid database locks, and using Sqoop’s authentication methods to protect database credentials.

Sqoop Import :

Sqoop import command helps in implementation of the operation. With the help of the import command, we can import a table from the Relational database management system to the Hadoop database server. Records in Hadoop structure are stored in text files and each record is imported as a separate record in Hadoop database server. We can also create load and partition in Hive while importing data..Sqoop also supports incremental import of data which means in case we have imported a database and we want to add some more rows, so with the help of these functions we can only add the new rows to existing database, not the complete database.

Sqoop Export :

Sqoop export command helps in the implementation of operation. With the help of the export command which works as a reverse process of operation. Herewith the help of the export command we can transfer the data from the Hadoop database file system to the Relational database management system. The data which will be exported is processed into records before operation is completed. The export of data is done with two steps, first is to examine the database for metadata and second step involves migration of data.

Advantages of Sqoop :

With the help of Sqoop, we can perform transfer operations of data with a variety of structured data stores like Oracle, Teradata, etc.

Sqoop helps us to perform ETL operations in a very fast and cost-effective manner.

With the help of Sqoop, we can perform parallel processing of data which leads to fasten the overall process.

Sqoop uses the MapReduce mechanism for its operations which also supports fault tolerance.

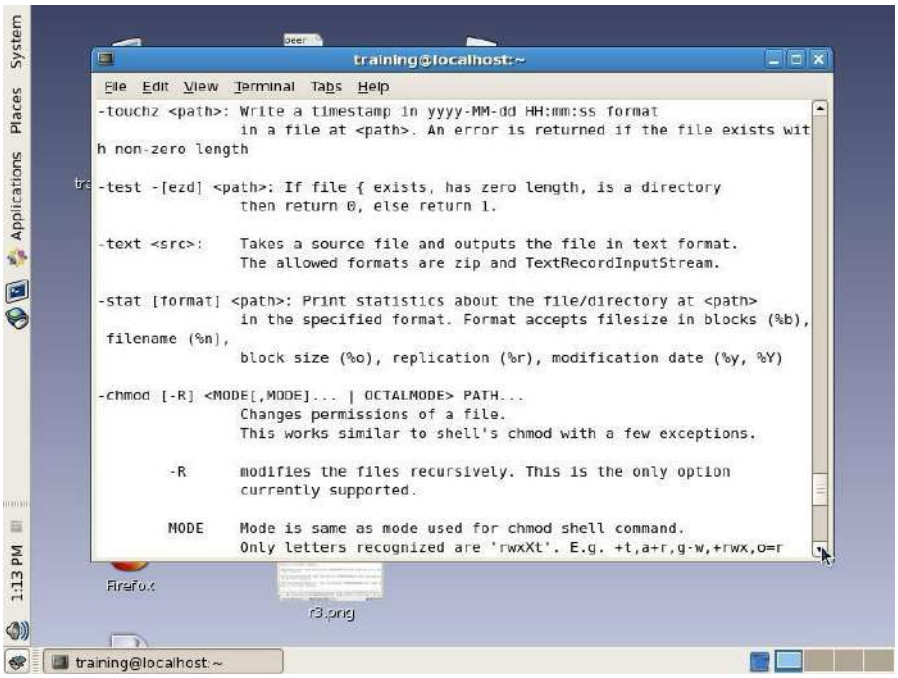
Disadvantages of Sqoop :

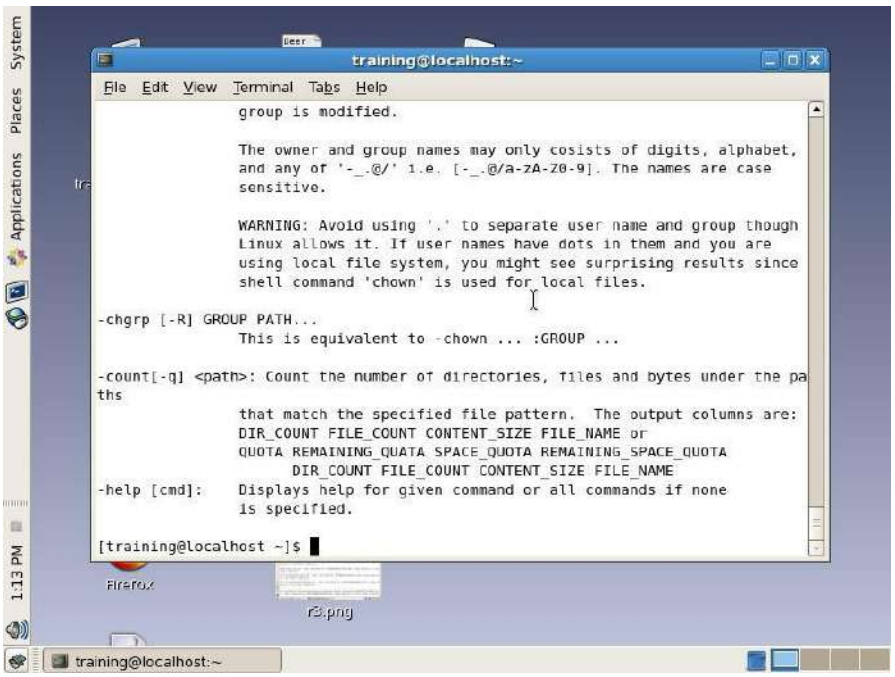
The failure occurs during the implementation of operation needed a special solution to handle the problem.

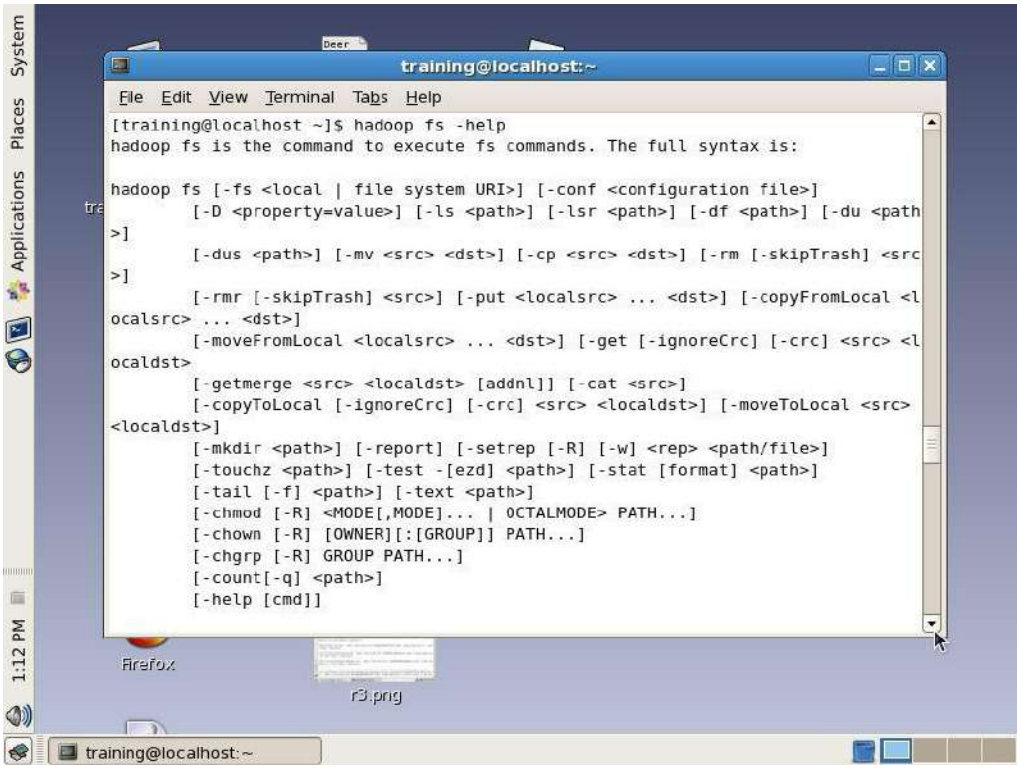
The Sqoop uses JDBC connection to establish a connection with the relational database management system which is an inefficient way.

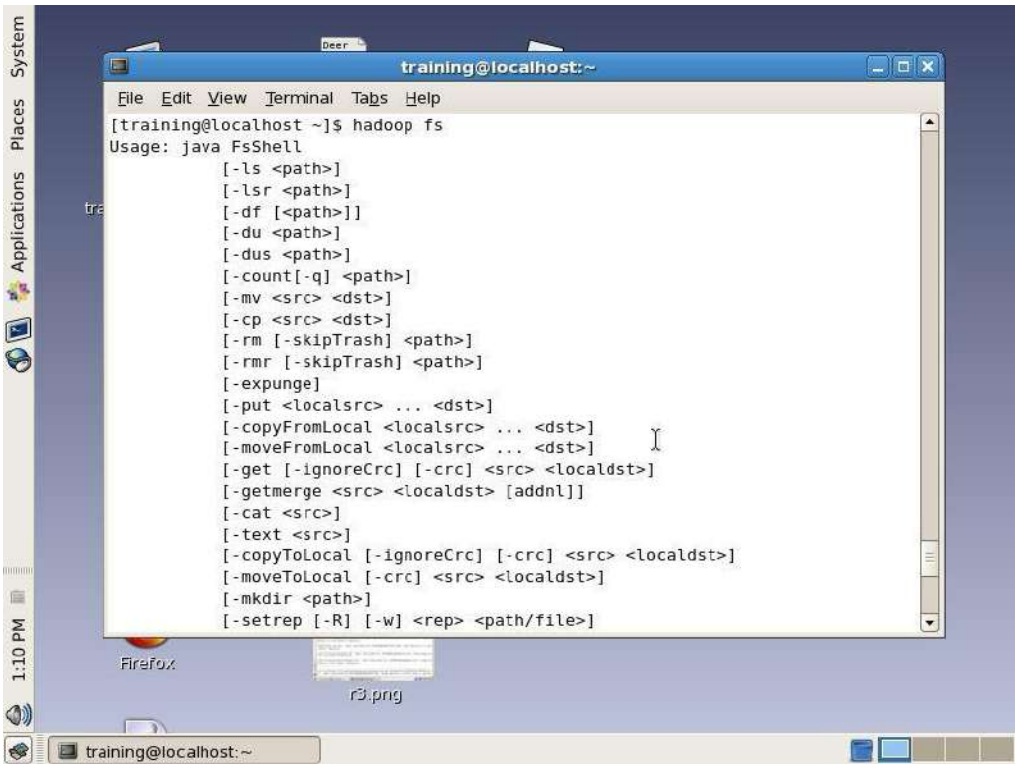
The performance of Sqoop export operation depends upon hardware configuration relational database management system.

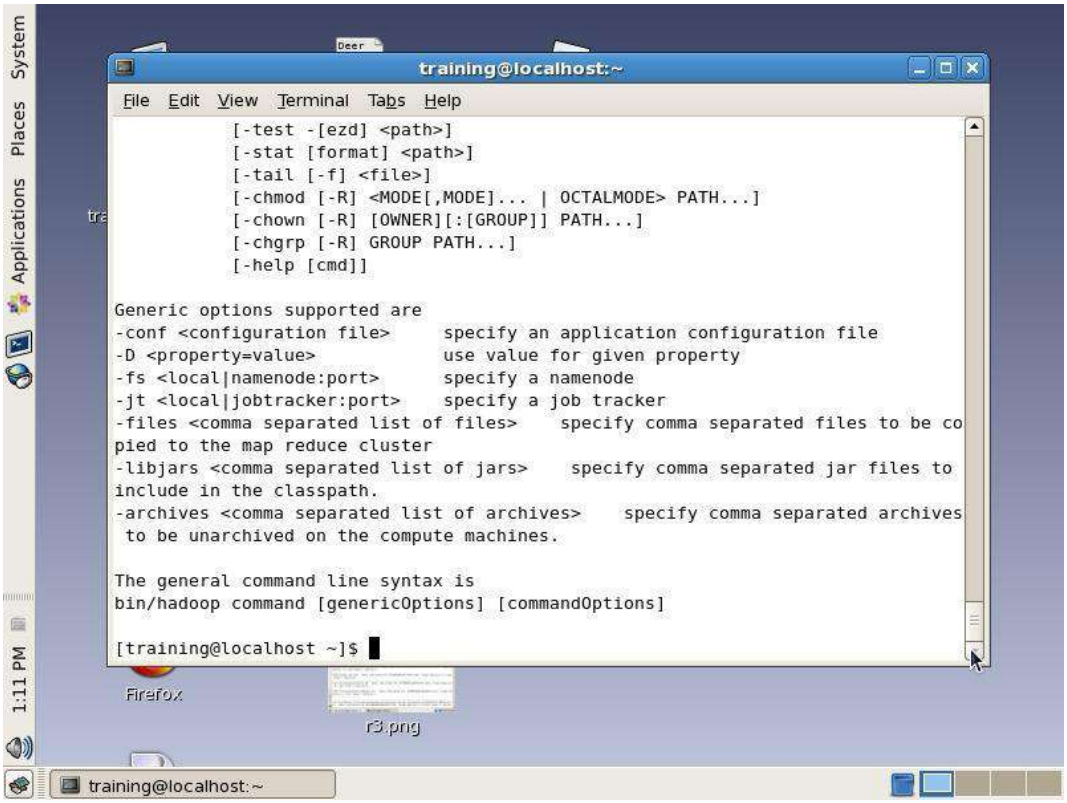
Output:

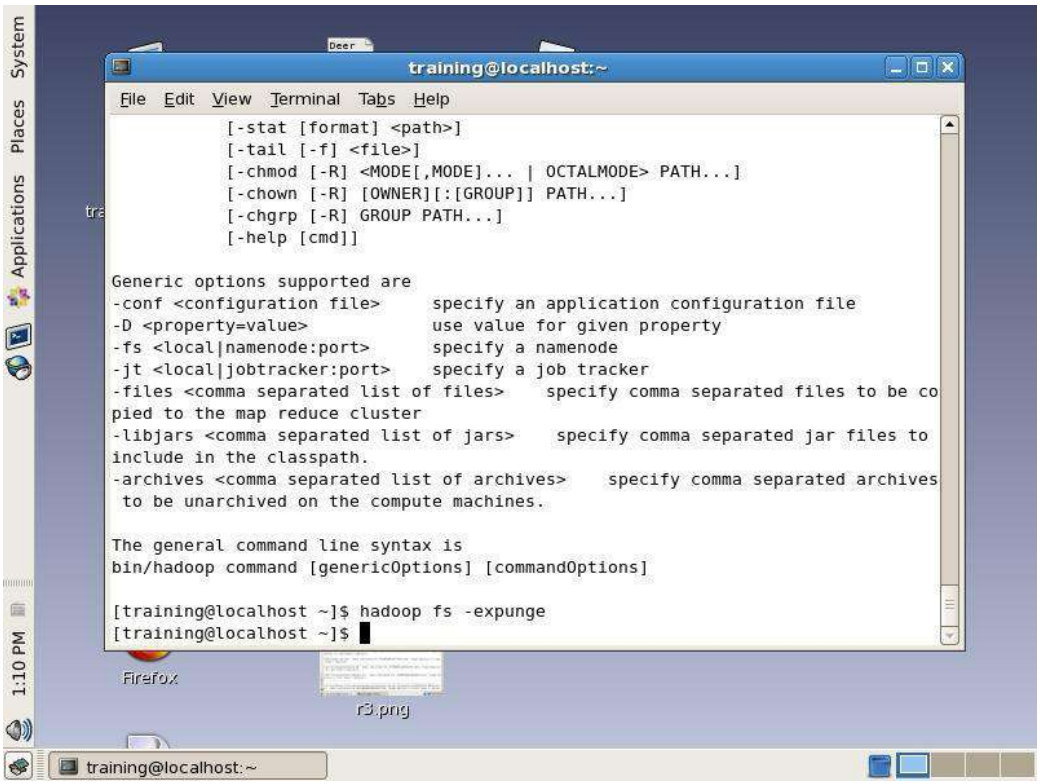


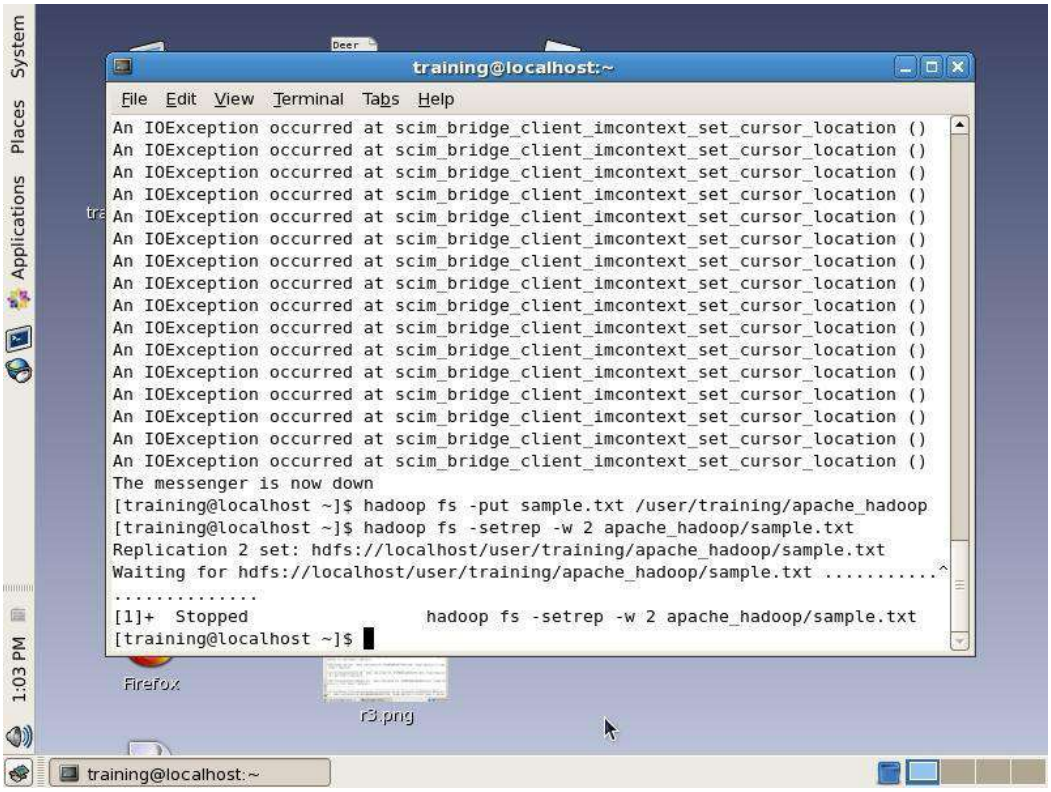


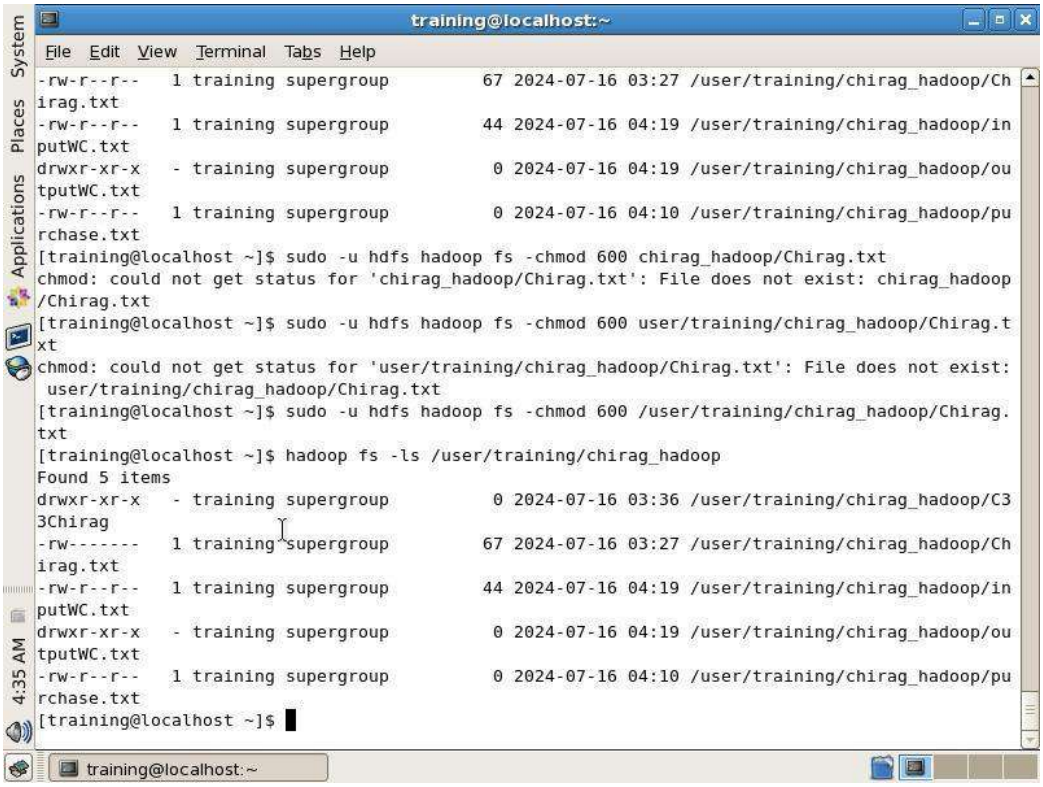

System
Places
Applications
1:12 PM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -help
hadoop fs is the command to execute fs commands. The full syntax is:
hadoop fs [-fs <local | file system URI>] [-conf <configuration file>]
[-D <property=value>] [-ls <path>] [-lsr <path>] [-df <path>] [-du <path
>]
[-dus <path>] [-mv <src> <dst>] [-cp <src> <dst>] [-rm [-skipTrash] <src
>]
[-rmr [-skipTrash] <src>] [-put <localsrc> ... <dst>] [-copyFromLocal <l
ocalsrc> ... <dst>]
[-moveFromLocal <localsrc> ... <dst>] [-get [-ignoreCrc] [-crc] <src> <l
ocaldst>
[-getmerge <src> <localdst> [addnl]] [-cat <src>]
[-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>] [-moveToLocal <src>
<localdst>]
[-mkdir <path>] [-report] [-setrep [-R] [-w] <rep> <path/file>]
[-touchz <path>] [-test -[ezd] <path>] [-stat [format] <path>]
[-tail [-f] <path>] [-text <path>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-count[-q] <path>]
[-help [cmd]]
Firefox
r3.png
training@localhost:~

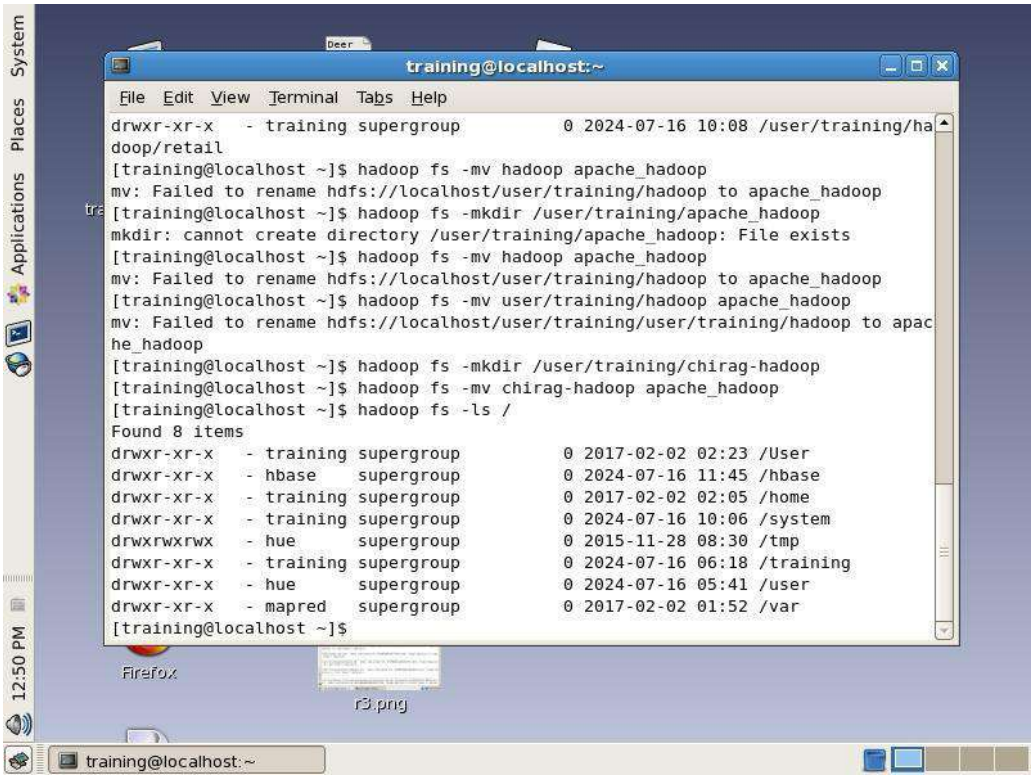

System
Places
Applications
1:10 PM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs
Usage: java FsShell
[-ls <path>]
[-lsr <path>]
[-df [<path>]]
[-du <path>]
[-dus <path>]
[-count[-q] <path>]
[-mv <src> <dst>]
[-cp <src> <dst>]
[-rm [-skipTrash] <path>]
[-rmr [-skipTrash] <path>]
[-expunge]
[-put <localsrc> ... <dst>]
[-copyFromLocal <localsrc> ... <dst>]
[-moveFromLocal <localsrc> ... <dst>]
[-get [-ignoreCrc] [-crc] <src> <localdst>]
[-getmerge <src> <localdst> [addnl]]
[-cat <src>]
[-text <src>]
[-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
[-moveToLocal [-crc] <src> <localdst>]
[-mkdir <path>]
[-setrep [-R] [-w] <rep> <path/file>]
Firefox
r3.png
training@localhost:~


System
Places
Applications
1:11 PM
training@localhost:~
File Edit View Terminal Tabs Help
[-test -[ezd] <path>]
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
Generic options supported are
-conf <configuration file> specify an application configuration file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated files to be co
pied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar files to
include in the classpath.
-archives <comma separated list of archives> specify comma separated archives
to be unarchived on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost ~]$
Firefox
r3.png
training@localhost:~

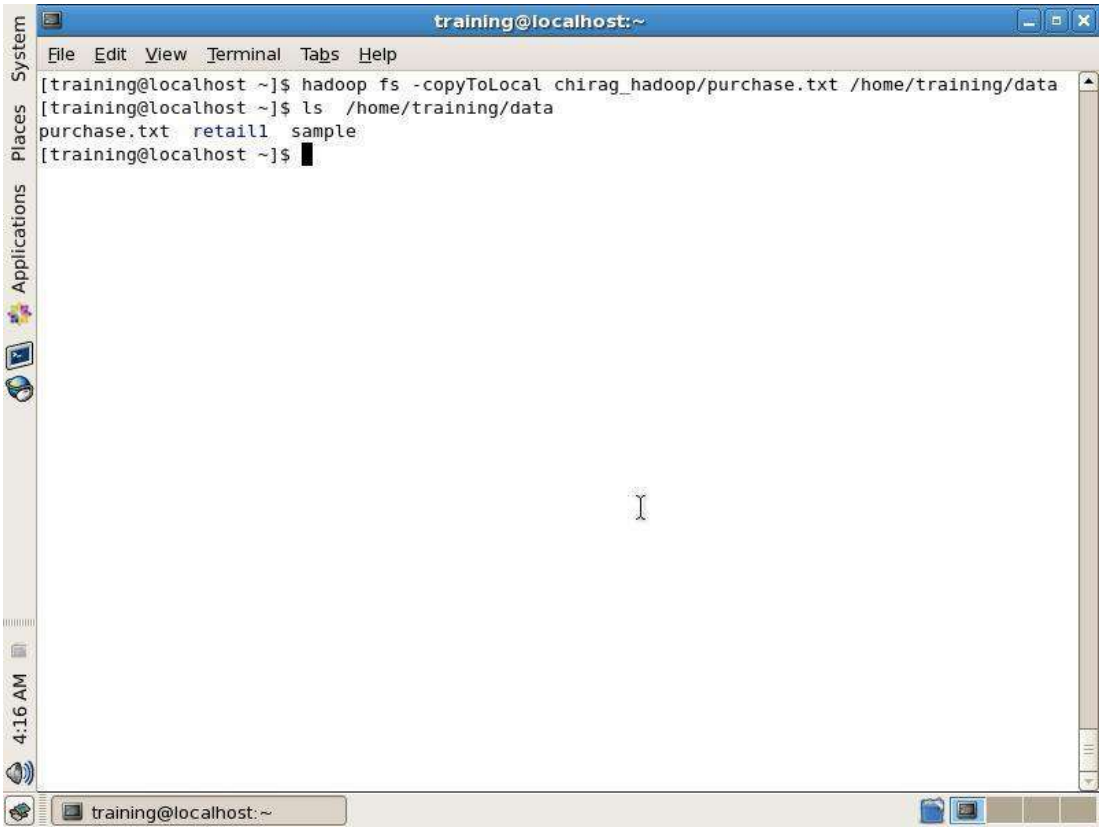

System
Places
Applications
1:10 PM
training@localhost:~
File Edit View Terminal Tabs Help
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
Generic options supported are
-conf <configuration file> specify an application configuration file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated files to be co
pied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar files to
include in the classpath.
-archives <comma separated list of archives> specify comma separated archives
to be unarchived on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost ~]$ hadoop fs -expunge
[training@localhost ~]$
Firefox
r3.png
training@localhost:~

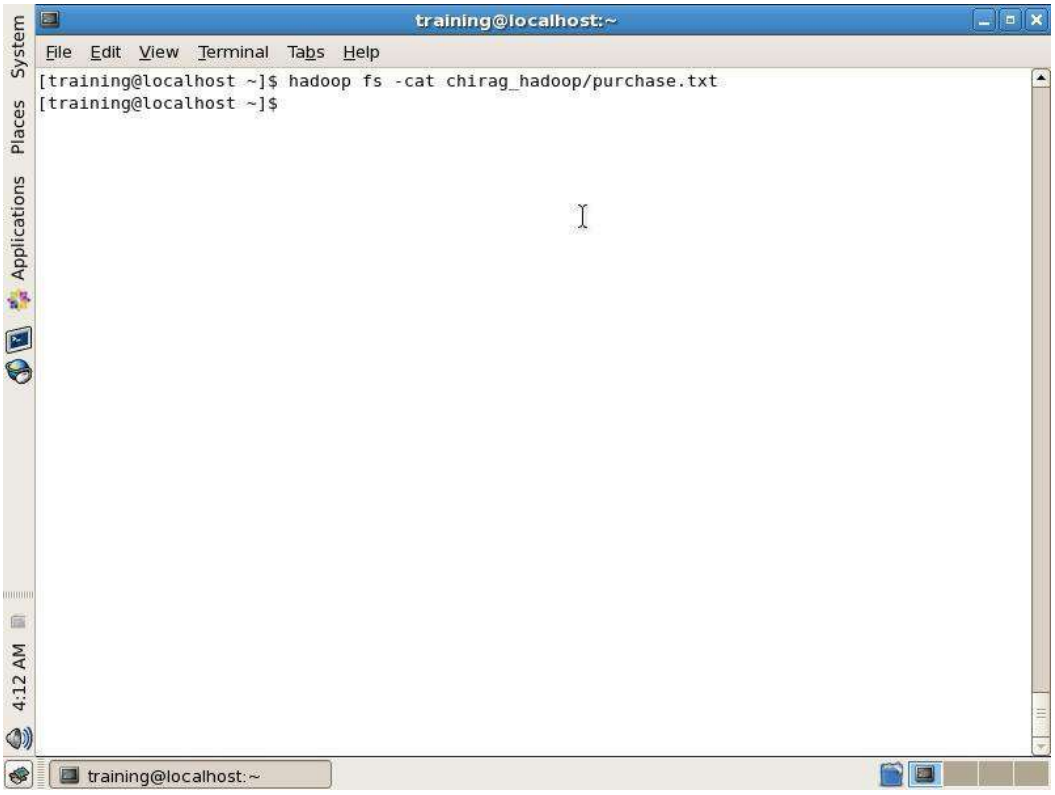

training@localhost:~
File Edit View Terminal Tabs Help
An IOException occurred at scim_bridge_client_imcontext_set_cursor_location ()
The messenger is now down
[training@localhost ~]$ hadoop fs -put sample.txt /user/training/apache_hadoop
[training@localhost ~]$ hadoop fs -setrep -w 2 apache_hadoop/sample.txt
Replication 2 set: hdfs://localhost/user/training/apache_hadoop/sample.txt
Waiting for hdfs://localhost/user/training/apache_hadoop/sample.txt .....
[1]+ Stopped hadoop fs -setrep -w 2 apache_hadoop/sample.txt
[training@localhost ~]$
Firefox
r3.png
System
Places
Applications
1:03 PM
training@localhost:~

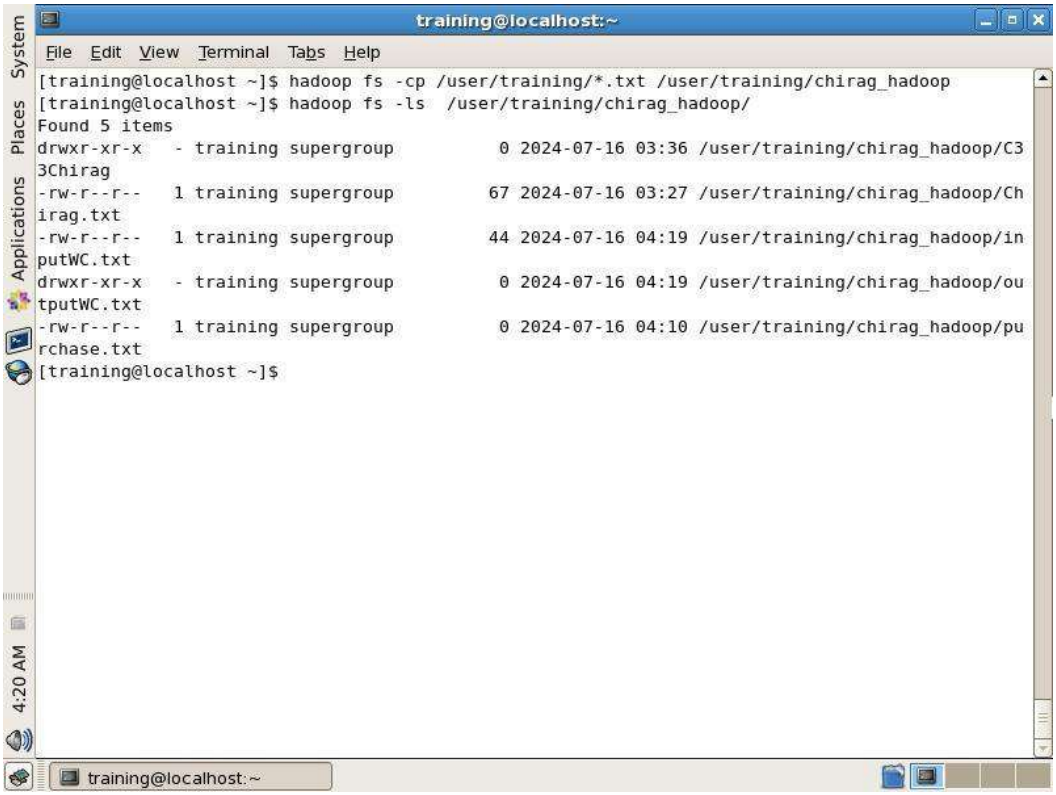

training@localhost:~
File Edit View Terminal Tabs Help
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
-rw-r--r-- 1 training supergroup 44 2024-07-16 04:19 /user/training/chirag_hadoop/inputWC.txt
drwxr-xr-x - training supergroup 0 2024-07-16 04:19 /user/training/chirag_hadoop/outputWC.txt
-rw-r--r-- 1 training supergroup 0 2024-07-16 04:10 /user/training/chirag_hadoop/purchase.txt
[training@localhost ~]$ sudo -u hdfs hadoop fs -chmod 600 chirag_hadoop/Chirag.txt
chmod: could not get status for 'chirag_hadoop/Chirag.txt': File does not exist: chirag_hadoop/Chirag.txt
[training@localhost ~]$ sudo -u hdfs hadoop fs -chmod 600 user/training/chirag_hadoop/Chirag.txt
chmod: could not get status for 'user/training/chirag_hadoop/Chirag.txt': File does not exist: user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$ sudo -u hdfs hadoop fs -chmod 600 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 5 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw------- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
-rw-r--r-- 1 training supergroup 44 2024-07-16 04:19 /user/training/chirag_hadoop/inputWC.txt
drwxr-xr-x - training supergroup 0 2024-07-16 04:19 /user/training/chirag_hadoop/outputWC.txt
-rw-r--r-- 1 training supergroup 0 2024-07-16 04:10 /user/training/chirag_hadoop/purchase.txt
[training@localhost ~]$
System
Places
Applications
4:35 AM
training@localhost:~

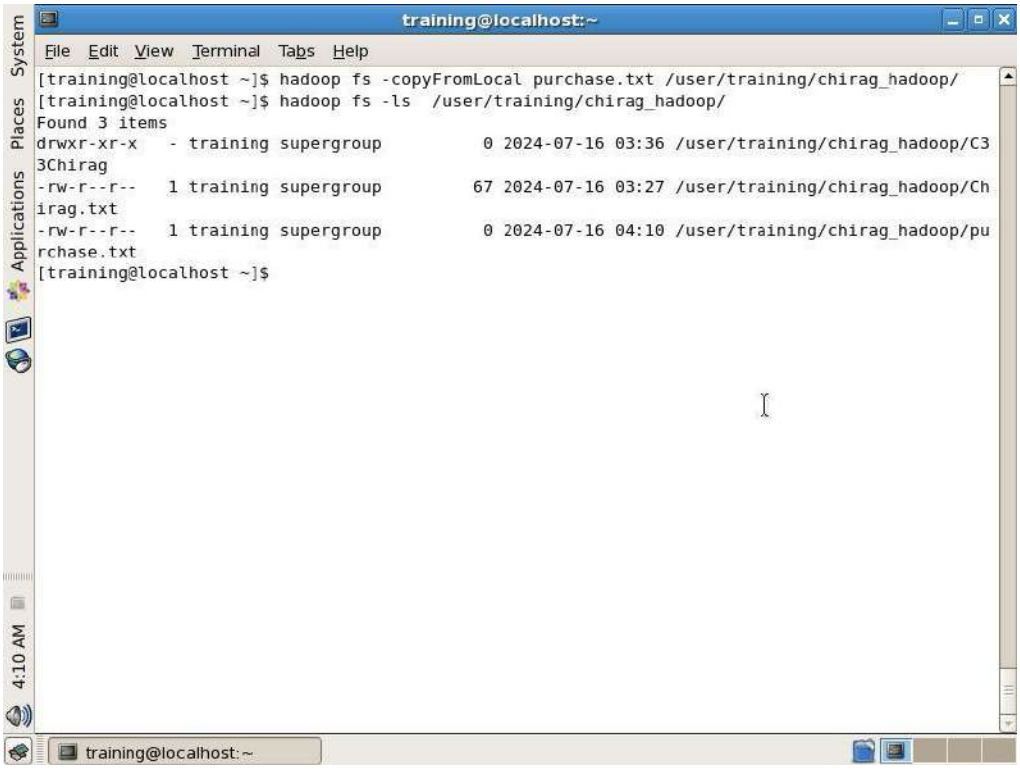


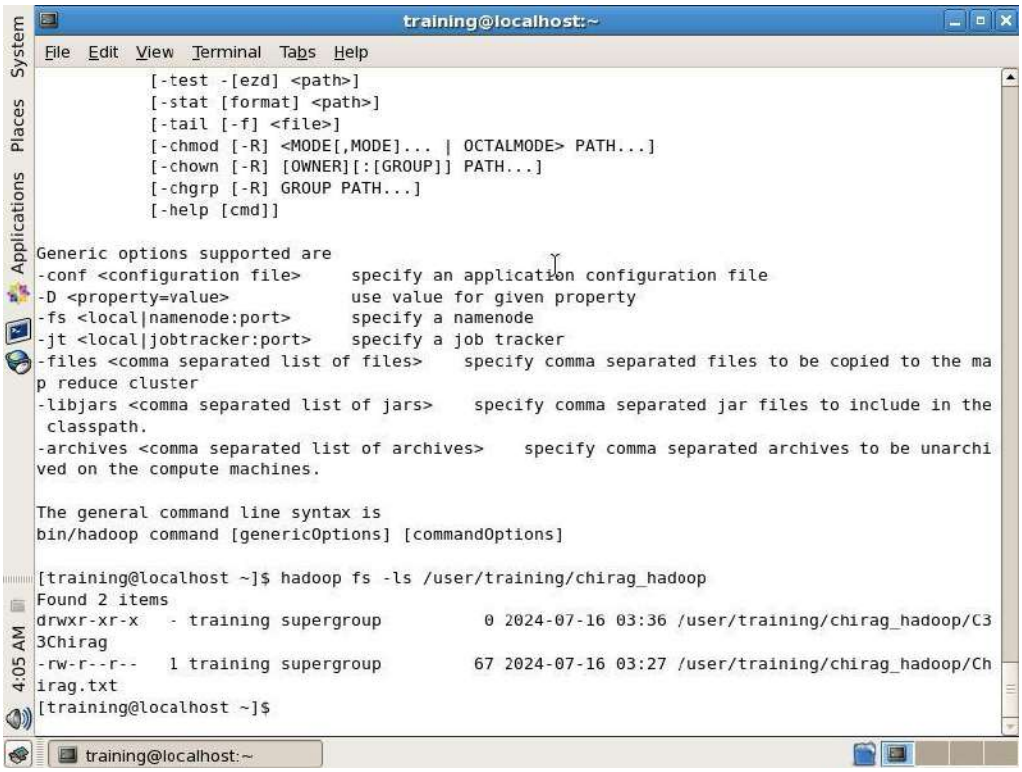

```
[training@localhost ~]$ hadoop fs -get chirag_hadoop/Chirag.txt /home/training
[training@localhost ~]$
```

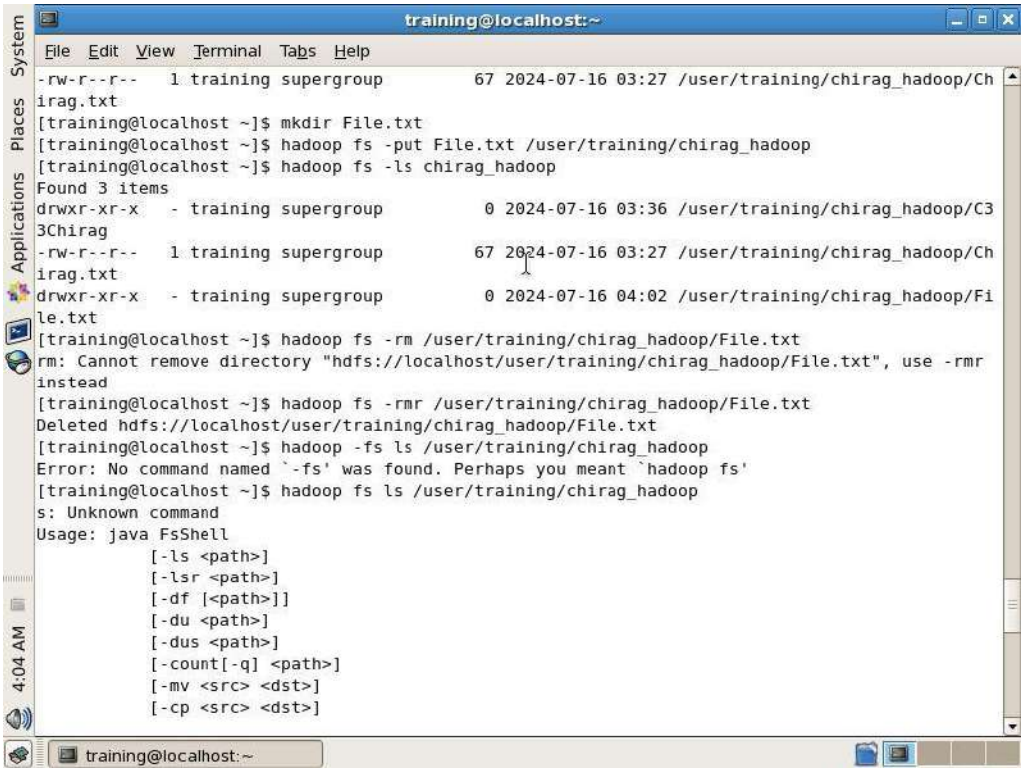


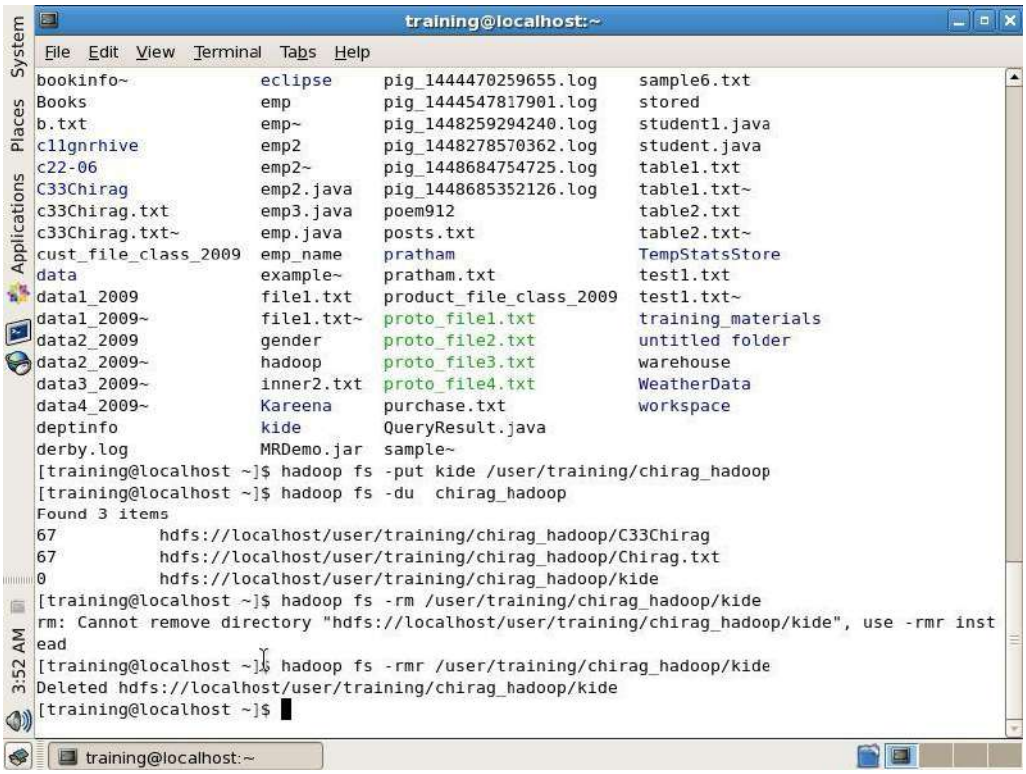

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -cat chirag_hadoop/purchase.txt
[training@localhost ~]$
System
Places
Applications
4:12 AM
training@localhost:~

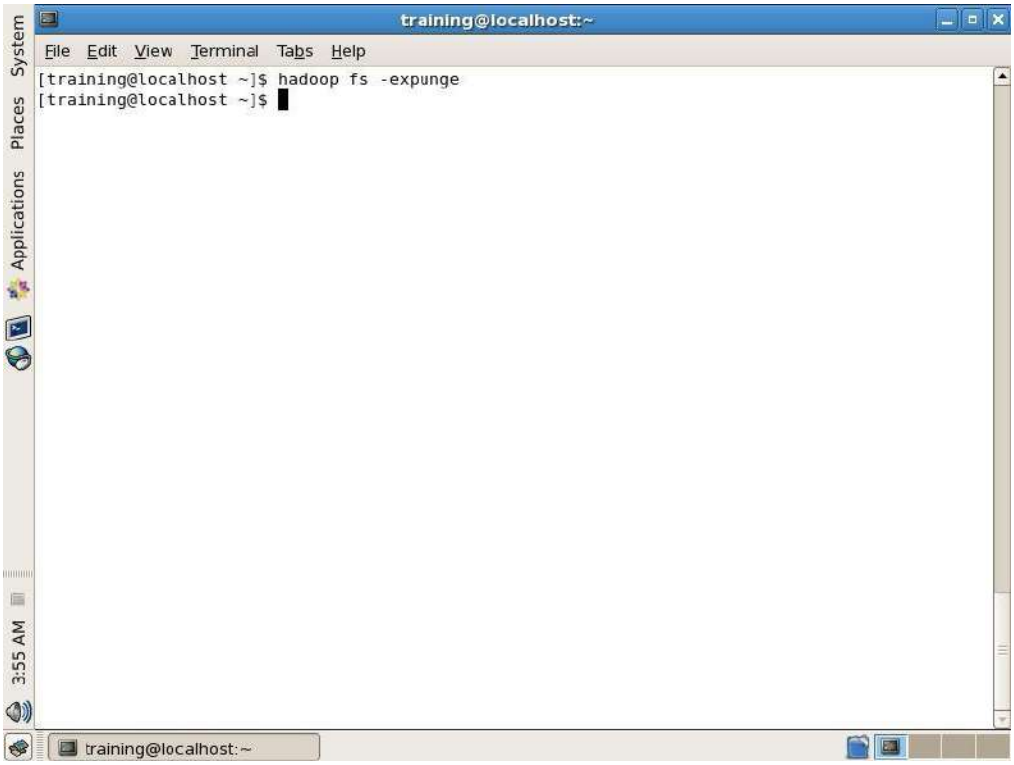

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -cp /user/training/*.txt /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop/
Found 5 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
-rw-r--r-- 1 training supergroup 44 2024-07-16 04:19 /user/training/chirag_hadoop/inputWC.txt
drwxr-xr-x - training supergroup 0 2024-07-16 04:19 /user/training/chirag_hadoop/outputWC.txt
-rw-r--r-- 1 training supergroup 0 2024-07-16 04:10 /user/training/chirag_hadoop/purchase.txt
[training@localhost ~]$
System
Places
Applications
4:20 AM
training@localhost:~

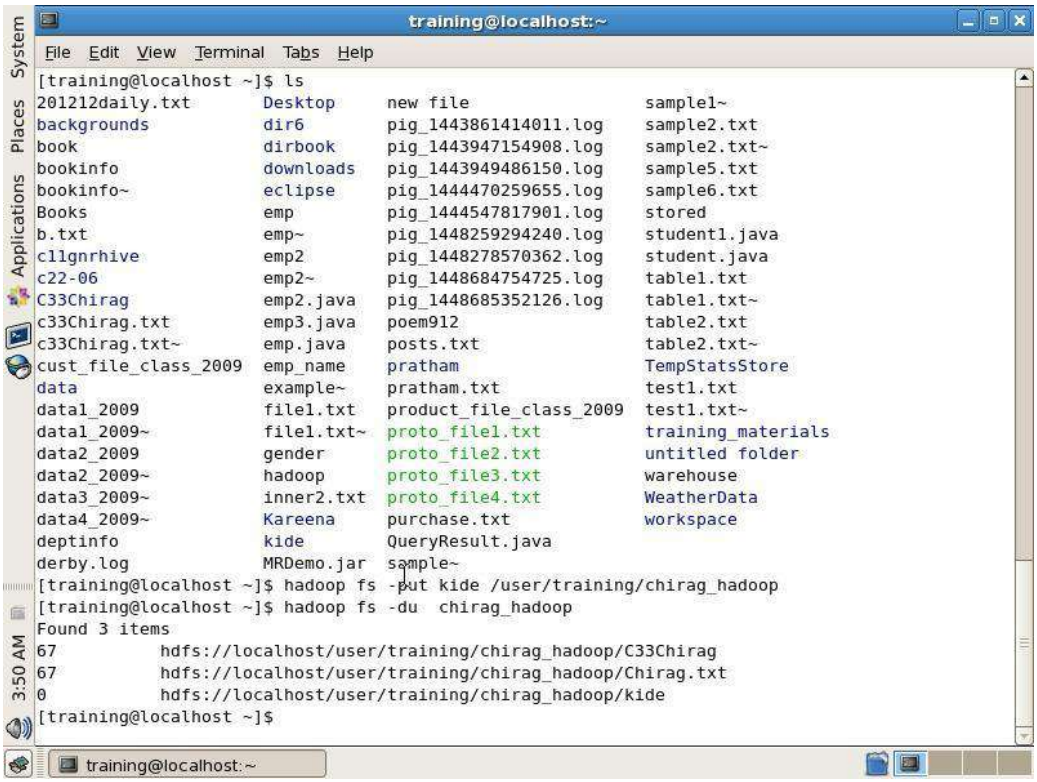

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -copyFromLocal purchase.txt /user/training/chirag_hadoop/
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop/
Found 3 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
-rw-r--r-- 1 training supergroup 0 2024-07-16 04:10 /user/training/chirag_hadoop/purchase.txt
[training@localhost ~]$
System Places Applications
4:10 AM
training@localhost:~

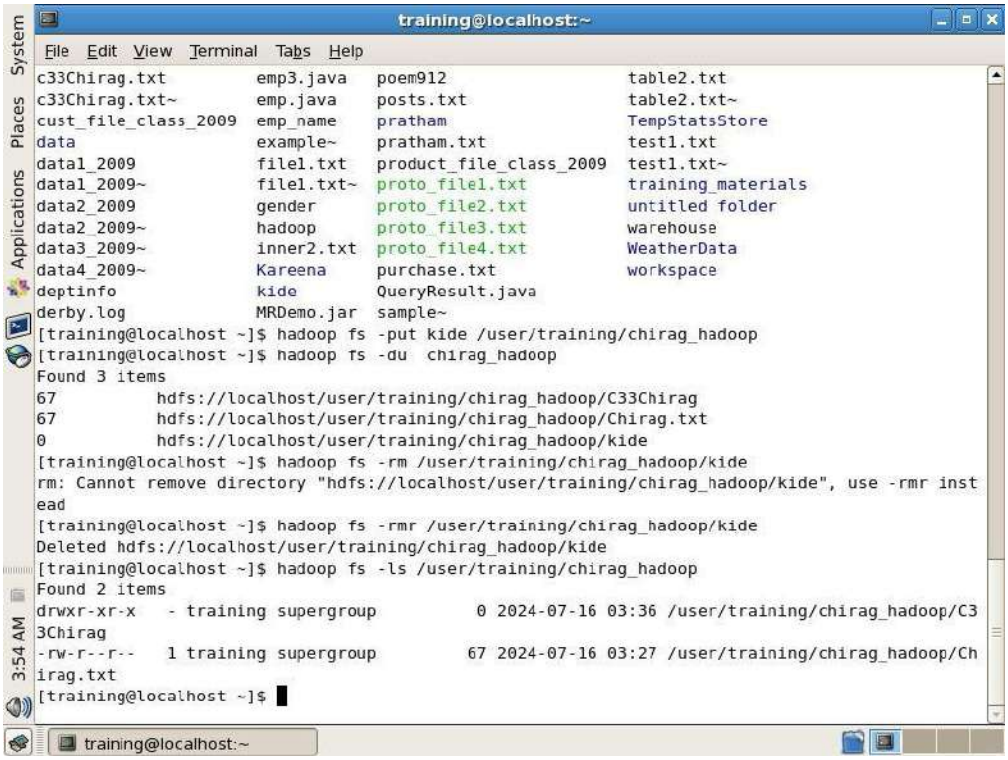

training@localhost:~
File Edit View Terminal Tabs Help
[-test -[ezd] <path>]
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
Generic options supported are
-conf <configuration file> specify an application configuration file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated files to be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar files to include in the classpath.
-archives <comma separated list of archives> specify comma separated archives to be unarchived on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 2 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$
System Places Applications
4:05 AM
training@localhost:~

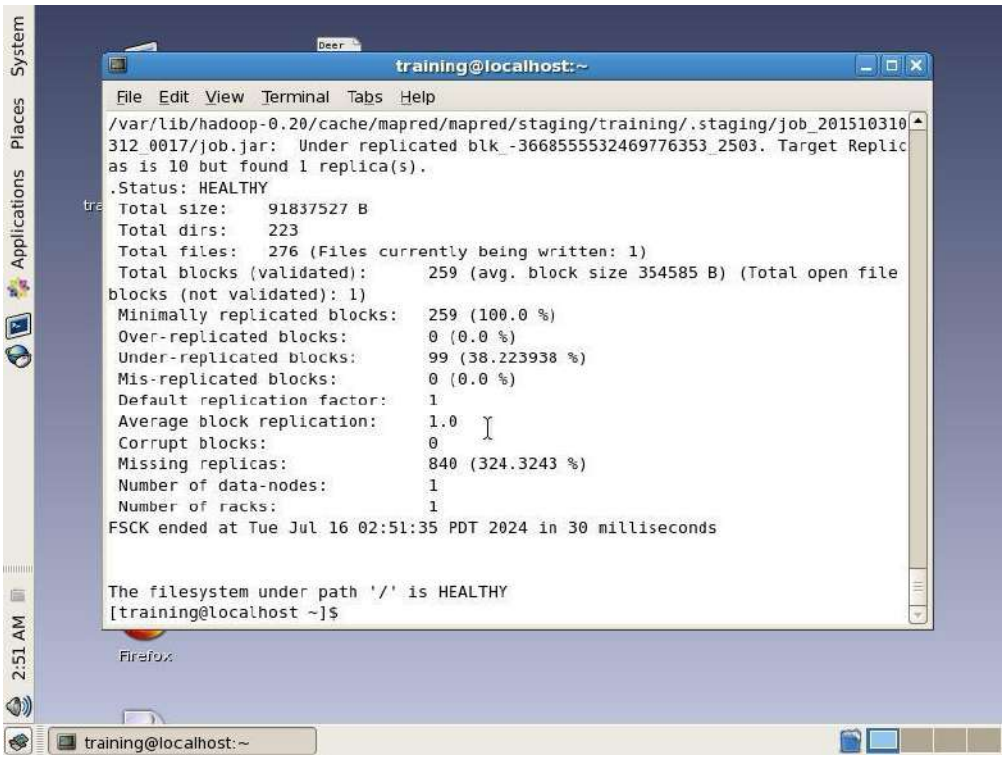

training@localhost:~
File Edit View Terminal Tabs Help
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$ mkdir File.txt
[training@localhost ~]$ hadoop fs -put File.txt /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -ls chirag_hadoop
Found 3 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
drwxr-xr-x - training supergroup 0 2024-07-16 04:02 /user/training/chirag_hadoop/File.txt
[training@localhost ~]$ hadoop fs -rm /user/training/chirag_hadoop/File.txt
rm: Cannot remove directory "hdfs://localhost/user/training/chirag_hadoop/File.txt", use -rmr instead
[training@localhost ~]$ hadoop fs -rmr /user/training/chirag_hadoop/File.txt
Deleted hdfs://localhost/user/training/chirag_hadoop/File.txt
[training@localhost ~]$ hadoop -fs ls /user/training/chirag_hadoop
Error: No command named `-fs' was found. Perhaps you meant `hadoop fs'
[training@localhost ~]$ hadoop fs ls /user/training/chirag_hadoop
s: Unknown command
Usage: java FsShell
[-ls <path>]
[-lsr <path>]
[-df [<path>]]
[-du <path>]
[-dus <path>]
[-count[-q] <path>]
[-mv <src> <dst>]
[-cp <src> <dst>]
System Places Applications
4:04 AM
training@localhost:~

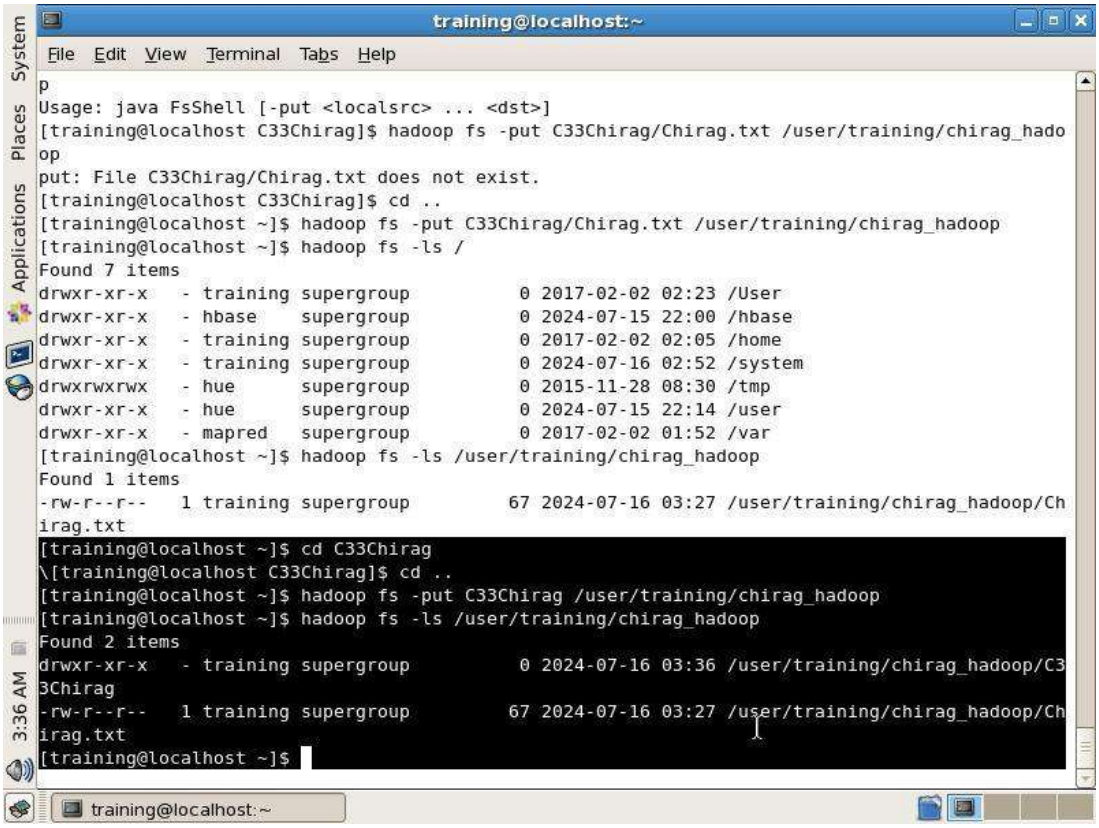

training@localhost:~
File Edit View Terminal Tabs Help
bookinfo~ eclipse pig_1444470259655.log sample6.txt
Books emp pig_1444547817901.log stored
b.txt emp~ pig_1448259294240.log student1.java
c11gnrhive emp2 pig_1448278570362.log student.java
c22-06 emp2~ pig_1448684754725.log table1.txt
C33Chirag emp2.java pig_1448685352126.log table1.txt~
c33Chirag.txt emp3.java poem912 table2.txt
c33Chirag.txt~ emp.java posts.txt table2.txt~
cust_file_class_2009 emp_name pratham TempStatsStore
data example~ pratham.txt test1.txt
data1_2009 file1.txt product_file_class_2009 test1.txt~
data1_2009~ file1.txt~ proto_file1.txt training_materials
data2_2009 gender proto_file2.txt untitled folder
data2_2009~ hadoop proto_file3.txt warehouse
data3_2009~ inner2.txt proto_file4.txt WeatherData
data4_2009~ Kareena purchase.txt workspace
deptinfo kide QueryResult.java
derby.log MRDemo.jar sample~
[training@localhost ~]$ hadoop fs -put kide /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -du chirag_hadoop
Found 3 items
67 hdfs://localhost/user/training/chirag_hadoop/C33Chirag
67 hdfs://localhost/user/training/chirag_hadoop/Chirag.txt
0 hdfs://localhost/user/training/chirag_hadoop/kide
[training@localhost ~]$ hadoop fs -rm /user/training/chirag_hadoop/kide
rm: Cannot remove directory "hdfs://localhost/user/training/chirag_hadoop/kide", use -rmr instead
[training@localhost ~]$ hadoop fs -rmr /user/training/chirag_hadoop/kide
Deleted hdfs://localhost/user/training/chirag_hadoop/kide
[training@localhost ~]$
System Places Applications
3:52 AM
training@localhost:~

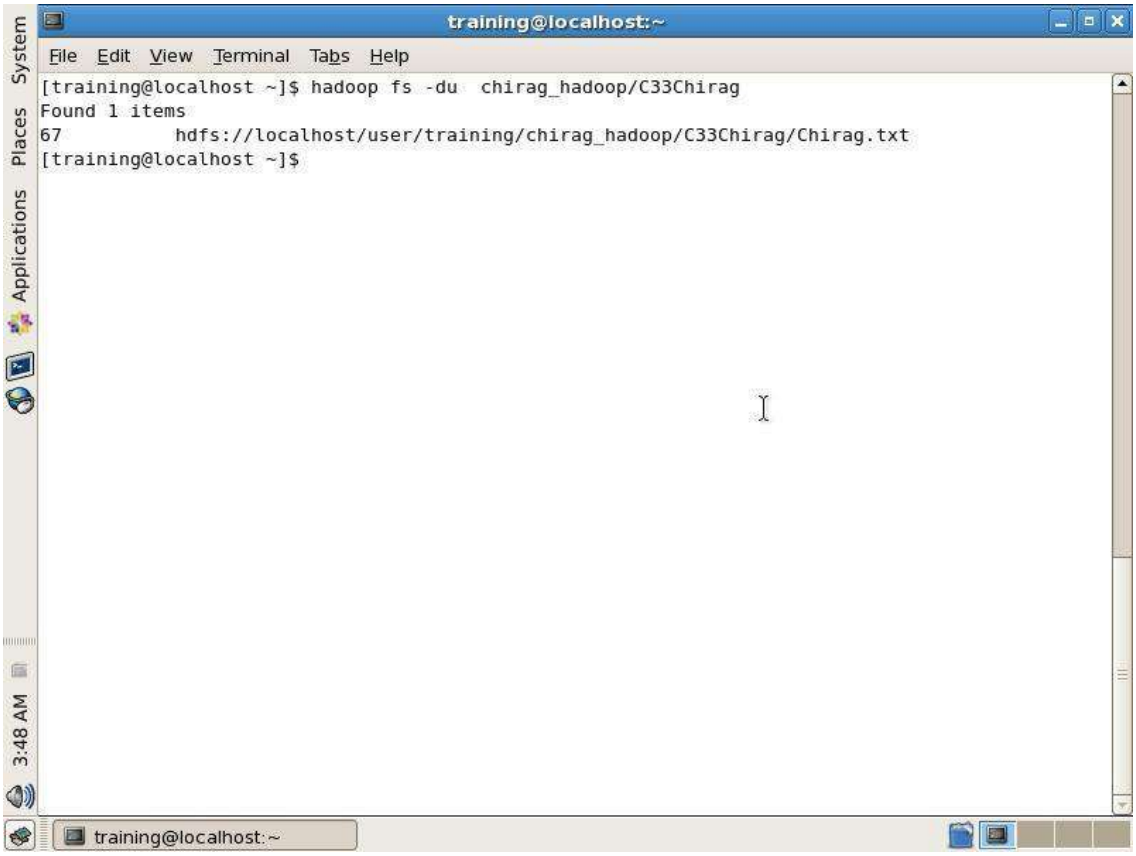

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -expunge
[training@localhost ~]$
System
Places
Applications
3:55 AM
training@localhost:~

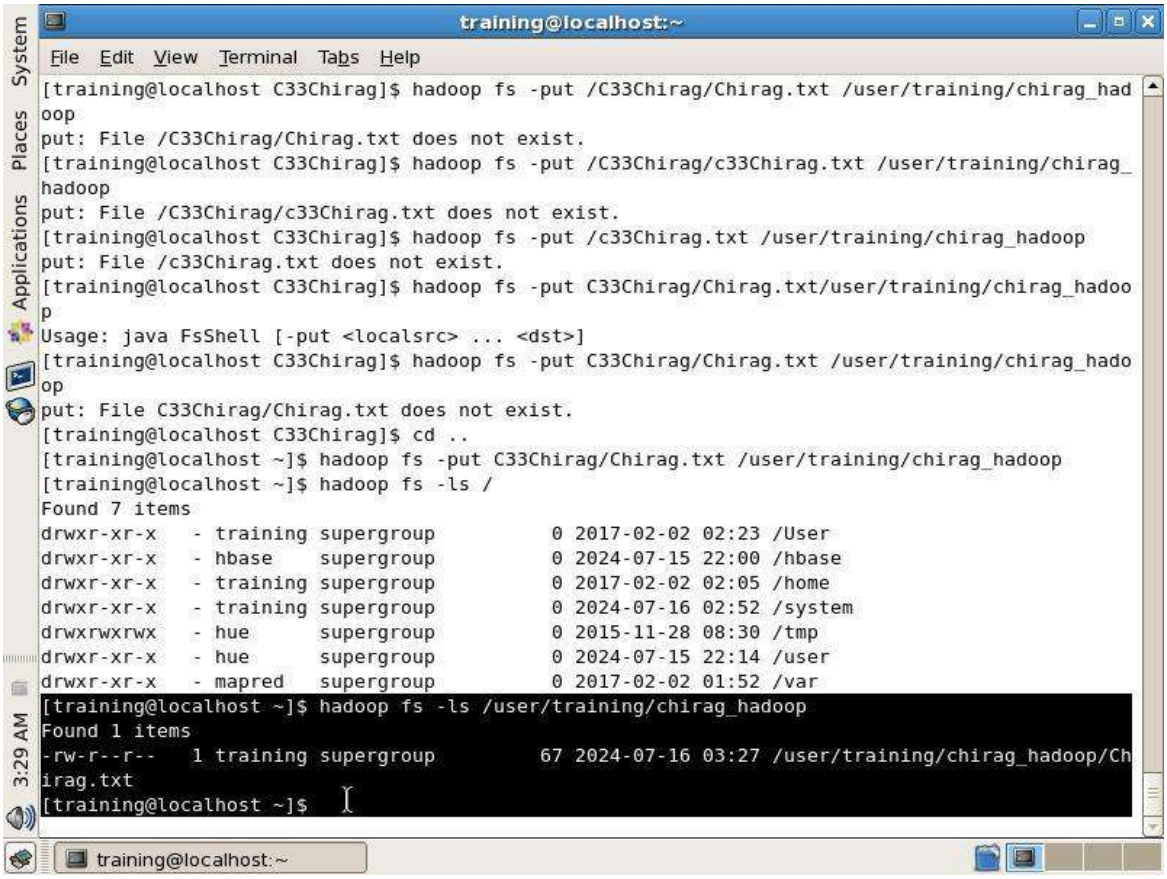

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ ls
201212daily.txt Desktop new file sample1~
backgrounds dir6 pig_1443861414011.log sample2.txt
book dirbook pig_1443947154908.log sample2.txt~
bookinfo downloads pig_1443949486150.log sample5.txt
bookinfo~ eclipse pig_1444470259655.log sample6.txt
Books emp pig_1444547817901.log stored
b.txt emp~ pig_1448259294240.log student1.java
c11gnrhive emp2 pig_1448278570362.log student.java
c22-06 emp2~ pig_1448684754725.log table1.txt
C33Chirag emp2.java pig_1448685352126.log table1.txt~
c33Chirag.txt emp3.java poem912 table2.txt
c33Chirag.txt~ emp.java posts.txt table2.txt~
cust_file_class_2009 emp_name pratham TempStatsStore
data example~ pratham.txt test1.txt
data1_2009 file1.txt product_file_class_2009 test1.txt~
data1_2009~ file1.txt~ proto_file1.txt training_materials
data2_2009 gender proto_file2.txt untitled folder
data2_2009~ hadoop proto_file3.txt warehouse
data3_2009~ inner2.txt proto_file4.txt WeatherData
data4_2009~ Kareena purchase.txt workspace
deptinfo kide QueryResult.java
derby.log MRDemo.jar sample~
[training@localhost ~]$ hadoop fs -put kide /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -du chirag_hadoop
Found 3 items
67 hdfs://localhost/user/training/chirag_hadoop/C33Chirag
67 hdfs://localhost/user/training/chirag_hadoop/Chirag.txt
0 hdfs://localhost/user/training/chirag_hadoop/kide
[training@localhost ~]$
System
Places
Applications
3:50 AM
training@localhost:~

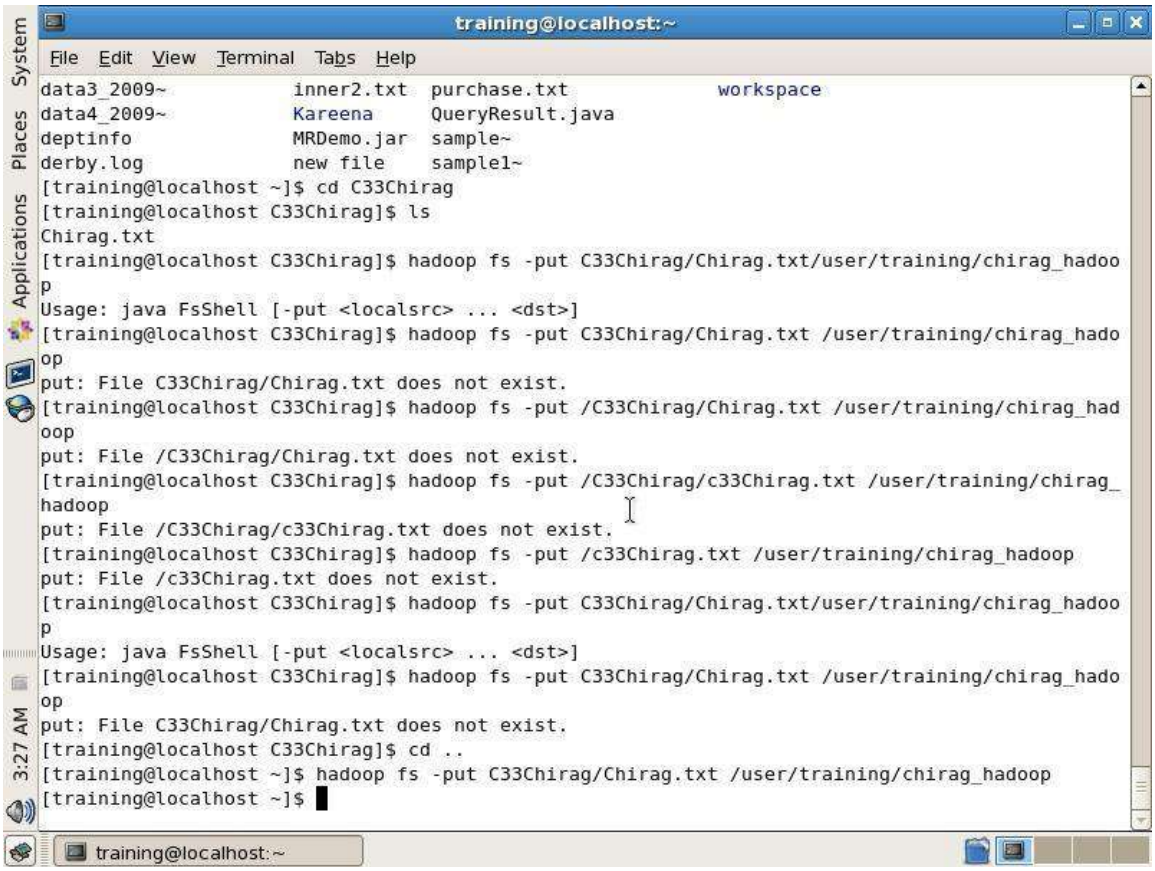

training@localhost:~
File Edit View Terminal Tabs Help
c33Chirag.txt emp3.java poem912 table2.txt
c33Chirag.txt~ emp.java posts.txt table2.txt~
cust_file_class_2009 emp_name pratham TempStatsStore
data example~ pratham.txt test1.txt
data1_2009 file1.txt product_file_class_2009 test1.txt~
data1_2009~ file1.txt~ proto_file1.txt training_materials
data2_2009 gender proto_file2.txt untitled folder
data2_2009~ hadoop proto_file3.txt warehouse
data3_2009~ inner2.txt proto_file4.txt WeatherData
data4_2009~ Kareena purchase.txt workspace
deptinfo kide QueryResult.java
derby.log MRDemo.jar sample~
[training@localhost ~]$ hadoop fs -put kide /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -du chirag_hadoop
Found 3 items
67 hdfs://localhost/user/training/chirag_hadoop/C33Chirag
67 hdfs://localhost/user/training/chirag_hadoop/Chirag.txt
0 hdfs://localhost/user/training/chirag_hadoop/kide
[training@localhost ~]$ hadoop fs -rm /user/training/chirag_hadoop/kide
rm: Cannot remove directory "hdfs://localhost/user/training/chirag_hadoop/kide", use -rmr instead
[training@localhost ~]$ hadoop fs -rmr /user/training/chirag_hadoop/kide
Deleted hdfs://localhost/user/training/chirag_hadoop/kide
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 2 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$
System Places Applications
3:54 AM
training@localhost:~

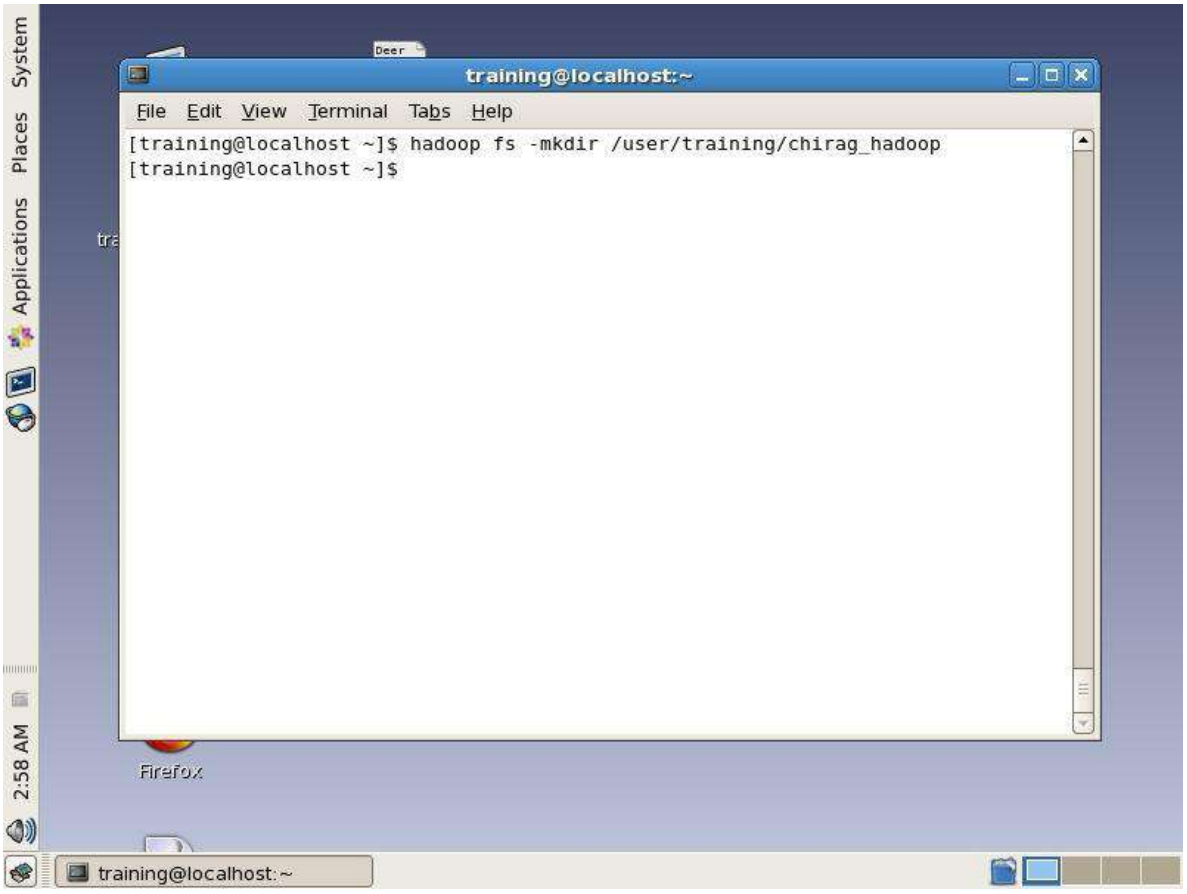

training@localhost:~
File Edit View Terminal Tabs Help
/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_201510310312_0017/job.jar: Under replicated blk_-3668555532469776353_2503. Target Replicas is 10 but found 1 replica(s).
.Status: HEALTHY
Total size: 91837527 B
Total dirs: 223
Total files: 276 (Files currently being written: 1)
Total blocks (validated): 259 (avg. block size 354585 B) (Total open file blocks (not validated): 1)
Minimally replicated blocks: 259 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 99 (38.223938 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 840 (324.3243 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Tue Jul 16 02:51:35 PDT 2024 in 30 milliseconds
The filesystem under path '/' is HEALTHY
[training@localhost ~]$
Firefox
System Places Applications
2:51 AM
training@localhost:~

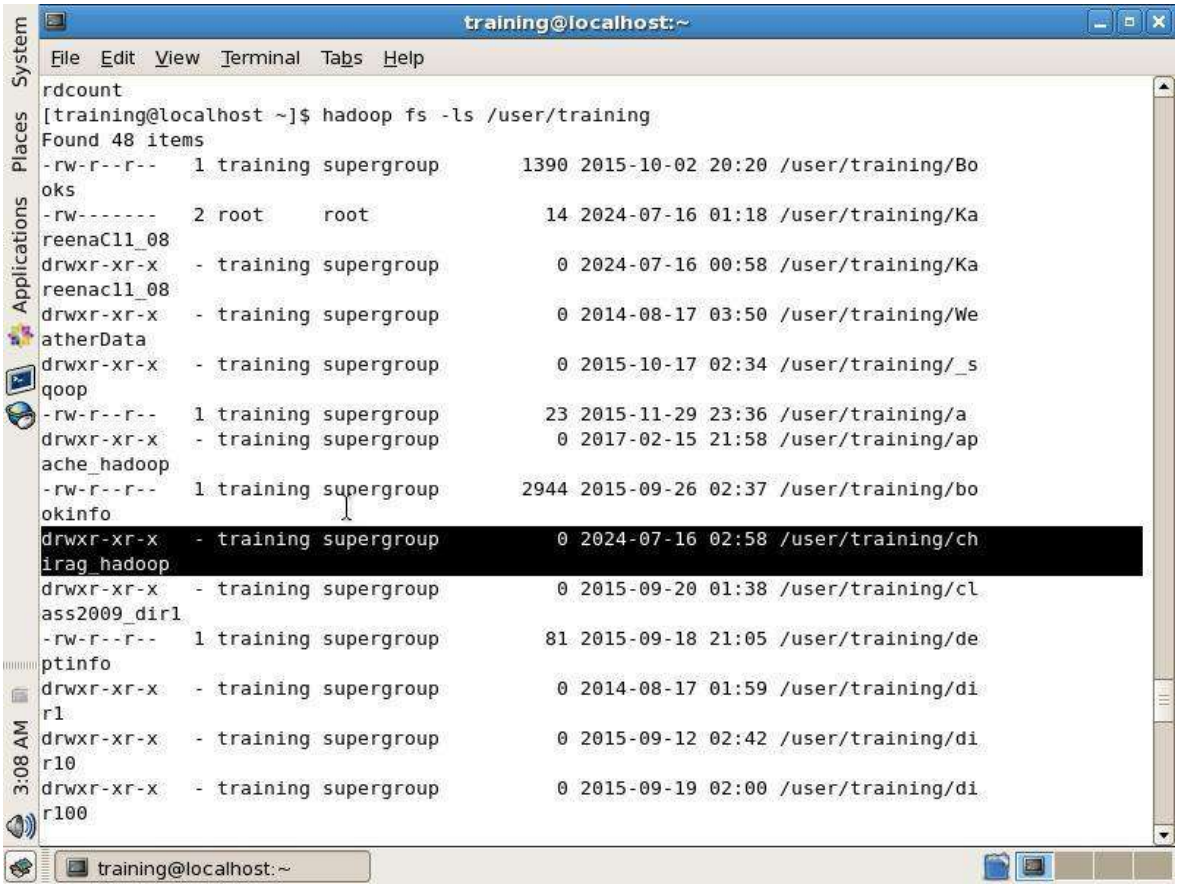

training@localhost:~
File Edit View Terminal Tabs Help
p
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hadoop
put: File C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ cd ..
[training@localhost ~]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -ls /
Found 7 items
drwxr-xr-x - training supergroup 0 2017-02-02 02:23 /User
drwxr-xr-x - hbase supergroup 0 2024-07-15 22:00 /hbase
drwxr-xr-x - training supergroup 0 2017-02-02 02:05 /home
drwxr-xr-x - training supergroup 0 2024-07-16 02:52 /system
drwxrwxrwx - hue supergroup 0 2015-11-28 08:30 /tmp
drwxr-xr-x - hue supergroup 0 2024-07-15 22:14 /user
drwxr-xr-x - mapred supergroup 0 2017-02-02 01:52 /var
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 1 items
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$ cd C33Chirag
\[training@localhost C33Chirag]$ cd ..
[training@localhost ~]$ hadoop fs -put C33Chirag /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 2 items
drwxr-xr-x - training supergroup 0 2024-07-16 03:36 /user/training/chirag_hadoop/C33Chirag
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Chirag.txt
[training@localhost ~]$
System Places Applications
3:36 AM
training@localhost:~

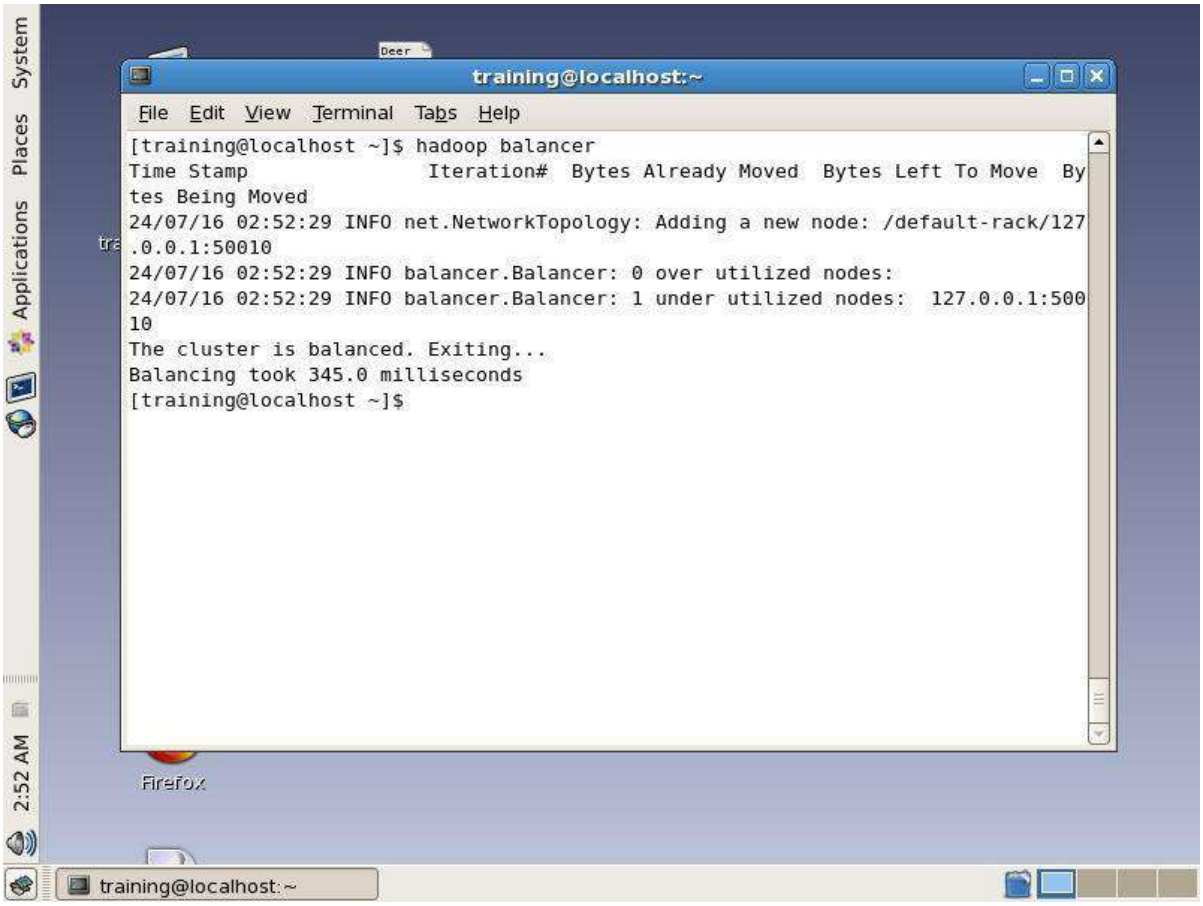

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -du chirag_hadoop/C33Chirag
Found 1 items
67 hdfs://localhost/user/training/chirag_hadoop/C33Chirag/Chirag.txt
[training@localhost ~]$
System Places Applications
3:48 AM
training@localhost:~

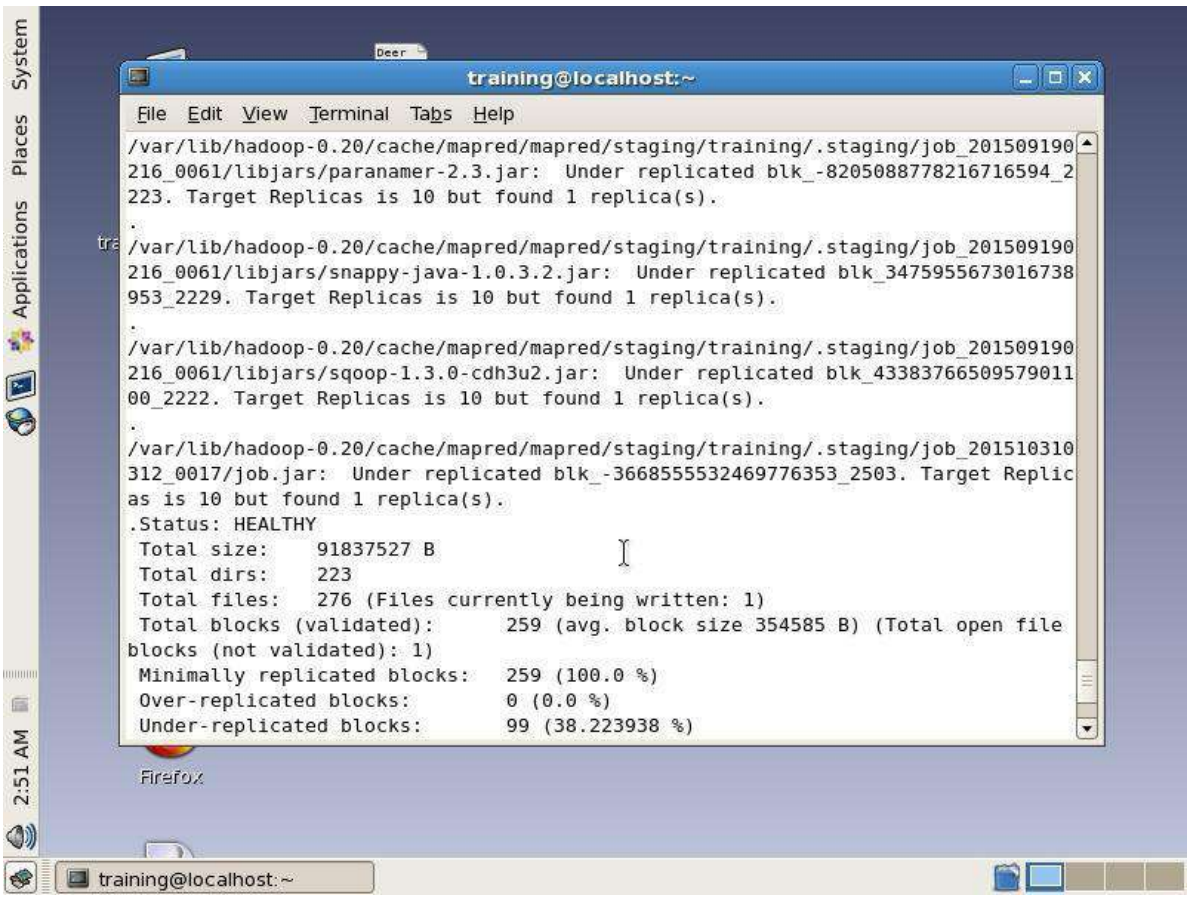

training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost C33Chirag]$ hadoop fs -put /C33Chirag/Chirag.txt /user/training/chirag_had
oop
put: File /C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put /C33Chirag/c33Chirag.txt /user/training/chirag_
hadoop
put: File /C33Chirag/c33Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put /c33Chirag.txt /user/training/chirag_hadoop
put: File /c33Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt/user/training/chirag_hadoo
p
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hado
op
put: File C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ cd ..
[training@localhost ~]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hadoop
[training@localhost ~]$ hadoop fs -ls /
Found 7 items
drwxr-xr-x - training supergroup 0 2017-02-02 02:23 /User
drwxr-xr-x - hbase supergroup 0 2024-07-15 22:00 /hbase
drwxr-xr-x - training supergroup 0 2017-02-02 02:05 /home
drwxr-xr-x - training supergroup 0 2024-07-16 02:52 /system
drwxrwxrwx - hue supergroup 0 2015-11-28 08:30 /tmp
drwxr-xr-x - hue supergroup 0 2024-07-15 22:14 /user
drwxr-xr-x - mapred supergroup 0 2017-02-02 01:52 /var
[training@localhost ~]$ hadoop fs -ls /user/training/chirag_hadoop
Found 1 items
-rw-r--r-- 1 training supergroup 67 2024-07-16 03:27 /user/training/chirag_hadoop/Ch
irag.txt
[training@localhost ~]$
System Places Applications
3:29 AM
training@localhost:~

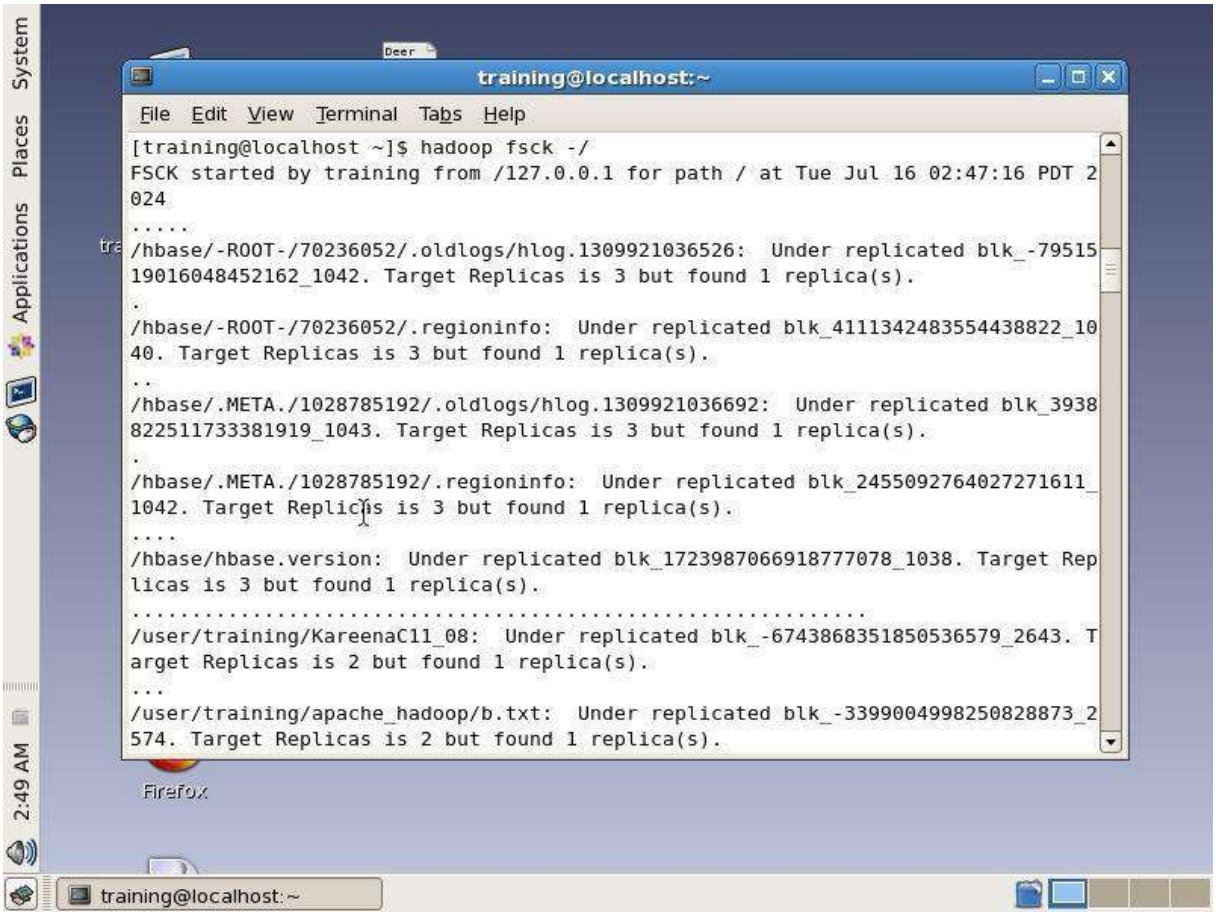

training@localhost:~
File Edit View Terminal Tabs Help
data3_2009~ inner2.txt purchase.txt workspace
data4_2009~ Kareena QueryResult.java
deptinfo MRDemo.jar sample~
derby.log new file sample1~
[training@localhost ~]$ cd C33Chirag
[training@localhost C33Chirag]$ ls
Chirag.txt
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt/user/training/chirag_hadoo
p
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hado
op
put: File C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put /C33Chirag/Chirag.txt /user/training/chirag_had
oop
put: File /C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put /C33Chirag/c33Chirag.txt /user/training/chirag_
hadoop
put: File /C33Chirag/c33Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put /c33Chirag.txt /user/training/chirag_hadoop
put: File /c33Chirag.txt does not exist.
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt/user/training/chirag_hadoo
p
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost C33Chirag]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hado
op
put: File C33Chirag/Chirag.txt does not exist.
[training@localhost C33Chirag]$ cd ..
[training@localhost ~]$ hadoop fs -put C33Chirag/Chirag.txt /user/training/chirag_hadoop
[training@localhost ~]$
System Places Applications
3:27 AM
training@localhost:~

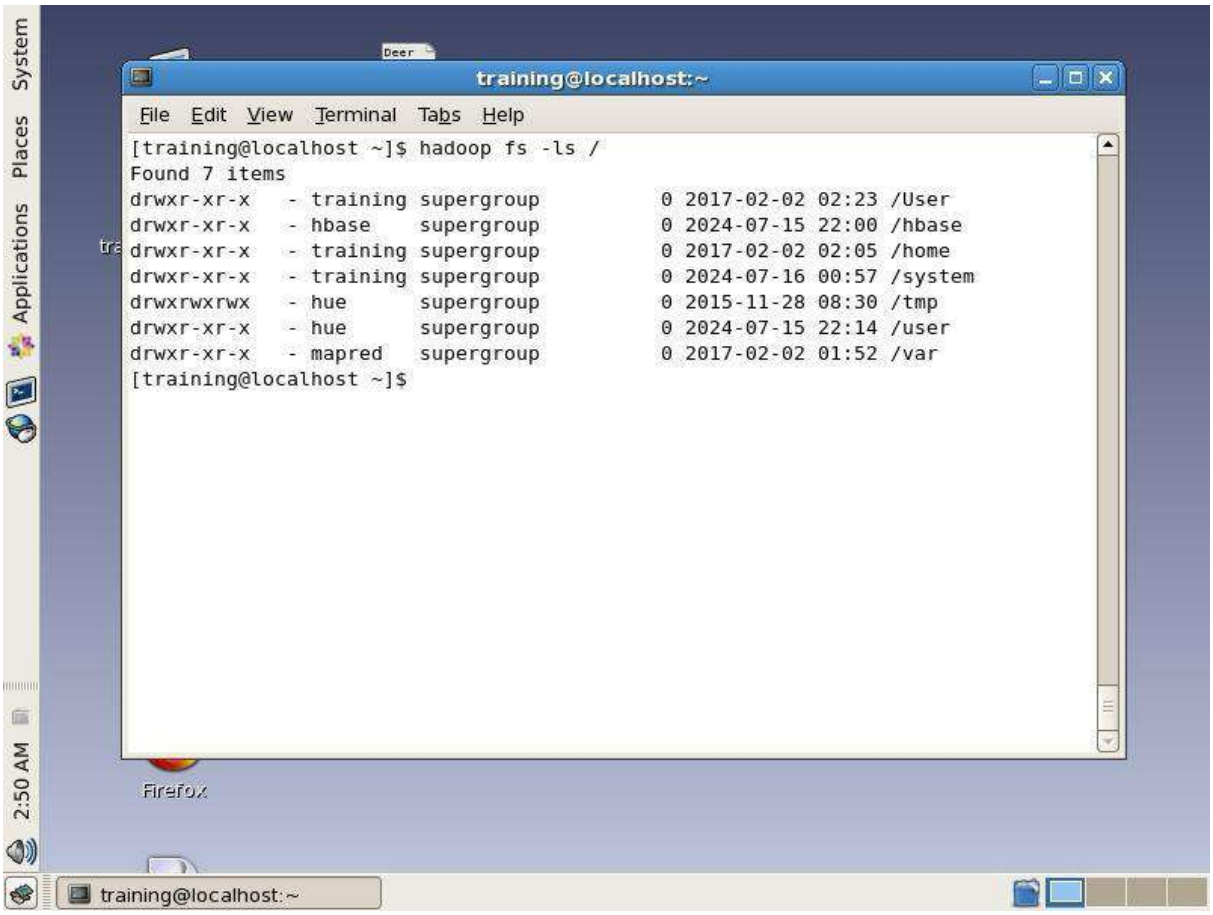

System
Places
Applications
2:58 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -mkdir /user/training/chirag_hadoop
[training@localhost ~]$
Firefox
training@localhost:~

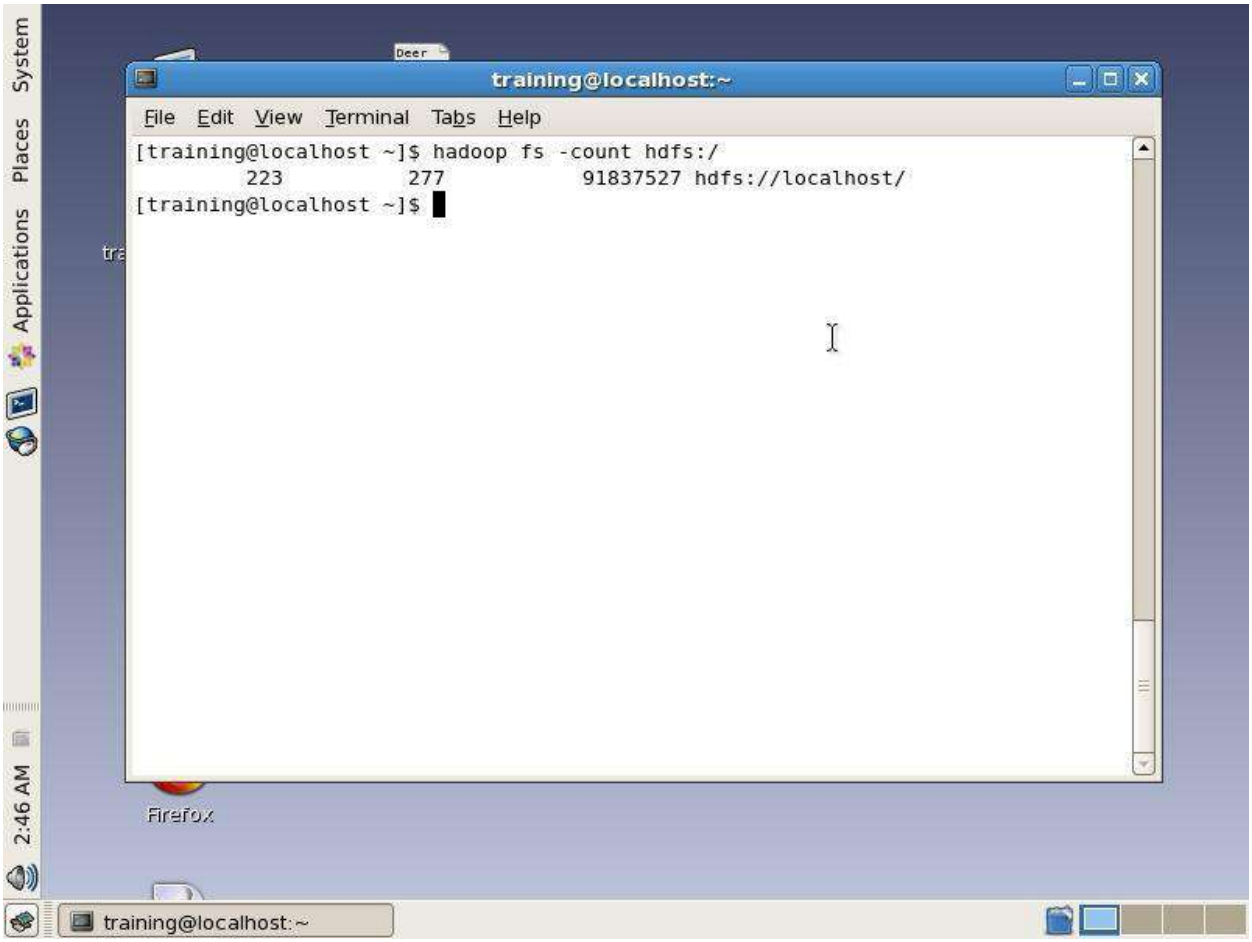

System
Places
Applications
3:08 AM
training@localhost:~
File Edit View Terminal Tabs Help
rdcount
[training@localhost ~]$ hadoop fs -ls /user/training
Found 48 items
-rw-r--r-- 1 training supergroup 1390 2015-10-02 20:20 /user/training/Bo
oks
-rw------- 2 root root 14 2024-07-16 01:18 /user/training/Ka
reenaC11_08
drwxr-xr-x - training supergroup 0 2024-07-16 00:58 /user/training/Ka
reenac11_08
drwxr-xr-x - training supergroup 0 2014-08-17 03:50 /user/training/We
atherData
drwxr-xr-x - training supergroup 0 2015-10-17 02:34 /user/training/_s
qoop
-rw-r--r-- 1 training supergroup 23 2015-11-29 23:36 /user/training/a
drwxr-xr-x - training supergroup 0 2017-02-15 21:58 /user/training/ap
ache_hadoop
-rw-r--r-- 1 training supergroup 2944 2015-09-26 02:37 /user/training/bo
okinfo
drwxr-xr-x - training supergroup 0 2024-07-16 02:58 /user/training/ch
irag_hadoop
drwxr-xr-x - training supergroup 0 2015-09-20 01:38 /user/training/cl
ass2009_dir1
-rw-r--r-- 1 training supergroup 81 2015-09-18 21:05 /user/training/de
ptinfo
drwxr-xr-x - training supergroup 0 2014-08-17 01:59 /user/training/di
r1
drwxr-xr-x - training supergroup 0 2015-09-12 02:42 /user/training/di
r10
drwxr-xr-x - training supergroup 0 2015-09-19 02:00 /user/training/di
r100
training@localhost:~

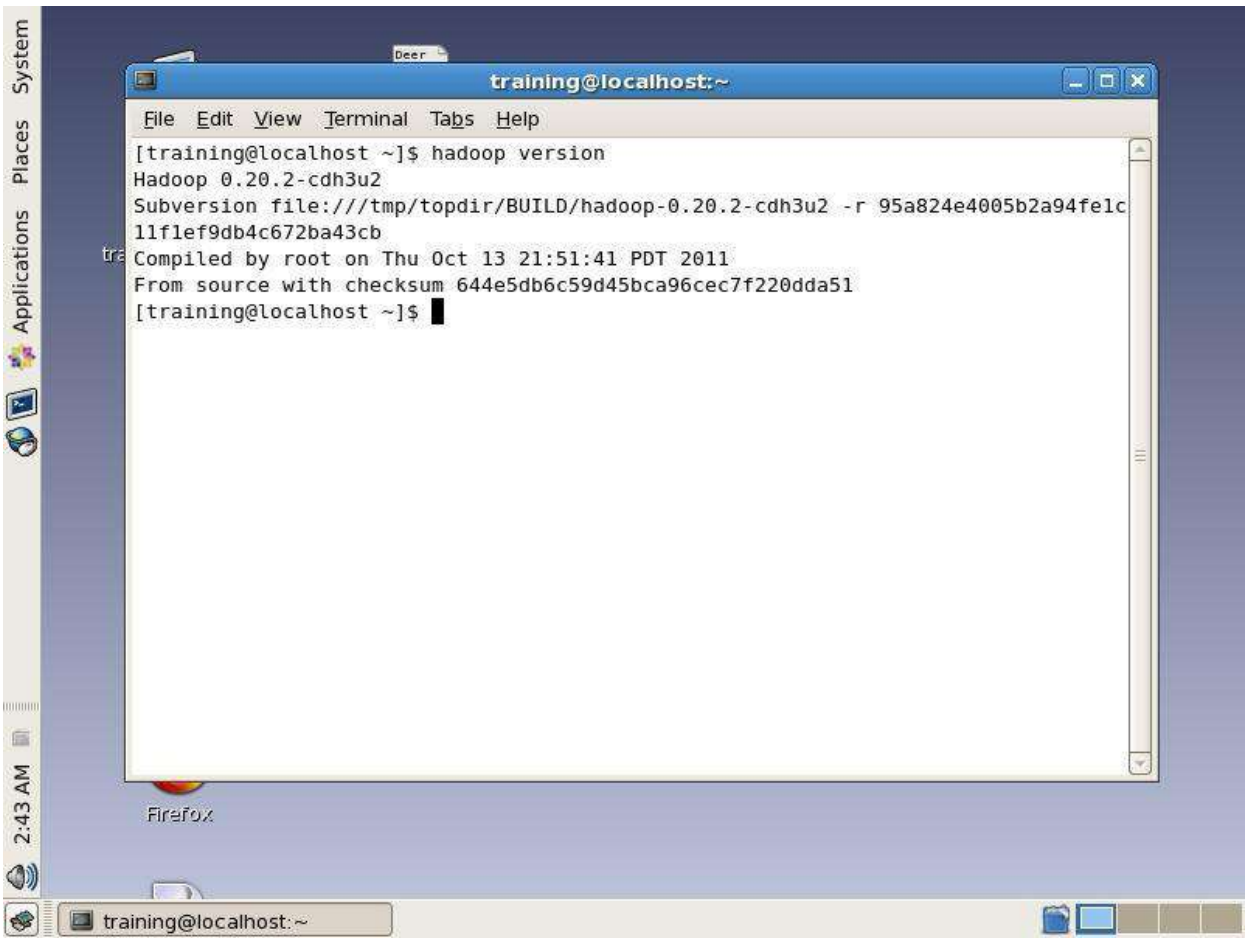

System
Places
Applications
2:52 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop balancer
Time Stamp Iteration# Bytes Already Moved Bytes Left To Move By
tes Being Moved
24/07/16 02:52:29 INFO net.NetworkTopology: Adding a new node: /default-rack/127
.0.0.1:50010
24/07/16 02:52:29 INFO balancer.Balancer: 0 over utilized nodes:
24/07/16 02:52:29 INFO balancer.Balancer: 1 under utilized nodes: 127.0.0.1:500
10
The cluster is balanced. Exiting...
Balancing took 345.0 milliseconds
[training@localhost ~]$
Firefox
training@localhost:~

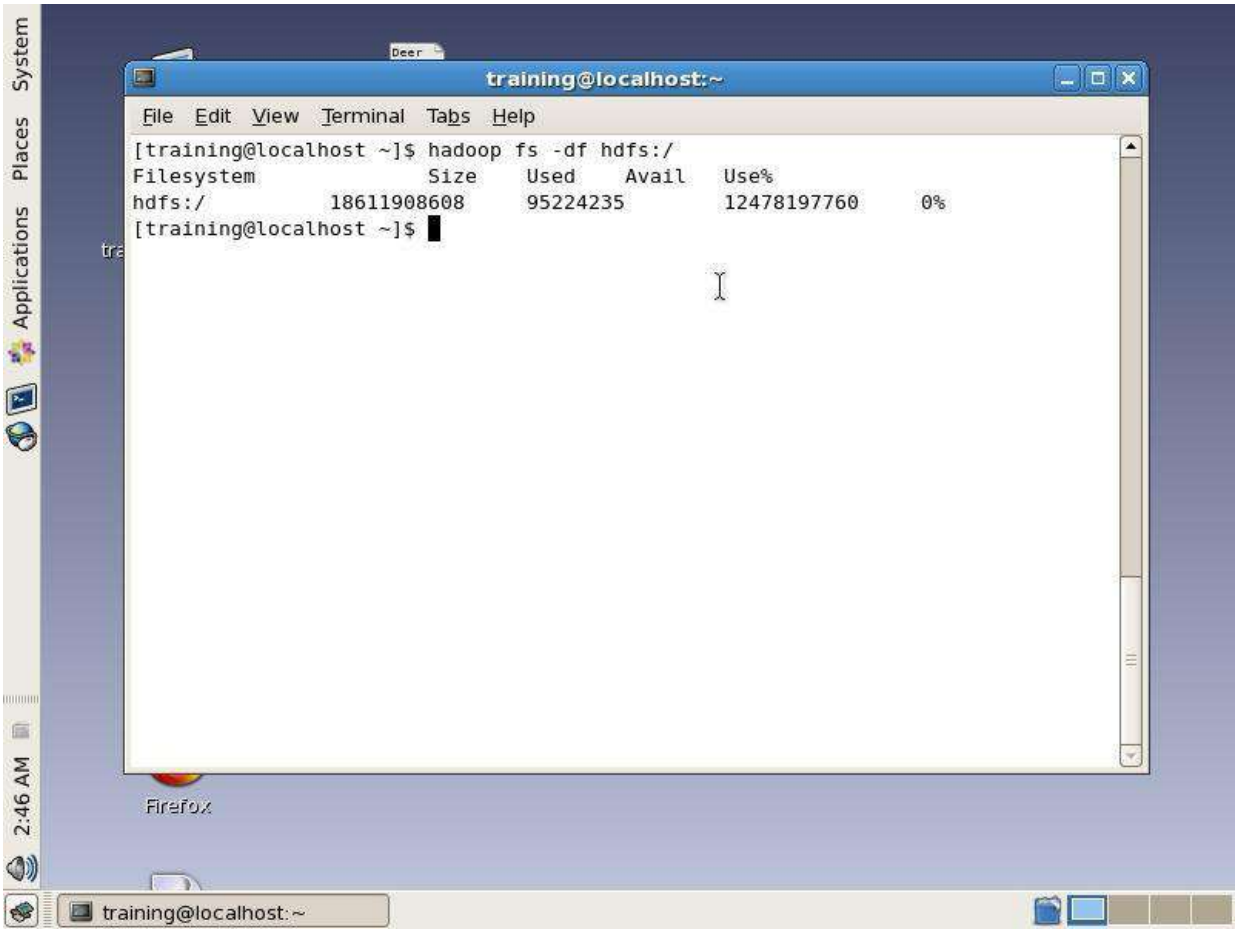

System
Places
Applications
2:51 AM
training@localhost:~
File Edit View Terminal Tabs Help
/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_201509190
216_0061/libjars/paranamer-2.3.jar: Under replicated blk_-8205088778216716594_2
223. Target Replicas is 10 but found 1 replica(s).
.
/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_201509190
216_0061/libjars/snappy-java-1.0.3.2.jar: Under replicated blk_3475955673016738
953_2229. Target Replicas is 10 but found 1 replica(s).
.
/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_201509190
216_0061/libjars/sqoop-1.3.0-cdh3u2.jar: Under replicated blk_43383766509579011
00_2222. Target Replicas is 10 but found 1 replica(s).
.
/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_201510310
312_0017/job.jar: Under replicated blk_-3668555532469776353_2503. Target Replic
as is 10 but found 1 replica(s).
.Status: HEALTHY
Total size: 91837527 B
Total dirs: 223
Total files: 276 (Files currently being written: 1)
Total blocks (validated): 259 (avg. block size 354585 B) (Total open file
blocks (not validated): 1)
Minimally replicated blocks: 259 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 99 (38.223938 %)
Firefox
training@localhost:~


System
Places
Applications
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fsck -/
FSCK started by training from /127.0.0.1 for path / at Tue Jul 16 02:47:16 PDT 2024
.....
/hbase/-ROOT-/70236052/.oldlogs/hlog.1309921036526: Under replicated blk_-7951519016048452162_1042. Target Replicas is 3 but found 1 replica(s).
.
/hbase/-ROOT-/70236052/.regioninfo: Under replicated blk_4111342483554438822_1040. Target Replicas is 3 but found 1 replica(s).
..
/hbase/.META./1028785192/.oldlogs/hlog.1309921036692: Under replicated blk_3938822511733381919_1043. Target Replicas is 3 but found 1 replica(s).
.
/hbase/.META./1028785192/.regioninfo: Under replicated blk_2455092764027271611_1042. Target Replicas is 3 but found 1 replica(s).
....
/hbase/hbase.version: Under replicated blk_1723987066918777078_1038. Target Replicas is 3 but found 1 replica(s).
.....
/user/training/KareenaC11_08: Under replicated blk_-6743868351850536579_2643. Target Replicas is 2 but found 1 replica(s).
...
/user/training/apache_hadoop/b.txt: Under replicated blk_-3399004998250828873_2574. Target Replicas is 2 but found 1 replica(s).
Firefox
2:49 AM
training@localhost:~


System
Places
Applications
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -ls /
Found 7 items
drwxr-xr-x - training supergroup 0 2017-02-02 02:23 /User
drwxr-xr-x - hbase supergroup 0 2024-07-15 22:00 /hbase
drwxr-xr-x - training supergroup 0 2017-02-02 02:05 /home
drwxr-xr-x - training supergroup 0 2024-07-16 00:57 /system
drwxrwxrwx - hue supergroup 0 2015-11-28 08:30 /tmp
drwxr-xr-x - hue supergroup 0 2024-07-15 22:14 /user
drwxr-xr-x - mapred supergroup 0 2017-02-02 01:52 /var
[training@localhost ~]$
Firefox
2:50 AM
training@localhost:~


System
Places
Applications
2:46 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -count hdfs:/
223 277 91837527 hdfs://localhost/
[training@localhost ~]$
Firefox
training@localhost:~


System
Places
Applications
2:43 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop version
Hadoop 0.20.2-cdh3u2
Subversion file:///tmp/topdir/BUILD/hadoop-0.20.2-cdh3u2 -r 95a824e4005b2a94fe1c
11f1ef9db4c672ba43cb
Compiled by root on Thu Oct 13 21:51:41 PDT 2011
From source with checksum 644e5db6c59d45bca96cec7f220dda51
[training@localhost ~]$
Firefox
training@localhost:~


System
Places
Applications
2:46 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -df hdfs:/
Filesystem Size Used Avail Use%
hdfs:/ 18611908608 95224235 12478197760 0%
[training@localhost ~]$
Firefox
training@localhost:~

Experiment 03

Aim: Programming exercises in HBASE

Theory:

HBase is a column-oriented non-relational database management system that runs on top of the Hadoop Distributed File System (HDFS). HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases. It is well suited for real-time data processing or random read/write access to large volumes of Data.

Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java™ much like a typical Apache MapReduce application. HBase does support writing applications in Apache Avro, REST, and Thrift.
An HBase system is designed to scale linearly. It comprises a set of standard tables with rows and columns, much like a traditional database. Each table must have an element defined as a primary key, and all access attempts to HBase tables must use this primary key.

Avro, as a component, supports a rich set of primitive data types including numeric, binary data, and strings; and a number of complex types including arrays, maps, enumerations, and records. A sort order can also be defined for the data.

HBase relies on ZooKeeper for high-performance coordination. ZooKeeper is built into HBase, but if you're running a production cluster, it's suggested that you have a dedicated ZooKeeper cluster that's integrated with your HBase cluster.
HBase works well with Hive, a query engine for batch processing of big data, to enable fault-tolerant big data applications.

Hbase shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.
The master server manages these region servers and all these tasks take place on HDFS

```
[cloudera@quickstart ~]$ hbase shell
2022-08-26 23:52:01,107 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.nati
ve.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017
```

Create Table

You can create a table using the create command, here you must specify the table name

and the Column Family name. The syntax to create a table in HBase shell is shown below.

create ‘<table name>’,’<column family>’

```
hbase(main):001:0> create 'register','accno','name','password','email','age'
0 row(s) in 2.7300 seconds

=> Hbase::Table - register
hbase(main):002:0> list
TABLE
register
1 row(s) in 0.0290 seconds

=> ["register"]
hbase(main):003:0>
```

Listing a Table

Liist is the command that is used to list all the tables in HBase. Given below is the syntax

of the list command.

hbase(main):001:0 > list

```
hbase(main):001:0> create 'register','accno','name','password','email','age'
0 row(s) in 2.7300 seconds

=> Hbase::Table - register
hbase(main):002:0> list
TABLE
register
1 row(s) in 0.0290 seconds

=> ["register"]
hbase(main):003:0>
```

Disabling a Table

To delete a table or change its settings, you need to first disable the table using the disable command. After disabling the table, you can still sense its existence through list and exists commands.Given below is the syntax to disable a table:

disable '<table name>'

```
hbase(main):003:0> disable 'register'
0 row(s) in 2.4400 seconds

hbase(main):004:0> is_disabled 'register'
true
0 row(s) in 0.0310 seconds

hbase(main):005:0> enable 'register'
0 row(s) in 1.3020 seconds

hbase(main):006:0>
```

cloudera

STEP 5

Enabling a Table

Syntax to enable a table:

enable '<table name>'

```
hbase(main):003:0> disable 'register'
0 row(s) in 2.4400 seconds

hbase(main):004:0> is_disabled 'register'
true
0 row(s) in 0.0310 seconds

hbase(main):005:0> enable 'register'
0 row(s) in 1.3020 seconds

hbase(main):006:0>
```

cloudera

STEP 6

Describe Table

This command returns the description of the table. Its syntax is as follows:

hbase> describe 'table name'

Given below is the output of the described command on the register table:

```
hbase(main):007:0> describe 'register'
Table register is ENABLED
register
COLUMN FAMILIES DESCRIPTION
{NAME => 'accno', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_ME
MORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'age', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPR
ESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMO
RY => 'false', BLOCKCACHE => 'true'}
{NAME => 'email', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_ME
MORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'name', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMP
RESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEM
ORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'password', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1',
COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN
_MEMORY => 'false', BLOCKCACHE => 'true'}
5 row(s) in 0.0640 seconds

hbase(main):008:0>
```

STEP 7

Alter Command

Alter is the command used to make changes to an existing table. Using this command, you can change the maximum number of cells of a column family, set and delete table scope operators, and delete a column family from a table.

★ Changing the Maximum Number of Cells of a Column Family

Given below is the syntax to change the maximum number of cells of a column family.

hbase> alter 't1', NAME ⇒ 'f1', VERSIONS ⇒ 5

```
hbase(main):008:0> alter 'register',NAME => 'name',VERSION => 5
Unknown argument ignored for column family name: 1.8.7
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.0020 seconds
```

STEP 8

Exist Command

You can verify the existence of a table using the exists command. The following example shows how to use this command.

```
hbase(main):010:0> exists 'register'
Table register does exist
0 row(s) in 0.0350 seconds

hbase(main):011:0>
```

STEP 9

Drop Table

Using the drop command, you can delete a table. But before dropping a table, you have to disable it.

```
hbase(main):011:0> drop 'register'

ERROR: Table register is enabled. Disable it first.

Drop the named table. Table must first be disabled:
  hbase> drop 't1'
  hbase> drop 'ns1:t1'


hbase(main):012:0> disable 'register'
0 row(s) in 2.2970 seconds

hbase(main):013:0> drop 'register'
0 row(s) in 1.3030 seconds

hbase(main):014:0>
```

STEP 10

Creating Data (Put command)

Using the put command, you can insert rows into a table. Its syntax is as follows:

put ’<table name>’,’row1’,’<colfamily:colname>’,’<value>’

```
hbase(main):017:0> create 'register','personal data','account data'
0 row(s) in 1.2450 seconds

=> Hbase::Table - register
hbase(main):018:0> put 'register','1','personal data:name','raj'
0 row(s) in 0.1220 seconds

hbase(main):019:0> put 'register','1','personal data:age','11'
0 row(s) in 0.0140 seconds

hbase(main):020:0> put 'register','1','personal data:email','raj@gmail.com'
0 row(s) in 0.0150 seconds

hbase(main):021:0> put 'register','1','account data:accno','1'
0 row(s) in 0.0080 seconds

hbase(main):022:0>
```

STEP 11

Scan Table

The scan command is used to view the data in HTable. Using the scan command, you

can get the table data. Its syntax is as follows:

scan ‘<table name>’

```
hbase(main):022:0> scan 'register'
ROW                           COLUMN+CELL
 1                            column=account data:accno, timestamp=1661584106330, value=1
 1                            column=personal data:age, timestamp=1661584028285, value=11
 1                            column=personal data:email, timestamp=1661584066229, value=raj@gmail.com
 1                            column=personal data:name, timestamp=1661584013135, value=raj
1 row(s) in 0.0410 seconds

hbase(main):023:0>
```

STEP 12

Updating Data (Put Command)

You can update an existing cell value using the put command. The newly given value replaces the existing value, updating the row. To do so, just follow the same syntax and mention your new value as shown below.

put ‘table name’,’row ’,'Column family:column name',’new value’

```
hbase(main):023:0> put 'register','1','personal data:age','18'
0 row(s) in 0.0120 seconds

hbase(main):024:0> scan 'register'
ROW                           COLUMN+CELL
 1                            column=account data:accno, timestamp=1661584106330, value=1
 1                            column=personal data:age, timestamp=1661584211091, value=18
 1                            column=personal data:email, timestamp=1661584066229, value=raj@gmail.com
 1                            column=personal data:name, timestamp=1661584013135, value=raj
1 row(s) in 0.0180 seconds

hbase(main):025:0>
```

STEP 13

Read Data (Get Command)

The get command and the get() method of HTable class are used to read data from a table in HBase. Using the get command, you can get a single row of data at a time. Its syntax is as follows:

get ’<table name>’,’row1’

```
hbase(main):025:0> get 'register','1'
COLUMN                          CELL
 account data:accno             timestamp=1661584106330, value=1
 personal data:age              timestamp=1661584211091, value=18
 personal data:email            timestamp=1661584066229, value=raj@gmail.com
 personal data:name             timestamp=1661584013135, value=raj
4 row(s) in 0.0290 seconds

hbase(main):026:0> count 'register'
1 row(s) in 0.0280 seconds
```

STEP 14

Delete Data (Delete Command)

Deleting a Specific Cell in a Table

Using the delete command, you can delete a specific cell in a table. The syntax of

delete command is as follows:

delete ‘<table name>’, ‘<row>’, ‘<column name >’, ‘<time stamp>’

```
hbase(main):028:0> delete 'register','1','personal data:name',1661584013135
0 row(s) in 0.0360 seconds

hbase(main):029:0>
```

STEP 15

Count Command

You can count the number of rows of a table using the count command. Its syntax is as

follows:

count ‘<table name>’

```
hbase(main):025:0> get 'register','1'
COLUMN                          CELL
 account data:accno             timestamp=1661584106330, value=1
 personal data:age              timestamp=1661584211091, value=18
 personal data:email            timestamp=1661584066229, value=raj@gmail.com
 personal data:name             timestamp=1661584013135, value=raj
4 row(s) in 0.0290 seconds

hbase(main):026:0> count 'register'
1 row(s) in 0.0280 seconds
```

STEP 16

Truncate Command

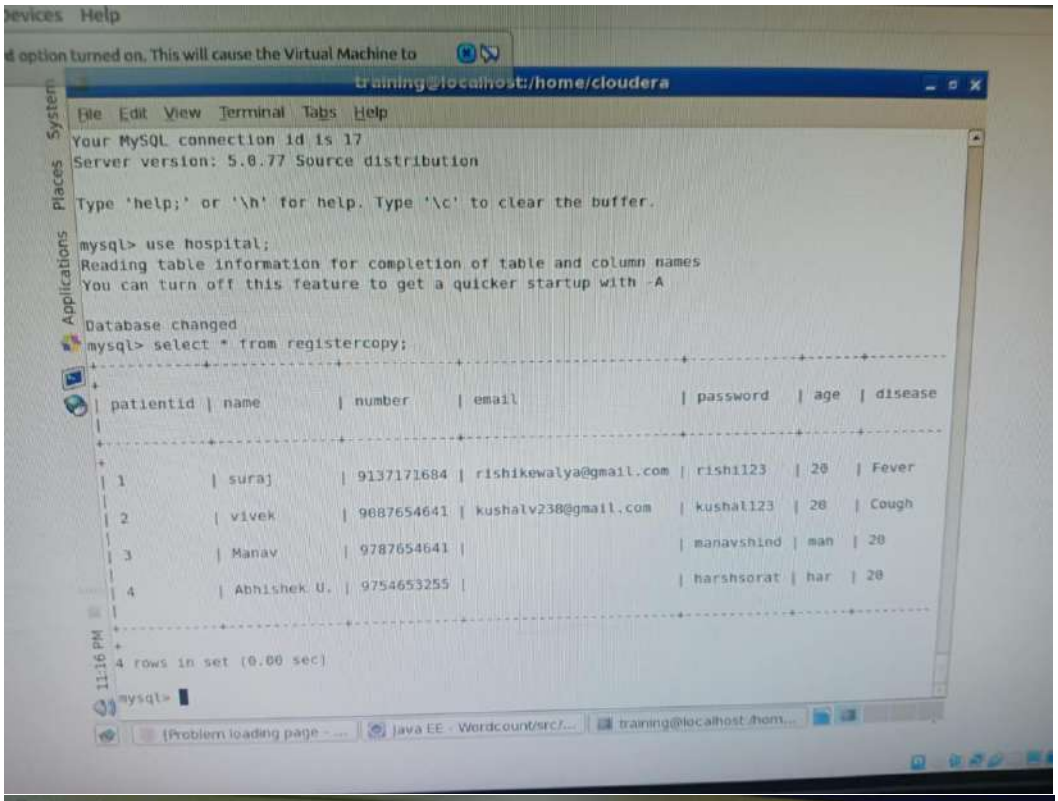
This command disables drops and recreates a table. The syntax of truncate is as

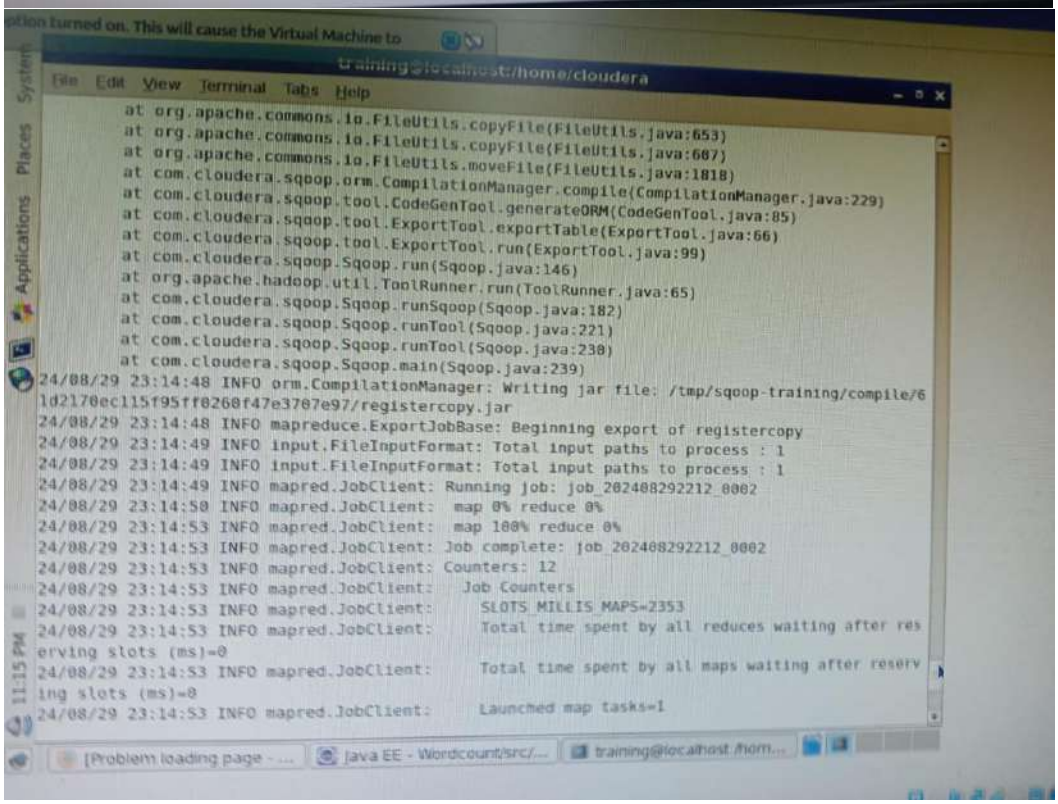
follows:

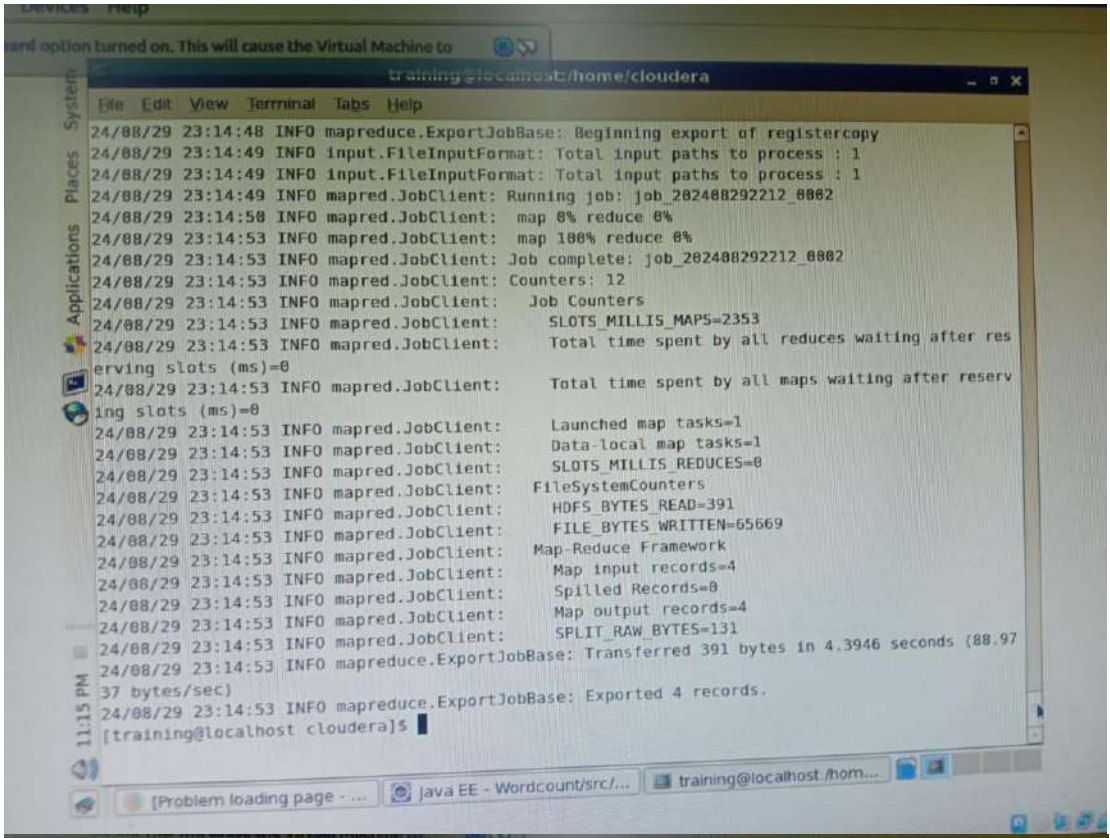
hbase> truncate 'table name

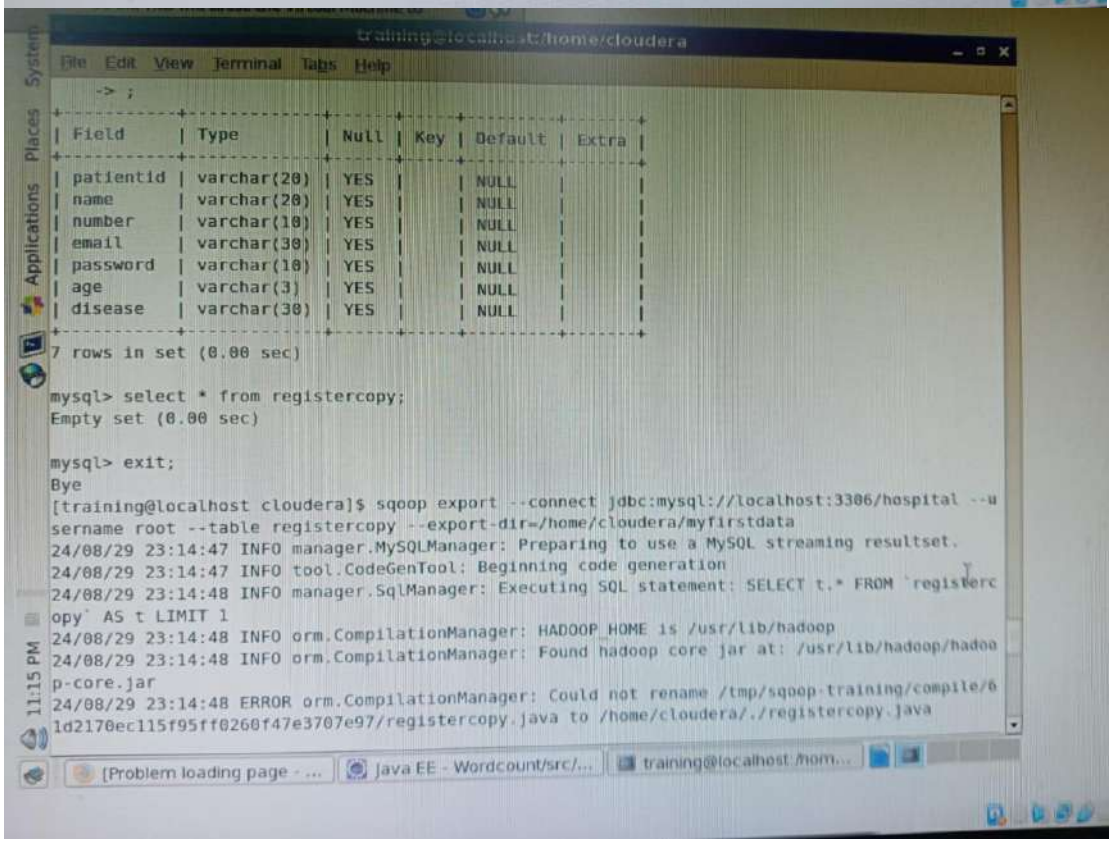
```
hbase(main):030:0> truncate 'register'
Truncating 'register' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.7130 seconds

hbase(main):031:0>
```

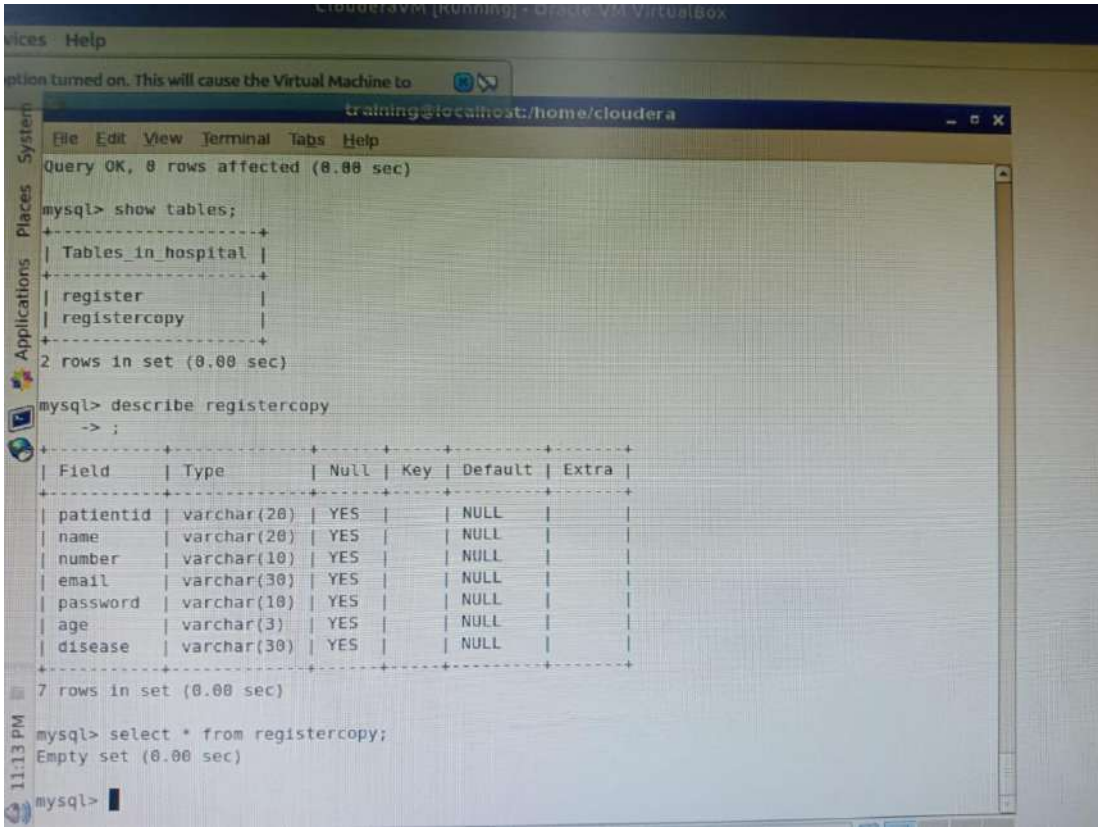
Output:

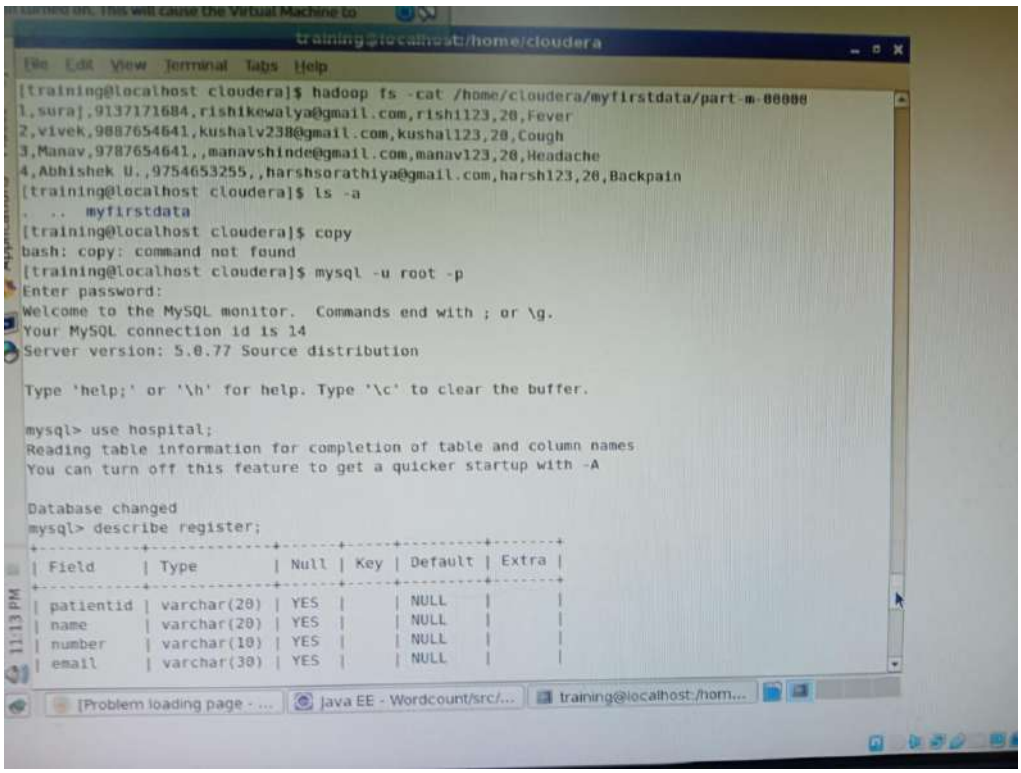

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
Your MySQL connection id is 17
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use hospital;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> select * from registercopy;
patientid	name	number	email	password	age	disease
1	suraj	9137171684	rishikewalya@gmail.com	rishi123	20	Fever
2	vivek	9087654641	kushalv238@gmail.com	kushal123	20	Cough
3	Manav	9787654641		manavshind	man	20
4	Abhishek U.	9754653255		harshsorat	har	20
4 rows in set (0.00 sec)
mysql>

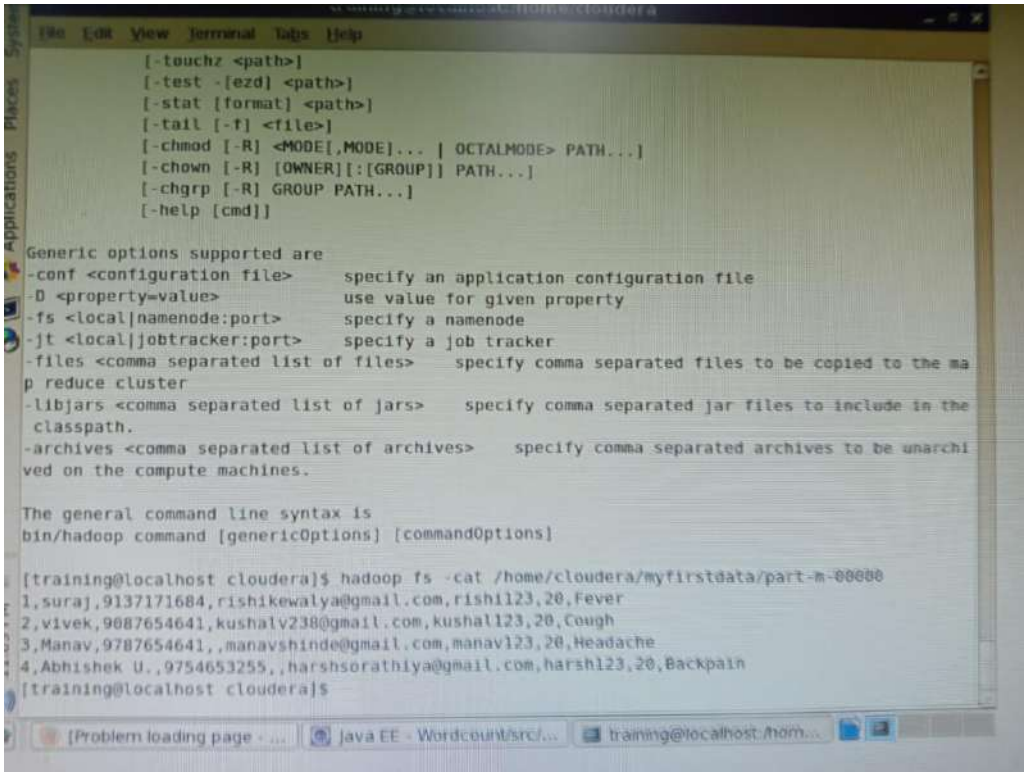

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
at org.apache.commons.io.FileUtils.copyFile(FileUtils.java:653)
at org.apache.commons.io.FileUtils.copyFile(FileUtils.java:607)
at org.apache.commons.io.FileUtils.moveFile(FileUtils.java:1818)
at com.cloudera.sqoop.orm.CompilationManager.compile(CompilationManager.java:229)
at com.cloudera.sqoop.tool.CodeGenTool.generateORM(CodeGenTool.java:85)
at com.cloudera.sqoop.tool.ExportTool.exportTable(ExportTool.java:66)
at com.cloudera.sqoop.tool.ExportTool.run(ExportTool.java:99)
at com.cloudera.sqoop.Sqoop.run(Sqoop.java:146)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:65)
at com.cloudera.sqoop.Sqoop.runSqoop(Sqoop.java:182)
at com.cloudera.sqoop.Sqoop.runTool(Sqoop.java:221)
at com.cloudera.sqoop.Sqoop.runTool(Sqoop.java:230)
at com.cloudera.sqoop.Sqoop.main(Sqoop.java:239)
24/08/29 23:14:48 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/61d2170ec115f95ff0260f47e3707e97/registercopy.jar
24/08/29 23:14:48 INFO mapreduce.ExportJobBase: Beginning export of registercopy
24/08/29 23:14:49 INFO input.FileInputFormat: Total input paths to process : 1
24/08/29 23:14:49 INFO input.FileInputFormat: Total input paths to process : 1
24/08/29 23:14:49 INFO mapred.JobClient: Running job: job_202408292212_0002
24/08/29 23:14:50 INFO mapred.JobClient: map 0% reduce 0%
24/08/29 23:14:53 INFO mapred.JobClient: map 100% reduce 0%
24/08/29 23:14:53 INFO mapred.JobClient: Job complete: job_202408292212_0002
24/08/29 23:14:53 INFO mapred.JobClient: Counters: 12
24/08/29 23:14:53 INFO mapred.JobClient: Job Counters
24/08/29 23:14:53 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=2353
24/08/29 23:14:53 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
24/08/29 23:14:53 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
24/08/29 23:14:53 INFO mapred.JobClient: Launched map tasks=1

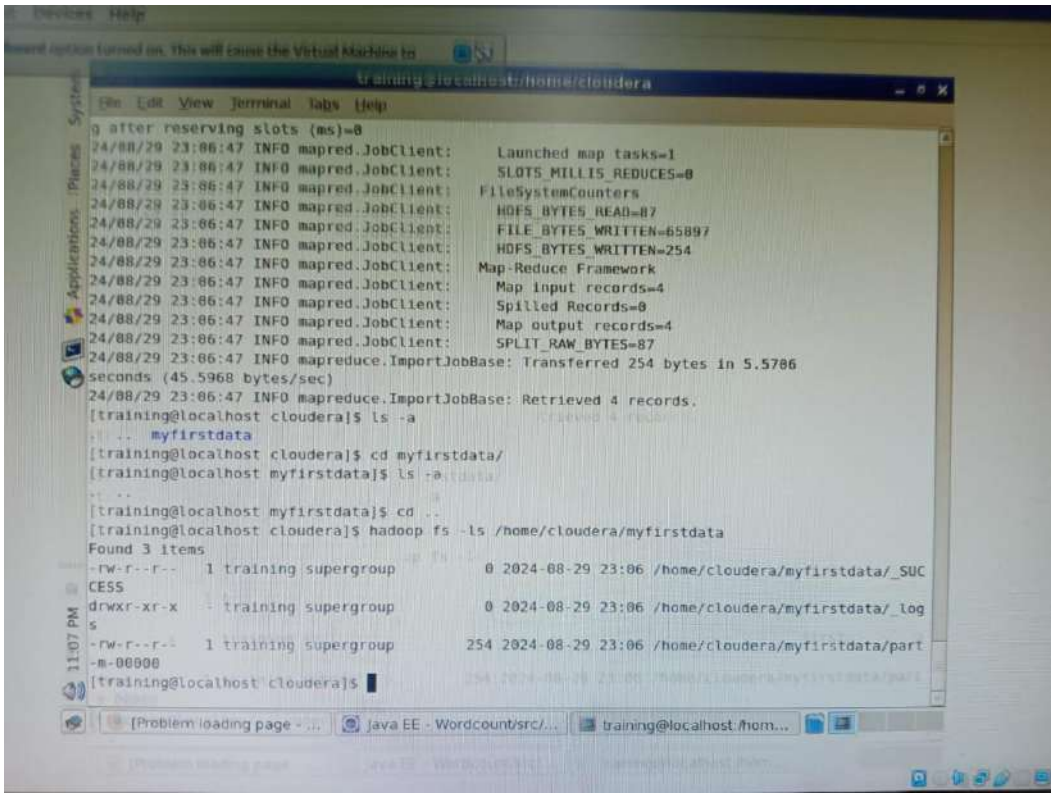

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
24/08/29 23:14:48 INFO mapreduce.ExportJobBase: Beginning export of registercopy
24/08/29 23:14:49 INFO input.FileInputFormat: Total input paths to process : 1
24/08/29 23:14:49 INFO input.FileInputFormat: Total input paths to process : 1
24/08/29 23:14:49 INFO mapred.JobClient: Running job: job_202408292212_0002
24/08/29 23:14:50 INFO mapred.JobClient: map 0% reduce 0%
24/08/29 23:14:53 INFO mapred.JobClient: map 100% reduce 0%
24/08/29 23:14:53 INFO mapred.JobClient: Job complete: job_202408292212_0002
24/08/29 23:14:53 INFO mapred.JobClient: Counters: 12
24/08/29 23:14:53 INFO mapred.JobClient: Job Counters
24/08/29 23:14:53 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=2353
24/08/29 23:14:53 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
24/08/29 23:14:53 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
24/08/29 23:14:53 INFO mapred.JobClient: Launched map tasks=1
24/08/29 23:14:53 INFO mapred.JobClient: Data-local map tasks=1
24/08/29 23:14:53 INFO mapred.JobClient: SLOTS_MILLIS_REDUCES=0
24/08/29 23:14:53 INFO mapred.JobClient: FileSystemCounters
24/08/29 23:14:53 INFO mapred.JobClient: HDFS_BYTES_READ=391
24/08/29 23:14:53 INFO mapred.JobClient: FILE_BYTES_WRITTEN=65669
24/08/29 23:14:53 INFO mapred.JobClient: Map-Reduce Framework
24/08/29 23:14:53 INFO mapred.JobClient: Map input records=4
24/08/29 23:14:53 INFO mapred.JobClient: Spilled Records=0
24/08/29 23:14:53 INFO mapred.JobClient: Map output records=4
24/08/29 23:14:53 INFO mapred.JobClient: SPLIT_RAW_BYTES=131
24/08/29 23:14:53 INFO mapreduce.ExportJobBase: Transferred 391 bytes in 4.3946 seconds (88.97 bytes/sec)
24/08/29 23:14:53 INFO mapreduce.ExportJobBase: Exported 4 records.
[training@localhost cloudera]$
11:15 PM

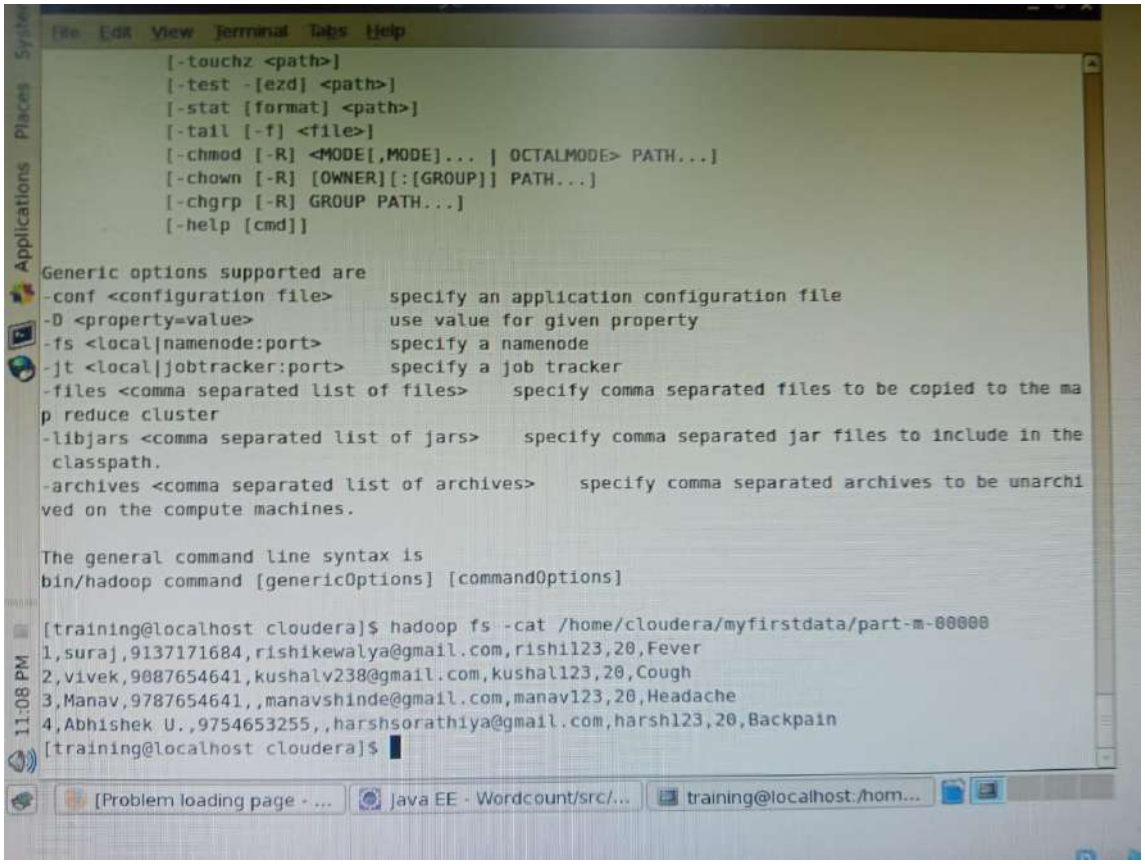

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
-> ;
+-----------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
patientid	varchar(20)	YES		NULL	
name	varchar(20)	YES		NULL	
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
password	varchar(10)	YES		NULL	
age	varchar(3)	YES		NULL	
disease	varchar(30)	YES		NULL	
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
mysql> select * from registercopy;
Empty set (0.00 sec)
mysql> exit;
Bye
[training@localhost cloudera]$ sqoop export --connect jdbc:mysql://localhost:3306/hospital --username root --table registercopy --export-dir=/home/cloudera/myfirstdata
24/08/29 23:14:47 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
24/08/29 23:14:47 INFO tool.CodeGenTool: Beginning code generation
24/08/29 23:14:48 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `registercopy` AS t LIMIT 1
24/08/29 23:14:48 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
24/08/29 23:14:48 INFO orm.CompilationManager: Found hadoop core jar at: /usr/lib/hadoop/hadoop-core.jar
24/08/29 23:14:48 ERROR orm.CompilationManager: Could not rename /tmp/sqoop-training/compile/61d2170ec115f95ff0260f47e3707e97/registercopy.java to /home/cloudera/./registercopy.java
11:15 PM

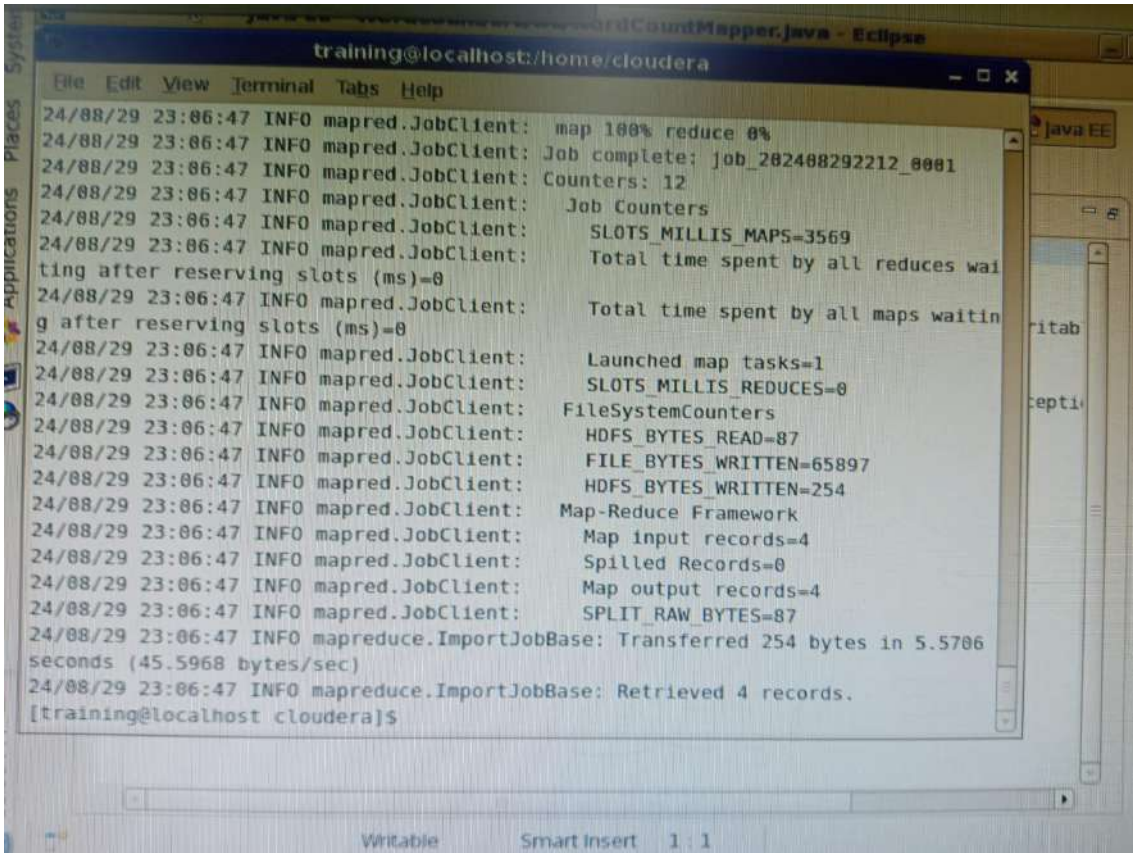

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
Query OK, 0 rows affected (0.00 sec)
mysql> show tables;
| Tables_in_hospital |
| register |
| registercopy |
2 rows in set (0.00 sec)
mysql> describe registercopy
-> ;
Field	Type	Null	Key	Default	Extra
patientid	varchar(20)	YES		NULL	
name	varchar(20)	YES		NULL	
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
password	varchar(10)	YES		NULL	
age	varchar(3)	YES		NULL	
disease	varchar(30)	YES		NULL	
7 rows in set (0.00 sec)
mysql> select * from registercopy;
Empty set (0.00 sec)
mysql>

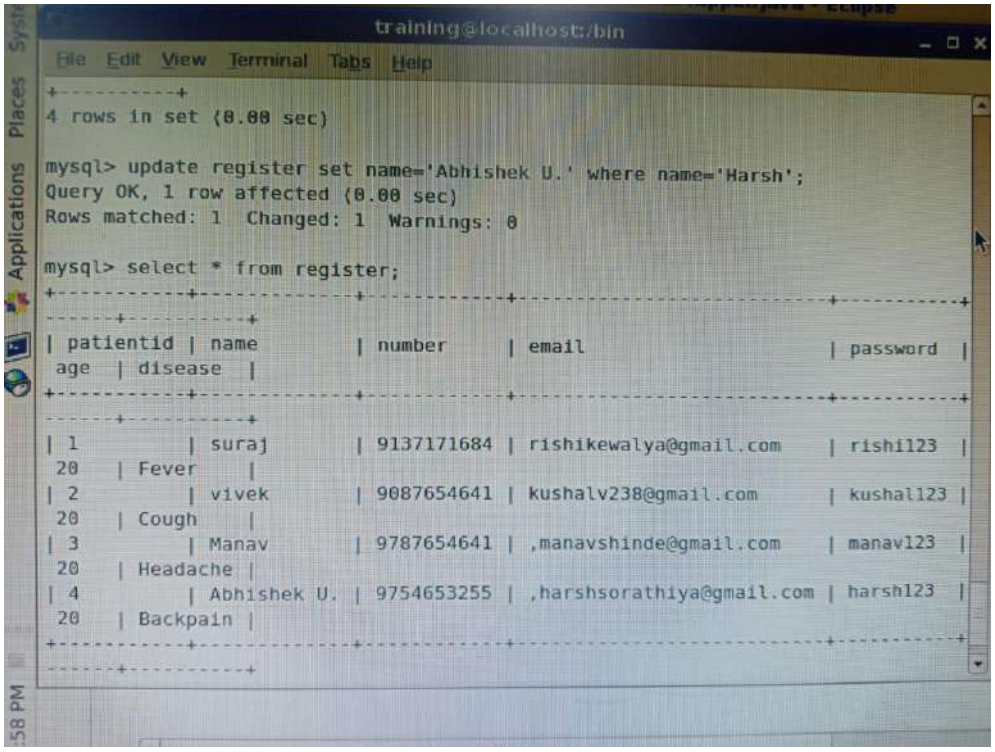

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
[training@localhost cloudera]$ hadoop fs -cat /home/cloudera/myfirstdata/part-m-00000
1,sura],9137171684,rishikewalya@gmail.com,rish123,20,Fever
2,vivek,9887654641,kushalv238@gmail.com,kushal123,20,Cough
3,Manav,9787654641,,manavshinde@gmail.com,manav123,20,Headache
4,Abhishek U.,9754653255,,harshsorathiya@gmail.com,harsh123,20,Backpain
[training@localhost cloudera]$ ls -a
. .. myfirstdata
[training@localhost cloudera]$ copy
bash: copy: command not found
[training@localhost cloudera]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use hospital;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> describe register;
Field	Type	Null	Key	Default	Extra
patientid	varchar(20)	YES		NULL	
name	varchar(20)	YES		NULL	
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
[Problem loading page - ...
Java EE - Wordcount/src/...
training@localhost:/hom...

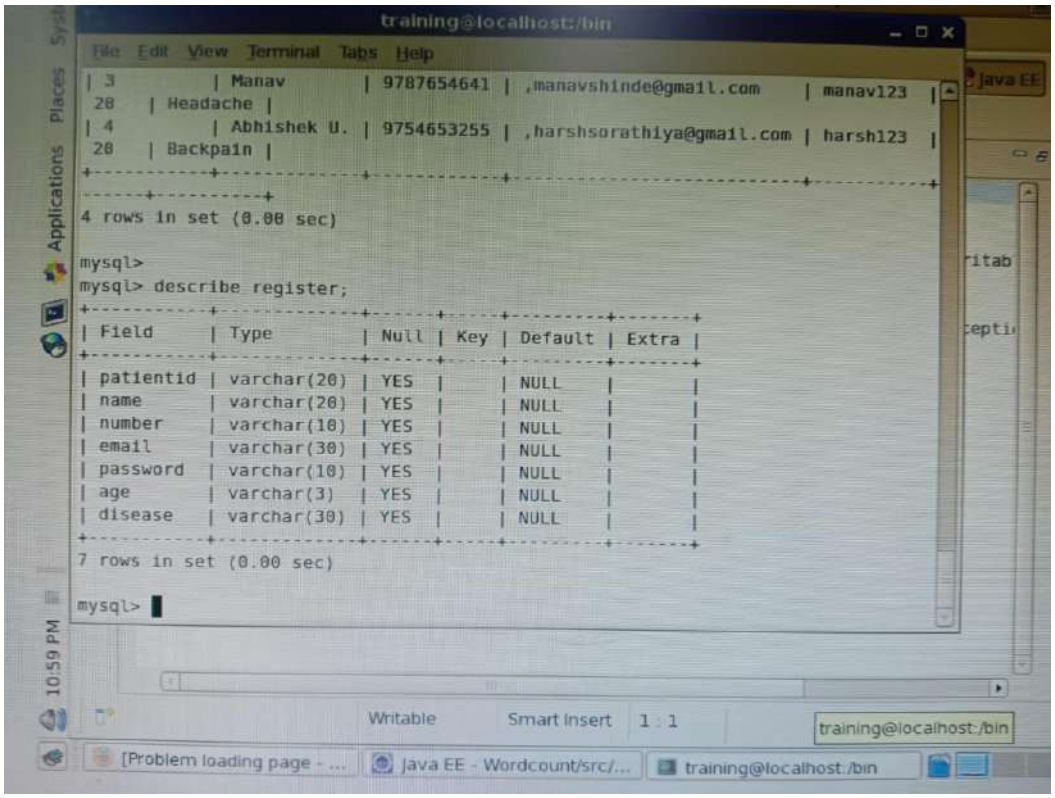

File Edit View Terminal Tabs Help
[-touchz <path>]
[-test -[ezd] <path>]
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
Generic options supported are
-conf <configuration file> specify an application configuration file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated files to be copied to the ma
p reduce cluster
-libjars <comma separated list of jars> specify comma separated jar files to include in the
classpath.
-archives <comma separated list of archives> specify comma separated archives to be unarchi
ved on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost cloudera]$ hadoop fs -cat /home/cloudera/myfirstdata/part-m-00000
1,suraj,9137171684,rishikewalya@gmail.com,rish123,20,Fever
2,vivek,9087654641,kushalv238@gmail.com,kushal123,20,Cough
3,Manav,9787654641,,manavshinde@gmail.com,manav123,20,Headache
4,Abhishek U.,9754653255,,harshsorathiya@gmail.com,harsh123,20,Backpain
[training@localhost cloudera]$
[Problem loading page - ...
Java EE - Wordcount/src/...
training@localhost:/hom...

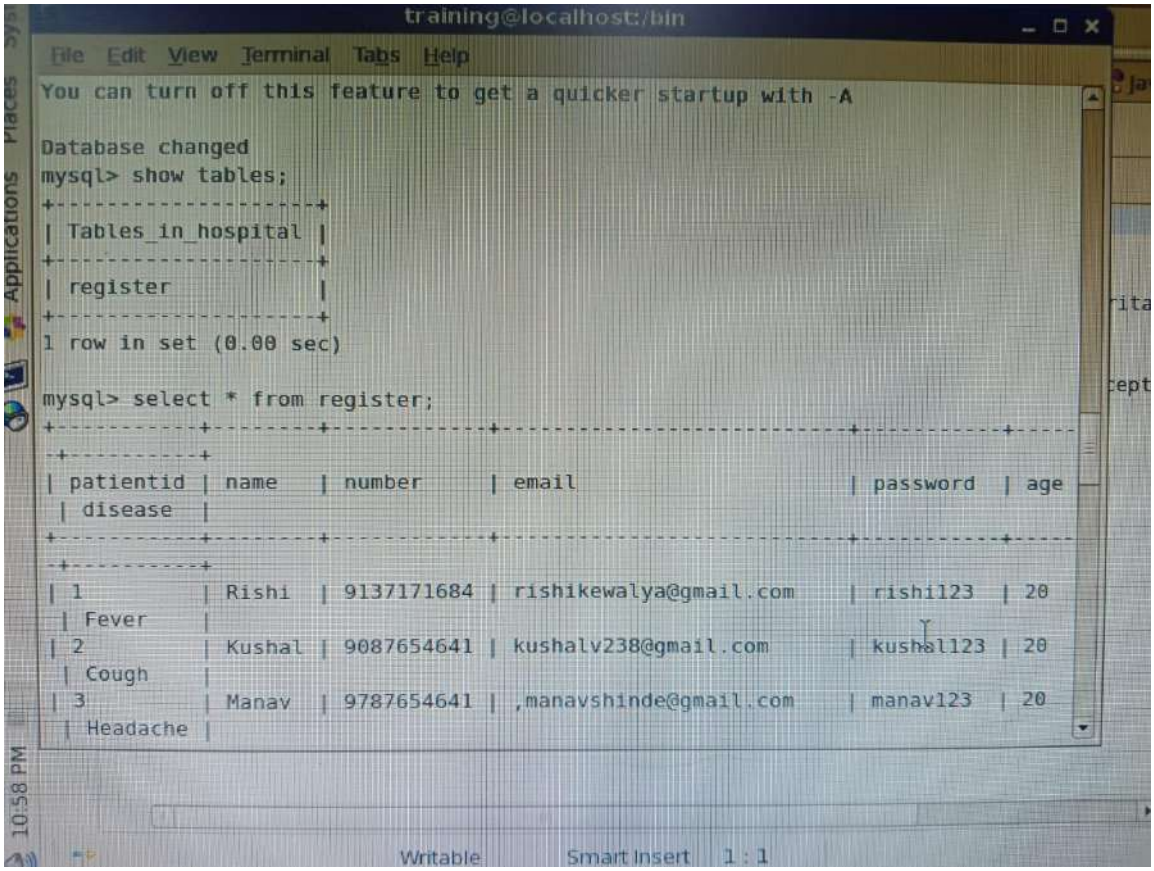

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
g after reserving slots (ms)=0
24/08/29 23:06:47 INFO mapred.JobClient: Launched map tasks=1
24/08/29 23:06:47 INFO mapred.JobClient: SLOTS_MILLIS_REDUCES=0
24/08/29 23:06:47 INFO mapred.JobClient: FileSystemCounters
24/08/29 23:06:47 INFO mapred.JobClient: HDFS_BYTES_READ=87
24/08/29 23:06:47 INFO mapred.JobClient: FILE_BYTES_WRITTEN=65897
24/08/29 23:06:47 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=254
24/08/29 23:06:47 INFO mapred.JobClient: Map-Reduce Framework
24/08/29 23:06:47 INFO mapred.JobClient: Map input records=4
24/08/29 23:06:47 INFO mapred.JobClient: Spilled Records=0
24/08/29 23:06:47 INFO mapred.JobClient: Map output records=4
24/08/29 23:06:47 INFO mapred.JobClient: SPLIT_RAW_BYTES=87
24/08/29 23:06:47 INFO mapreduce.ImportJobBase: Transferred 254 bytes in 5.5786
seconds (45.5968 bytes/sec)
24/08/29 23:06:47 INFO mapreduce.ImportJobBase: Retrieved 4 records.
[training@localhost cloudera]$ ls -a
.. myfirstdata
[training@localhost cloudera]$ cd myfirstdata/
[training@localhost myfirstdata]$ ls -a
.. ..
[training@localhost myfirstdata]$ cd ..
[training@localhost cloudera]$ hadoop fs -ls /home/cloudera/myfirstdata
Found 3 items
-rw-r--r-- 1 training supergroup 0 2024-08-29 23:06 /home/cloudera/myfirstdata/_SUC
CESS
drwxr-xr-x - training supergroup 0 2024-08-29 23:06 /home/cloudera/myfirstdata/_log
s
-rw-r--r-- 1 training supergroup 254 2024-08-29 23:06 /home/cloudera/myfirstdata/part
-m-00000
[training@localhost cloudera]$
[Problem loading page - ...
Java EE - Wordcount/src/...
training@localhost:/hom...
11:07 PM

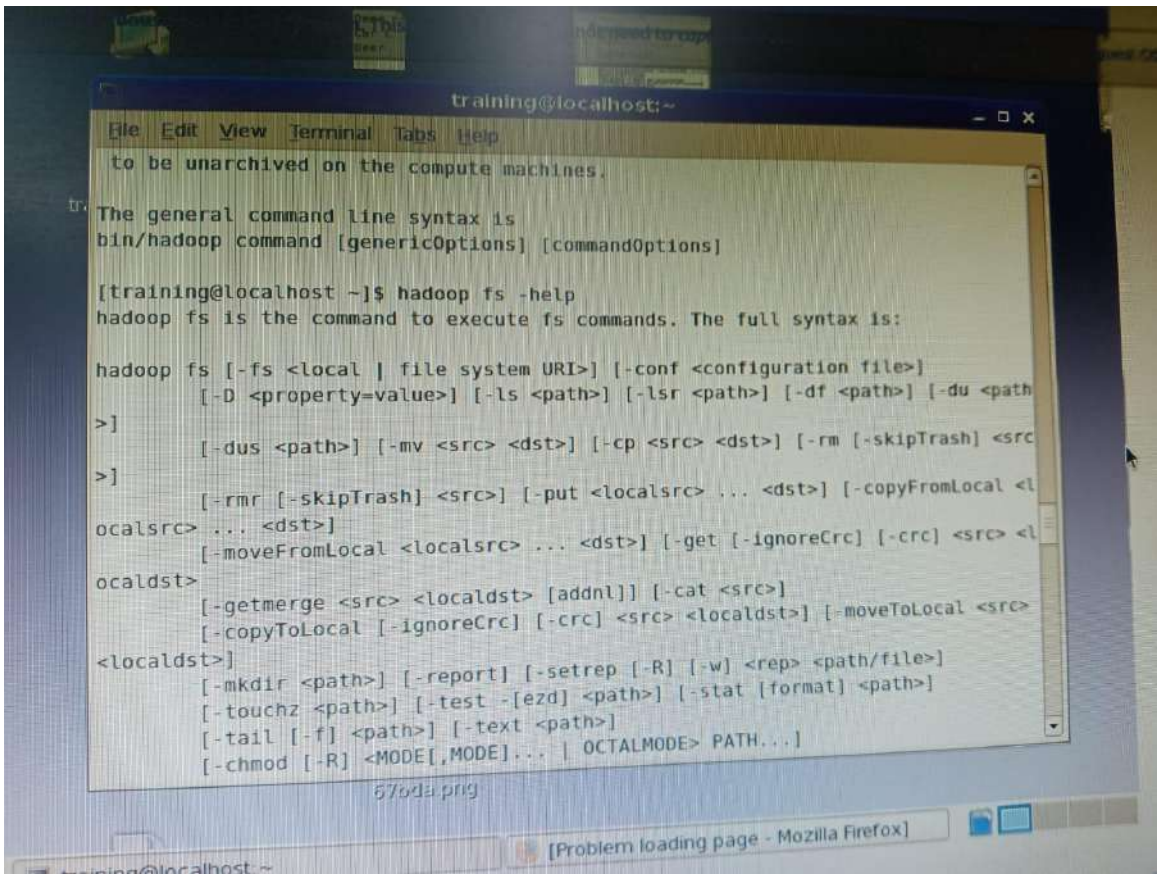

File Edit View Terminal Tabs Help
[-touchz <path>]
[-test -[ezd] <path>]
[-stat [format] <path>]
[-tail [-f] <file>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-chgrp [-R] GROUP PATH...]
[-help [cmd]]
Generic options supported are
-conf <configuration file> specify an application configuration file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated files to be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar files to include in the classpath.
-archives <comma separated list of archives> specify comma separated archives to be unarchived on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost cloudera]$ hadoop fs -cat /home/cloudera/myfirstdata/part-m-00000
1,suraj,9137171684,rishikewalya@gmail.com,rishi123,20,Fever
2,vivek,9087654641,kushalv238@gmail.com,kushal123,20,Cough
3,Manav,9787654641,,manavshinde@gmail.com,manav123,20,Headache
4,Abhishek U.,9754653255,,harshsorathiya@gmail.com,harsh123,20,Backpain
[training@localhost cloudera]$
[Problem loading page - ...
Java EE - Wordcount/src/...
training@localhost:/hom...
11:08 PM

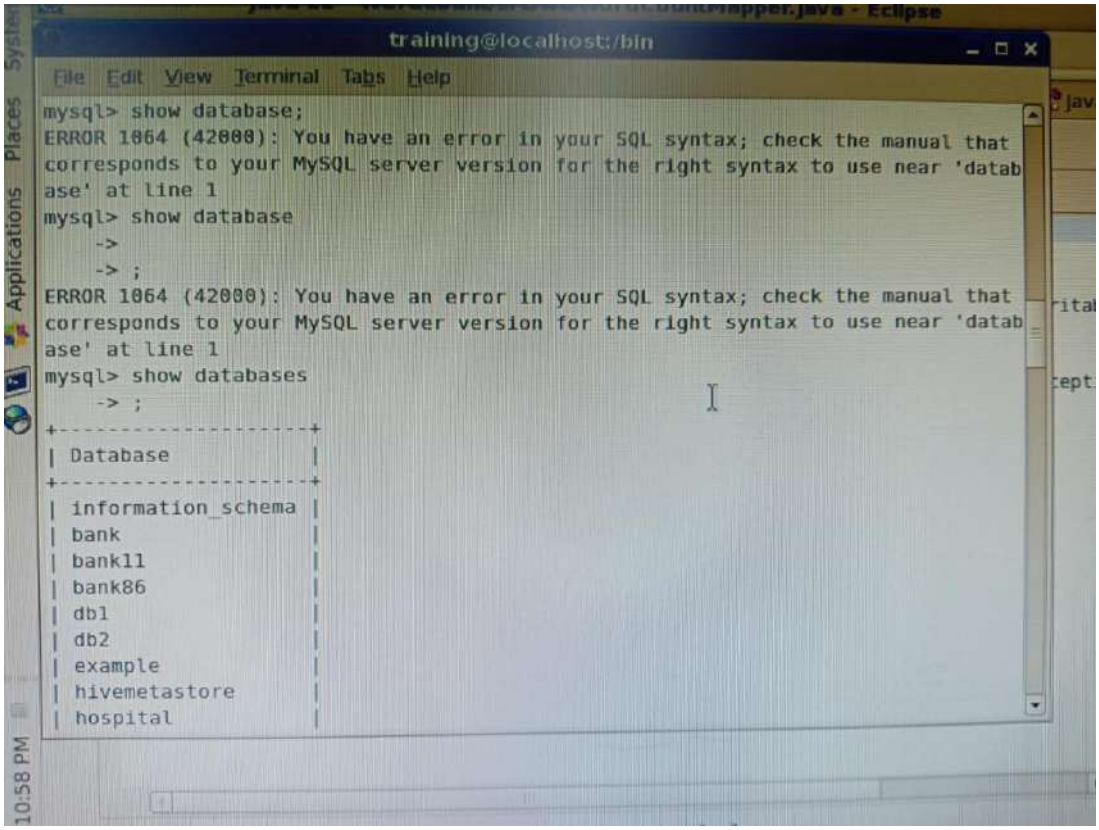

training@localhost:/home/cloudera
File Edit View Terminal Tabs Help
24/08/29 23:06:47 INFO mapred.JobClient: map 100% reduce 0%
24/08/29 23:06:47 INFO mapred.JobClient: Job complete: job_202408292212_0001
24/08/29 23:06:47 INFO mapred.JobClient: Counters: 12
24/08/29 23:06:47 INFO mapred.JobClient: Job Counters
24/08/29 23:06:47 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=3569
24/08/29 23:06:47 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
24/08/29 23:06:47 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
24/08/29 23:06:47 INFO mapred.JobClient: Launched map tasks=1
24/08/29 23:06:47 INFO mapred.JobClient: SLOTS_MILLIS_REDUCES=0
24/08/29 23:06:47 INFO mapred.JobClient: FileSystemCounters
24/08/29 23:06:47 INFO mapred.JobClient: HDFS_BYTES_READ=87
24/08/29 23:06:47 INFO mapred.JobClient: FILE_BYTES_WRITTEN=65897
24/08/29 23:06:47 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=254
24/08/29 23:06:47 INFO mapred.JobClient: Map-Reduce Framework
24/08/29 23:06:47 INFO mapred.JobClient: Map input records=4
24/08/29 23:06:47 INFO mapred.JobClient: Spilled Records=0
24/08/29 23:06:47 INFO mapred.JobClient: Map output records=4
24/08/29 23:06:47 INFO mapred.JobClient: SPLIT_RAW_BYTES=87
24/08/29 23:06:47 INFO mapreduce.ImportJobBase: Transferred 254 bytes in 5.5706 seconds (45.5968 bytes/sec)
24/08/29 23:06:47 INFO mapreduce.ImportJobBase: Retrieved 4 records.
[training@localhost cloudera]$
Writable
Smart Insert
1 : 1

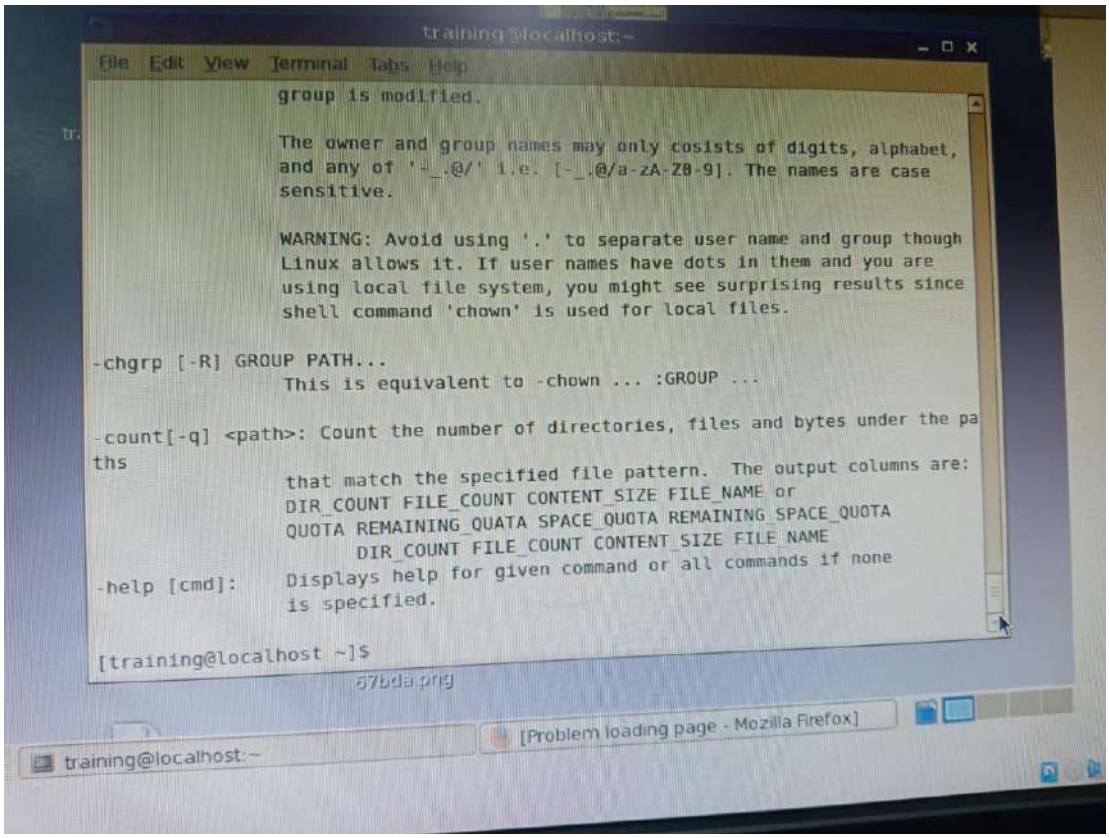

training@localhost:/bin
File Edit View Terminal Tabs Help
4 rows in set (0.00 sec)
mysql> update register set name='Abhishek U.' where name='Harsh';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> select * from register;
patientid	name	number	email	password	age	disease
1	suraj	9137171684	rishikewalya@gmail.com	rishi123	20	Fever
2	vivek	9087654641	kushalv238@gmail.com	kushal123	20	Cough
3	Manav	9787654641	,manavshinde@gmail.com	manav123	20	Headache
4	Abhishek U.	9754653255	,harshsorathiya@gmail.com	harsh123	20	Backpain

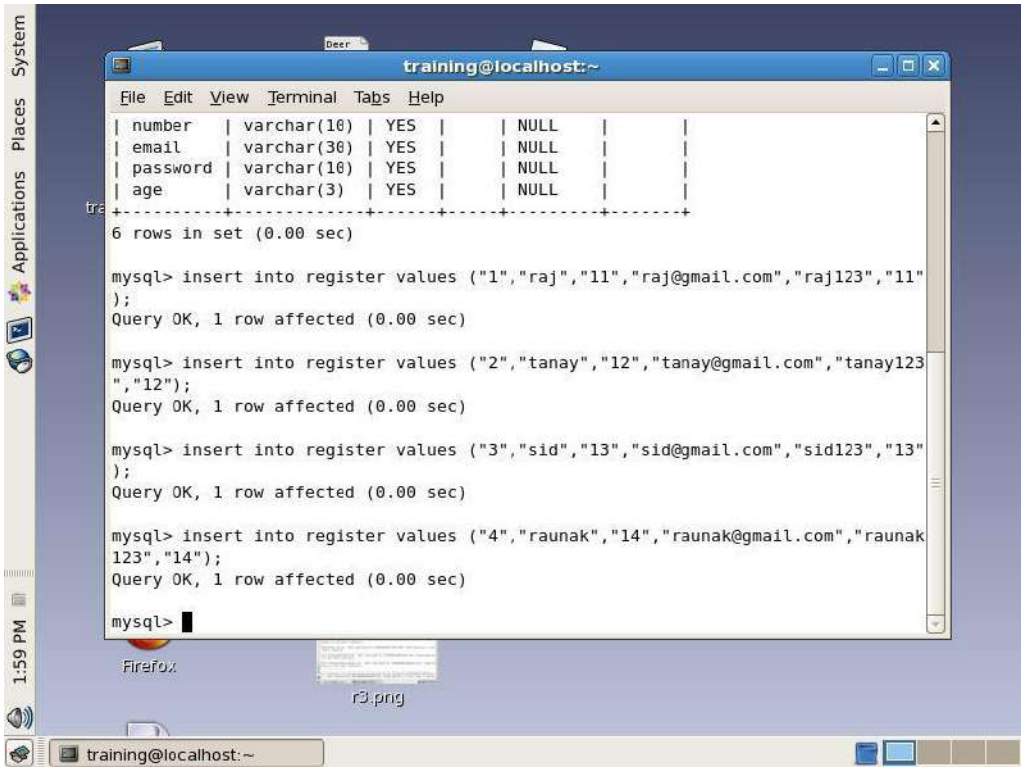

training@localhost:/bin
File Edit View Terminal Tabs Help
4 rows in set (0.00 sec)
mysql>
mysql> describe register;
Field	Type	Null	Key	Default	Extra
patientid	varchar(20)	YES		NULL	
name	varchar(20)	YES		NULL	
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
password	varchar(10)	YES		NULL	
age	varchar(3)	YES		NULL	
disease	varchar(30)	YES		NULL	
7 rows in set (0.00 sec)
mysql>
Writable
Smart Insert
1 : 1
training@localhost:/bin

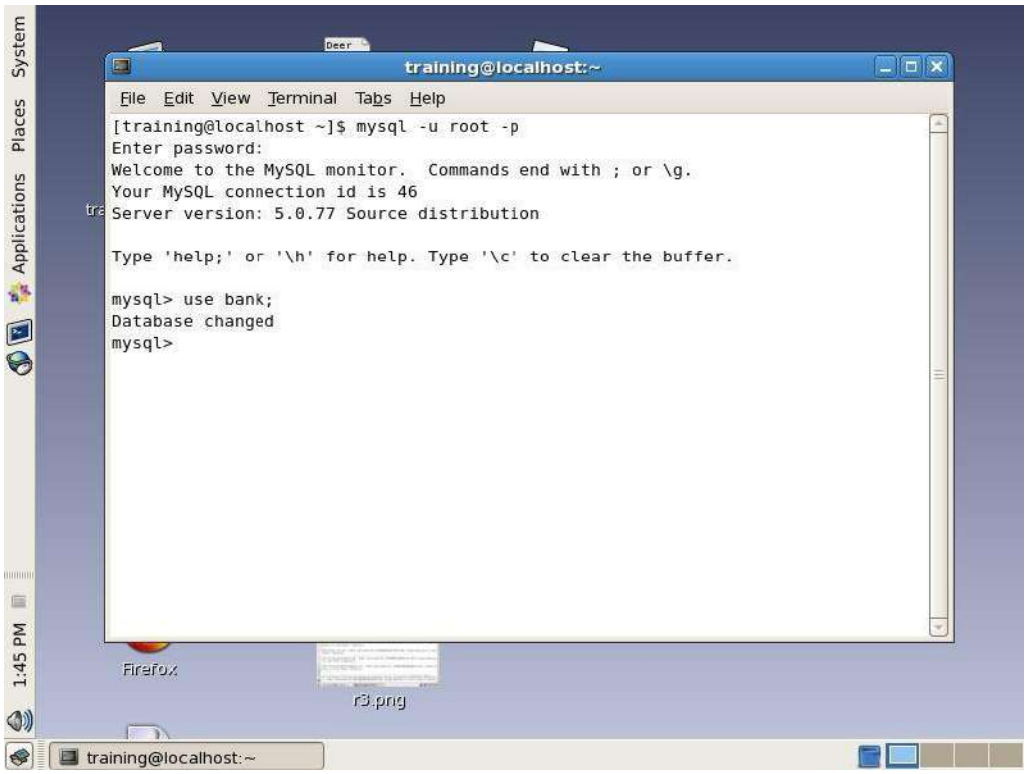

training@localhost:/bin
File Edit View Terminal Tabs Help
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+--------------------+
| Tables_in_hospital |
+--------------------+
| register |
+--------------------+
1 row in set (0.00 sec)
mysql> select * from register;
+-----------+--------+------------+---------------------------+-----------+-----
-+----------+
| patientid | name | number | email | password | age
| disease |
+-----------+--------+------------+---------------------------+-----------+-----
-+----------+
| 1 | Rishi | 9137171684 | rishikewalya@gmail.com | rishi123 | 20
| Fever |
| 2 | Kushal | 9087654641 | kushalv238@gmail.com | kushal123 | 20
| Cough |
| 3 | Manav | 9787654641 | ,manavshinde@gmail.com | manav123 | 20
| Headache |
Writable
Smart Insert
1 : 1

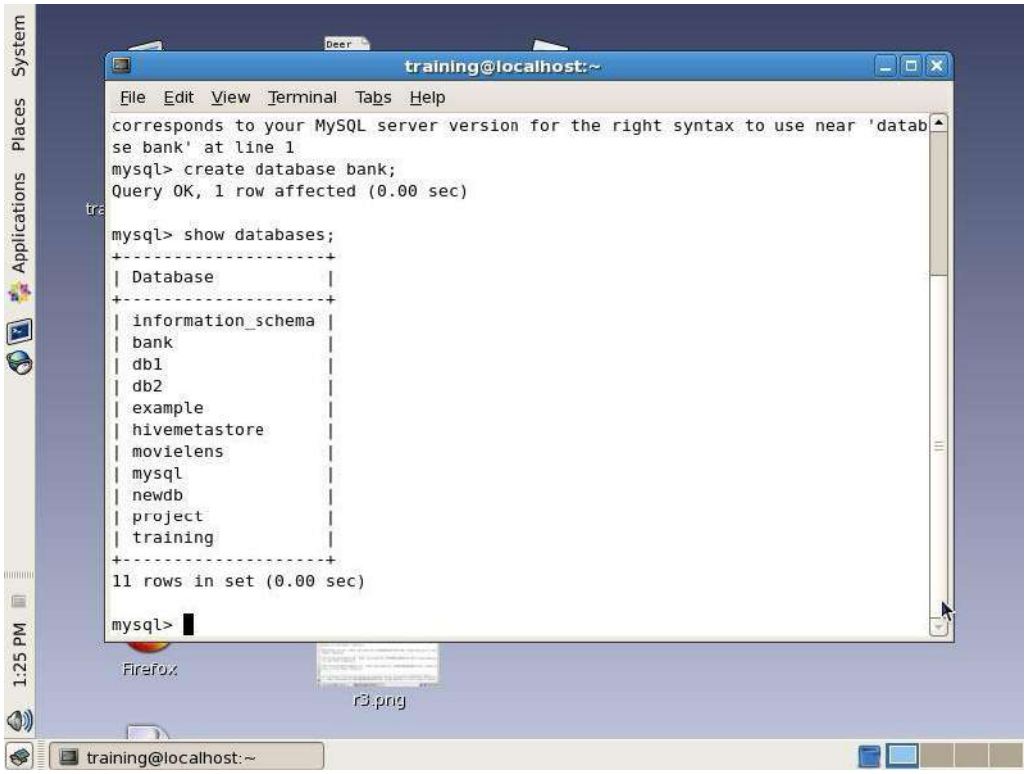

training@localhost:~
File Edit View Terminal Tabs Help
to be unarchived on the compute machines.
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
[training@localhost ~]$ hadoop fs -help
hadoop fs is the command to execute fs commands. The full syntax is:
hadoop fs [-fs <local | file system URI>] [-conf <configuration file>]
[-D <property=value>] [-ls <path>] [-lsr <path>] [-df <path>] [-du <path
>]
[-dus <path>] [-mv <src> <dst>] [-cp <src> <dst>] [-rm [-skipTrash] <src
>]
[-rmr [-skipTrash] <src>] [-put <localsrc> ... <dst>] [-copyFromLocal <l
ocalsrc> ... <dst>]
[-moveFromLocal <localsrc> ... <dst>] [-get [-ignoreCrc] [-crc] <src> <l
ocaldst>
[-getmerge <src> <localdst> [addnl]] [-cat <src>]
[-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>] [-moveToLocal <src>
<localdst>]
[-mkdir <path>] [-report] [-setrep [-R] [-w] <rep> <path/file>]
[-touchz <path>] [-test -[ezd] <path>] [-stat [format] <path>]
[-tail [-f] <path>] [-text <path>]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[Problem loading page - Mozilla Firefox]

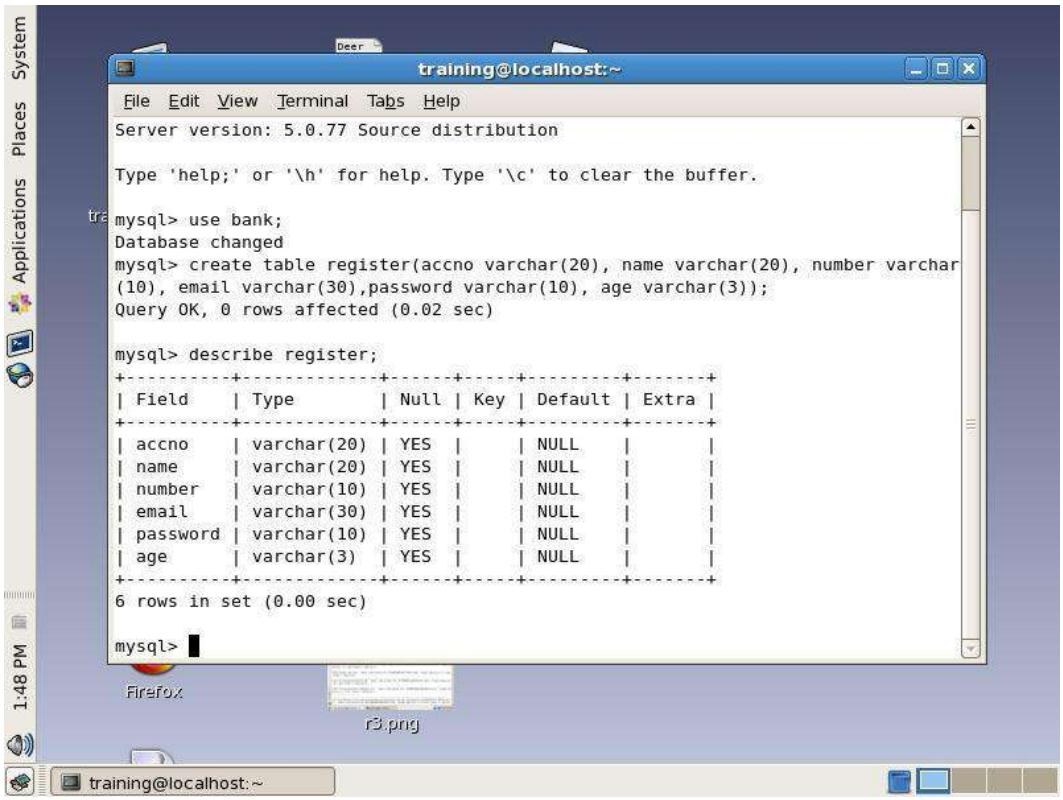

training@localhost:/bin
File Edit View Terminal Tabs Help
mysql> show database;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'database' at line 1
mysql> show database
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'database' at line 1
mysql> show databases
-> ;
+--------------------+
| Database |
+--------------------+
| information_schema |
| bank |
| bank11 |
| bank86 |
| db1 |
| db2 |
| example |
| hivemetastore |
| hospital |

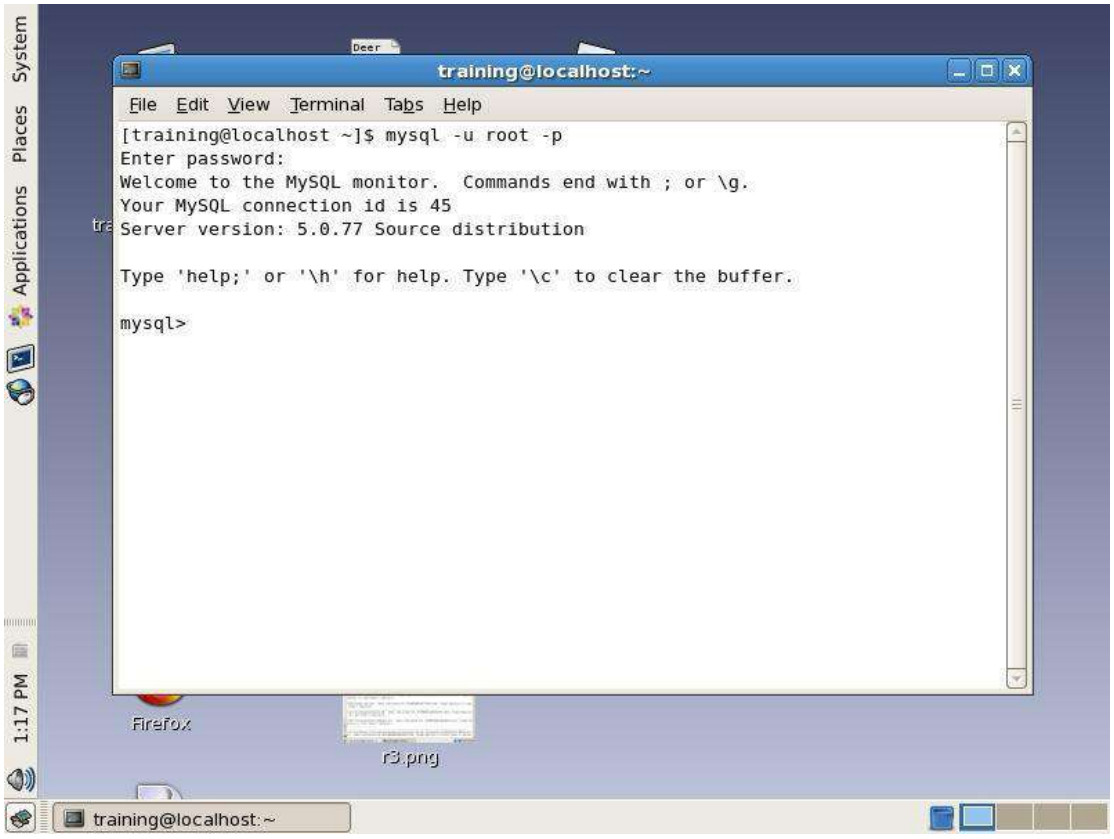

training@localhost:~
File Edit View Terminal Tabs Help
group is modified.
The owner and group names may only cosists of digits, alphabet, and any of '-_.@/' i.e. [-_.@/a-zA-Z0-9]. The names are case sensitive.
WARNING: Avoid using '.' to separate user name and group though Linux allows it. If user names have dots in them and you are using local file system, you might see surprising results since shell command 'chown' is used for local files.
-chgrp [-R] GROUP PATH...
This is equivalent to -chown ... :GROUP ...
-count[-q] <path>: Count the number of directories, files and bytes under the paths
that match the specified file pattern. The output columns are:
DIR_COUNT FILE_COUNT CONTENT_SIZE FILE_NAME or
QUOTA REMAINING_QUATA SPACE_QUOTA REMAINING_SPACE_QUOTA
DIR_COUNT FILE_COUNT CONTENT_SIZE FILE_NAME
-help [cmd]: Displays help for given command or all commands if none is specified.
[training@localhost ~]$
57bda.png
training@localhost:~
[Problem loading page - Mozilla Firefox]

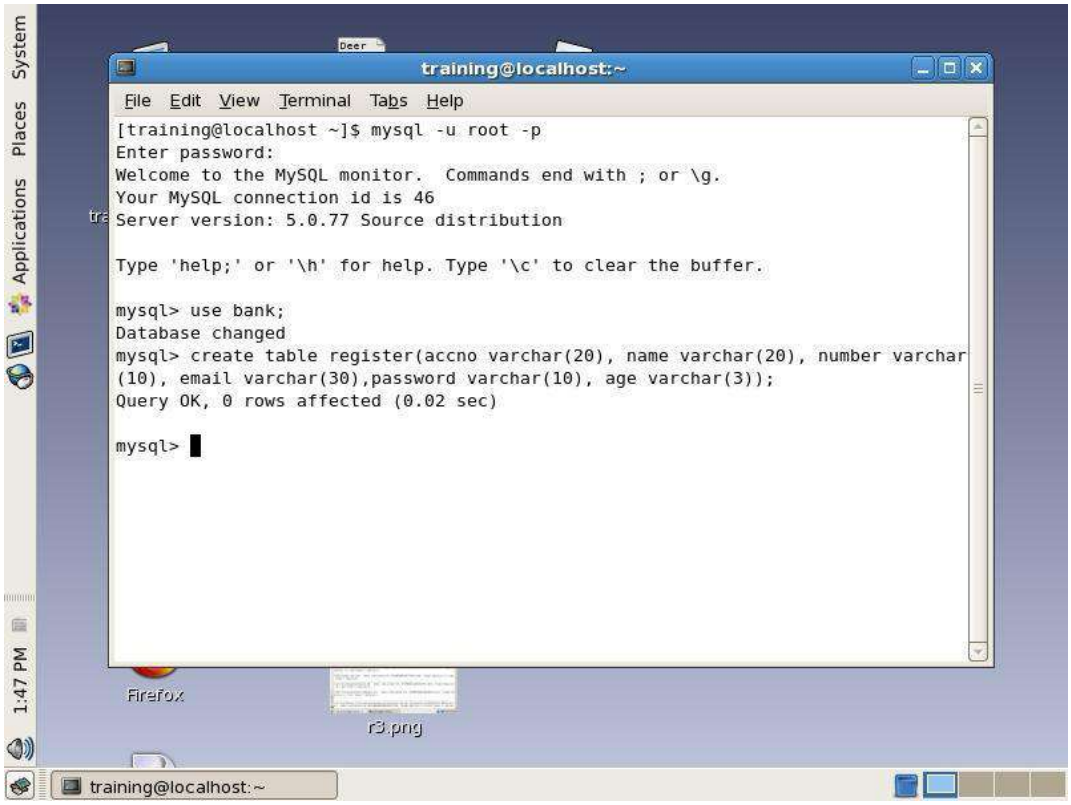

System
Places
Applications
1:59 PM
training@localhost:~
File Edit View Terminal Tabs Help
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
password	varchar(10)	YES		NULL	
age	varchar(3)	YES		NULL	
6 rows in set (0.00 sec)
mysql> insert into register values ("1","raj","11","raj@gmail.com","raj123","11"
);
Query OK, 1 row affected (0.00 sec)
mysql> insert into register values ("2","tanay","12","tanay@gmail.com","tanay123
","12");
Query OK, 1 row affected (0.00 sec)
mysql> insert into register values ("3","sid","13","sid@gmail.com","sid123","13"
);
Query OK, 1 row affected (0.00 sec)
mysql> insert into register values ("4","raunak","14","raunak@gmail.com","raunak
123","14");
Query OK, 1 row affected (0.00 sec)
mysql>
Firefox
r3.png
training@localhost:~

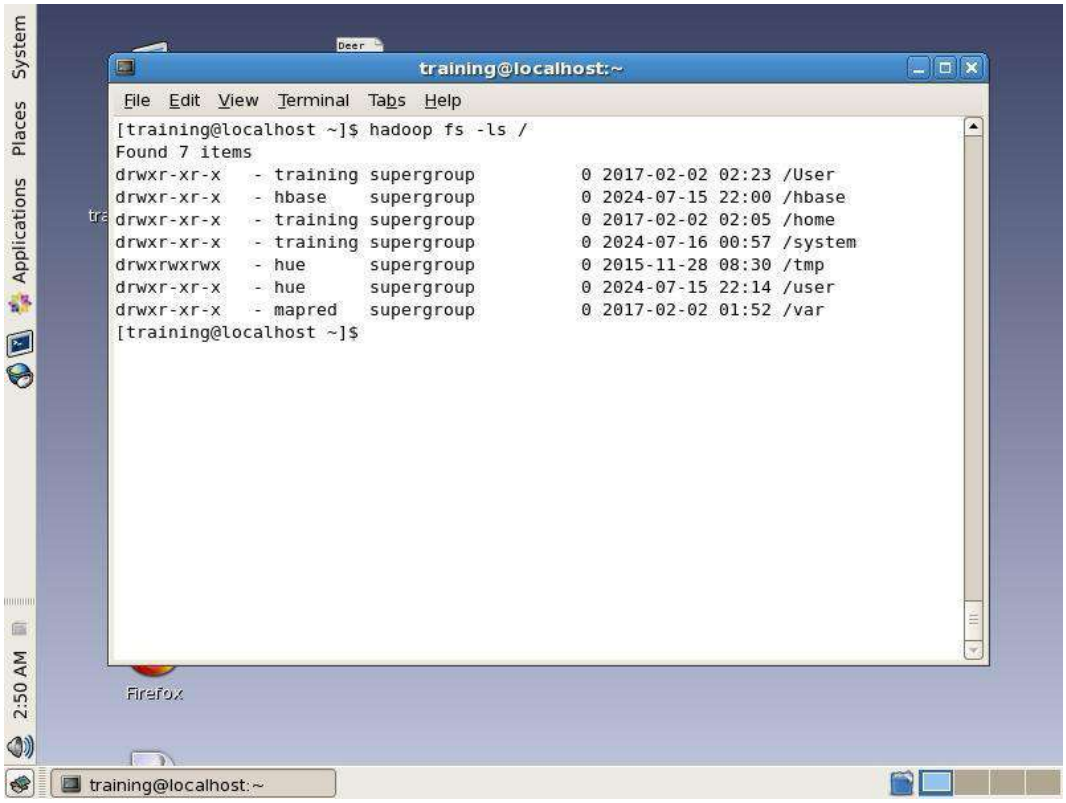

System
Places
Applications
1:45 PM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use bank;
Database changed
mysql>
Firefox
r3.png
training@localhost:~


System
Places
Applications
1:25 PM
training@localhost:~
File Edit View Terminal Tabs Help
corresponds to your MySQL server version for the right syntax to use near 'datab
se bank' at line 1
mysql> create database bank;
Query OK, 1 row affected (0.00 sec)
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| bank |
| db1 |
| db2 |
| example |
| hivemetastore |
| movielens |
| mysql |
| newdb |
| project |
| training |
+--------------------+
11 rows in set (0.00 sec)
mysql>
Firefox
r3.png
training@localhost:~


System
Places
Applications
1:48 PM
training@localhost:~
File Edit View Terminal Tabs Help
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use bank;
Database changed
mysql> create table register(accno varchar(20), name varchar(20), number varchar
(10), email varchar(30),password varchar(10), age varchar(3));
Query OK, 0 rows affected (0.02 sec)
mysql> describe register;
+----------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
accno	varchar(20)	YES		NULL	
name	varchar(20)	YES		NULL	
number	varchar(10)	YES		NULL	
email	varchar(30)	YES		NULL	
password	varchar(10)	YES		NULL	
age	varchar(3)	YES		NULL	
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
mysql>
Firefox
r3.png
training@localhost:~


System
Places
Applications
1:17 PM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 45
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
Firefox
r3.png
training@localhost:~


System
Places
Applications
1:47 PM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.0.77 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use bank;
Database changed
mysql> create table register(accno varchar(20), name varchar(20), number varchar
(10), email varchar(30),password varchar(10), age varchar(3));
Query OK, 0 rows affected (0.02 sec)
mysql>
Firefox
r3.png
training@localhost:~


System
Places
Applications
2:50 AM
training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hadoop fs -ls /
Found 7 items
drwxr-xr-x - training supergroup 0 2017-02-02 02:23 /User
drwxr-xr-x - hbase supergroup 0 2024-07-15 22:00 /hbase
drwxr-xr-x - training supergroup 0 2017-02-02 02:05 /home
drwxr-xr-x - training supergroup 0 2024-07-16 00:57 /system
drwxrwxrwx - hue supergroup 0 2015-11-28 08:30 /tmp
drwxr-xr-x - hue supergroup 0 2024-07-15 22:14 /user
drwxr-xr-x - mapred supergroup 0 2017-02-02 01:52 /var
[training@localhost ~]$
Firefox
training@localhost:~

BDA: Experiment – 4

Aim: Experiment for Word Counting using Hadoop Map-Reduce.

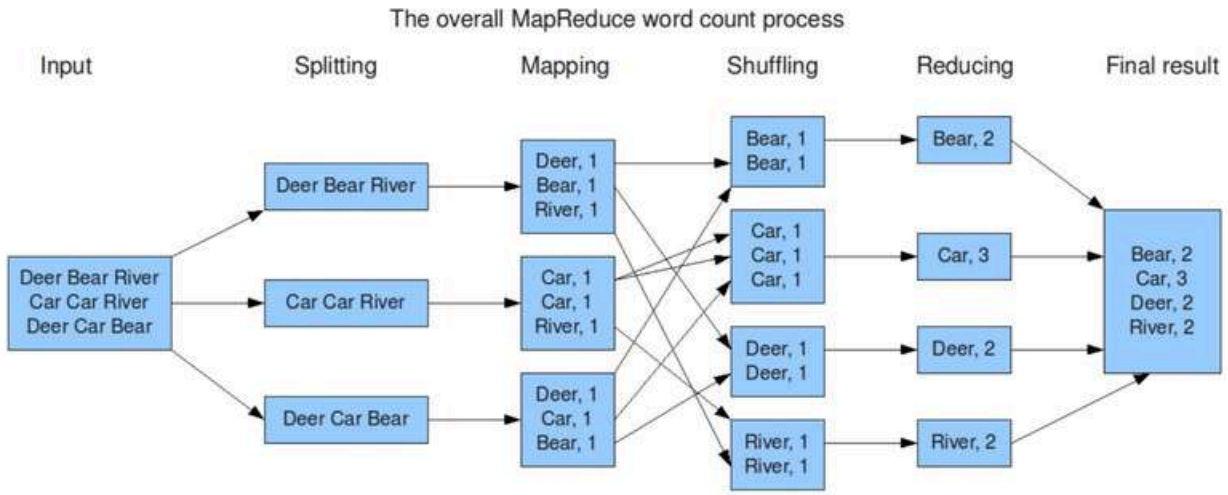
Theory:

- **Hadoop MapReduce** is the core framework of **Apache Hadoop** for processing large datasets in a distributed environment.
- It follows a programming model that divides a task into two fundamental phases: **Map** and **Reduce**.
- This framework enables parallel processing of data across a large cluster of commodity hardware, ensuring scalability and fault tolerance.
- MapReduce allows developers to write programs that process vast amounts of data using a divide-and-conquer approach. It is especially designed for processing unstructured and semi-structured data stored in **HDFS** (Hadoop Distributed File System).

Key Concepts

1. **Map Phase**: The **Map** phase is the first stage of the MapReduce job, where data is split into smaller sub-tasks. Each sub-task is processed in parallel by **Mapper** functions. These functions take input in the form of key-value pairs and output a transformed set of intermediate key-value pairs.
2. **Reduce Phase**: The **Reduce** phase is the second stage, where the intermediate data from the **Map** phase is grouped and aggregated. The **Reducer** functions take the intermediate key-value pairs, perform further processing (e.g., aggregating results), and output the final result.
3. **Key-Value Pairs**: The fundamental data structure in MapReduce is the key-value pair. The **Mapper** takes input data and generates intermediate key-value pairs, and the **Reducer** processes these intermediate pairs to generate the final output.

4. **Splitting**: The input data is divided into fixed-size chunks or **splits**. Each split is processed by a separate **Mapper** in parallel, which allows the job to scale across multiple nodes in a cluster.
5. **Shuffling and Sorting**: After the **Map** phase, the **Shuffle and Sort** step ensures that all values associated with the same key are sent to the same **Reducer**. This step is critical for grouping and sorting the intermediate data before it is passed to the **Reduce** phase.



MapReduce Workflow

The execution of a MapReduce job consists of the following key stages:

1. **Input Splitting**: The input data is divided into smaller, manageable chunks (splits), typically based on the block size in HDFS (e.g., 128MB). Each split is assigned to a different **Mapper**.
2. **Mapping**: The **Mapper** reads the input split, processes it, and generates intermediate key-value pairs. For instance, in a word count example, the Mapper would read a text file and emit each word along with a count of 1.
3. **Shuffling and Sorting**: The framework automatically groups and sorts the intermediate key-value pairs by key. The intermediate data

is shuffled so that values associated with the same key are brought together. Sorting ensures that all keys are processed in order.

4. **Reducing**: The **Reducer** processes the intermediate key-value pairs, performing operations like aggregation or filtering. For example, in the word count task, the Reducer would sum the counts for each word and output the total count for each word.
5. **Final Output**: The final result from the **Reduce** phase is written to **HDFS**. The output is typically stored in a format such as text files or sequence files.

Advantages/Features of Hadoop MapReduce

1. Scalability:
    a. Easily handles large datasets by distributing tasks across many nodes.
2. Fault Tolerance:
    a. Automatically handles failures and replicates data to avoid loss.
3. Distributed Processing:
    a. Processes data simultaneously across nodes for faster results.
4. Data Locality:
    a. Moves computation to where the data resides, improving efficiency.
5. Handling Unstructured Data

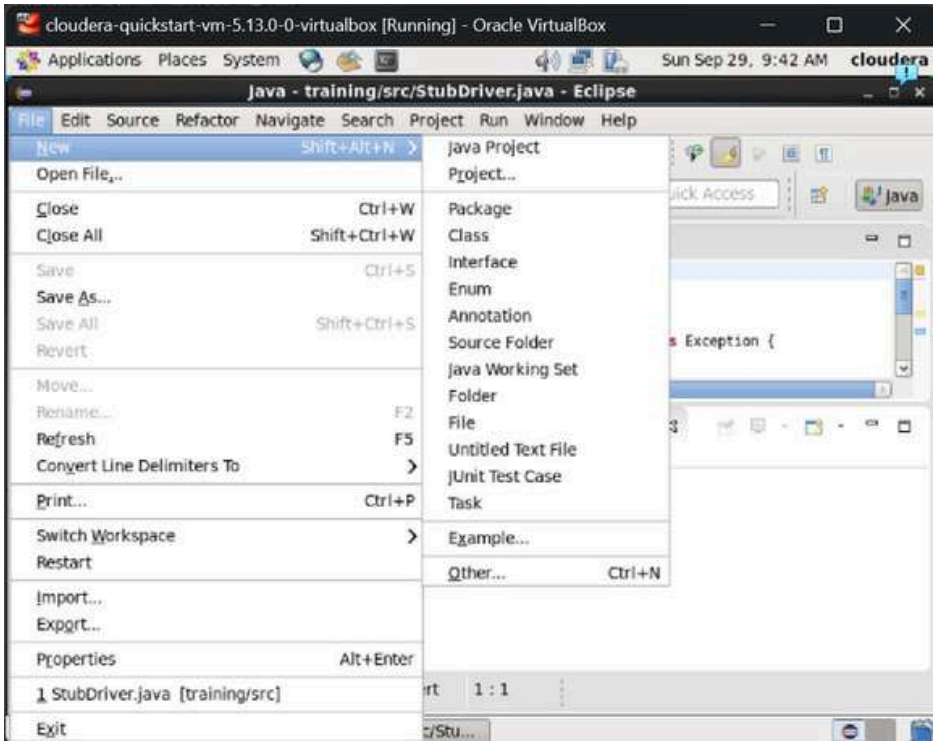
Limitations of Hadoop Map Reduce:

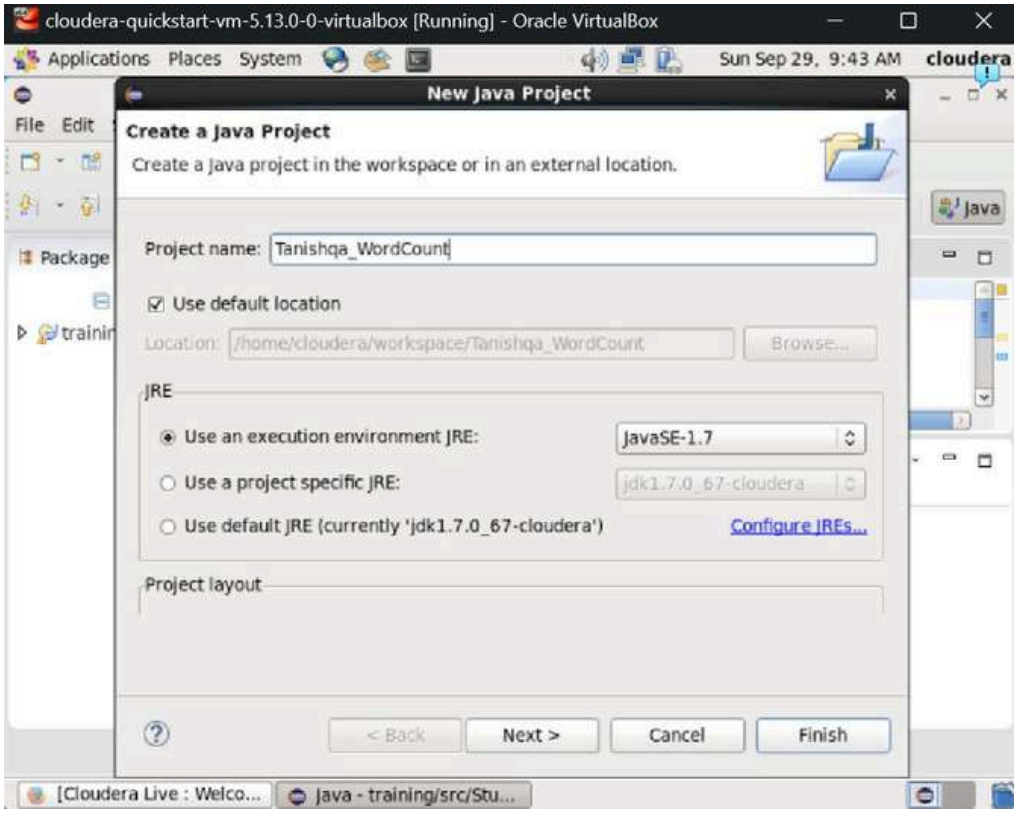
Complex Programming: Requires custom code for every task, making it harder to use.

High Latency: Not ideal for real-time processing due to batch-based nature.

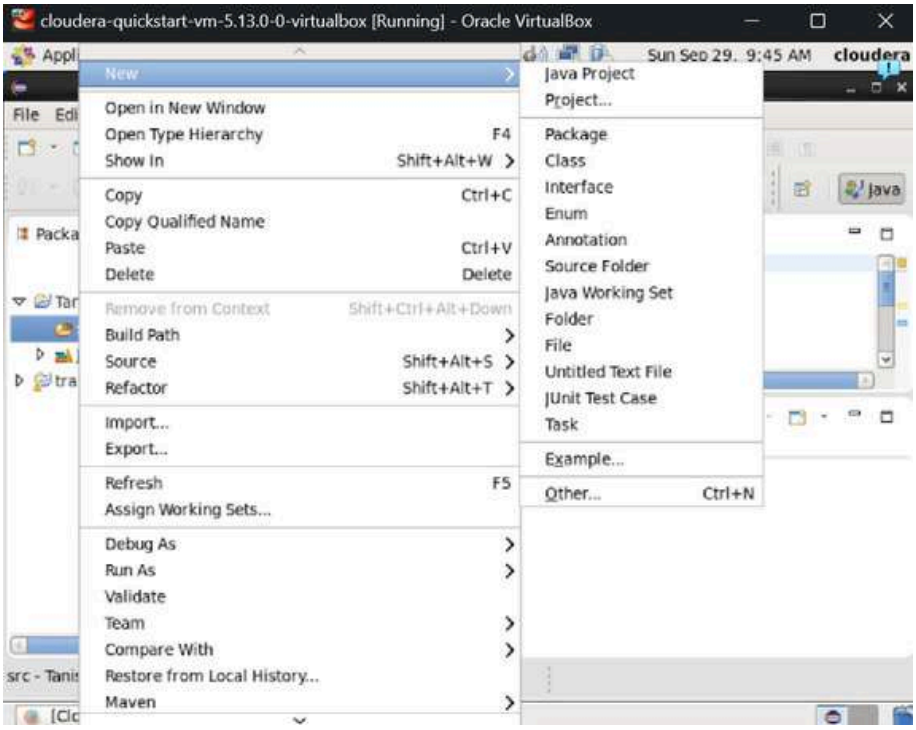
Not Suitable for Small Data: Overhead can be significant for smaller datasets.

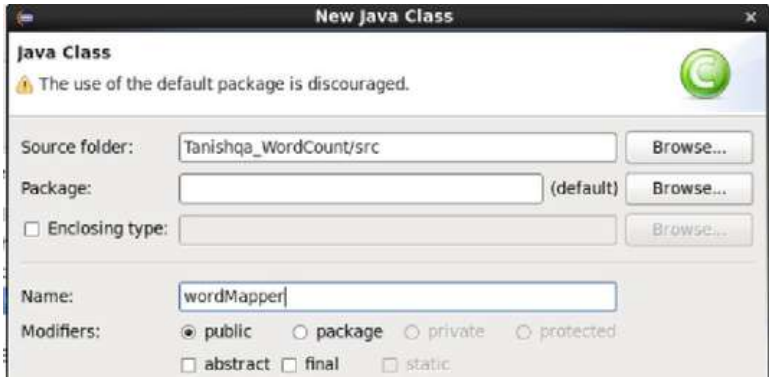
1. Open the Cloudera VM application using Oracle VirtualBox, navigate to the Eclipse application and create a new Java project.

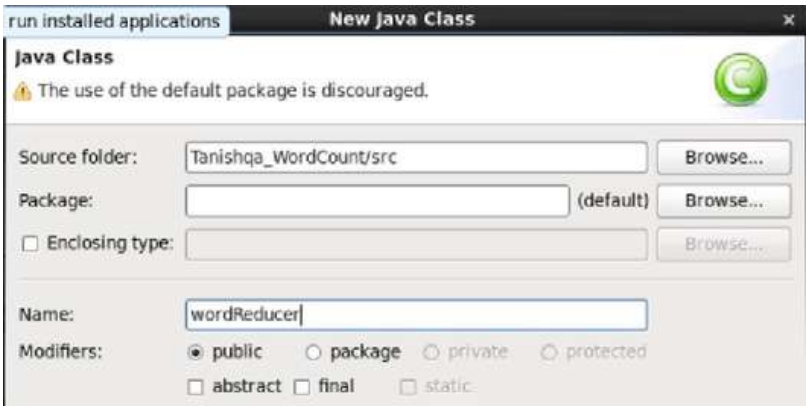




2. Right click on the project and create three classes – wordCount, wordMapper and wordReducer.







3. Copy and paste the following code into each of the created classes:

wordCount:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
public class wordCount {
        public static void main(String[] args) throws Exception {
                if (args.length != 2) {
                        System.out.printf("Usage: WordCount \n");
                        System.exit(-1);
                }
                Job job = new Job();
                job.setJarByClass(wordCount.class);
                job.setJobName("wordCount");
                FileInputFormat.setInputPaths(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                job.setMapperClass(wordMapper.class);
                job.setReducerClass(wordReducer.class);
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(IntWritable.class);
```

```
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                boolean success = job.waitForCompletion(true);
                System.exit(success ? 0 : 1);
        }
}
```

wordMapper-

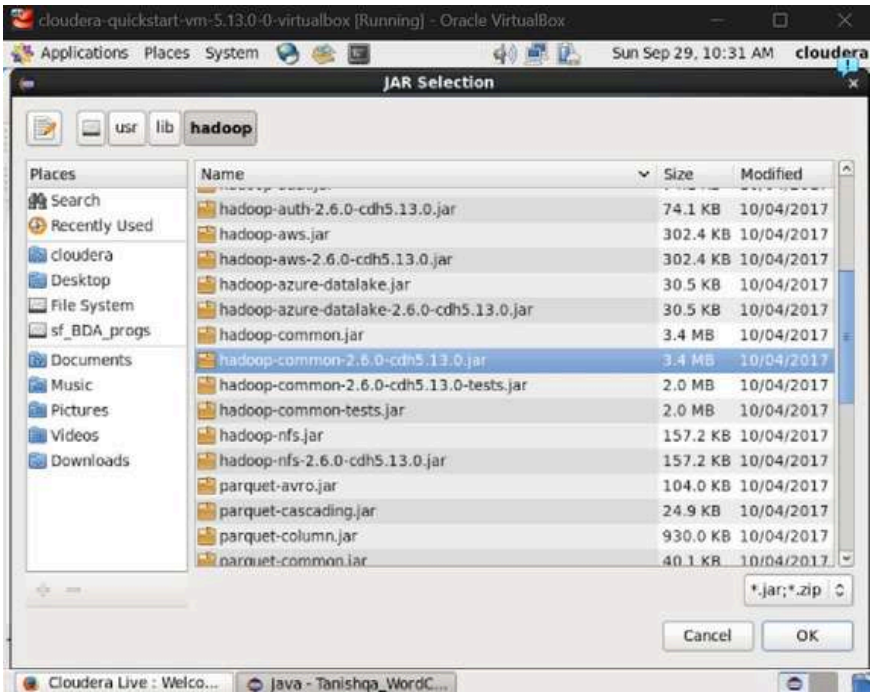
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class wordMapper extends Mapper {
        @Override
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
                String line = value.toString();
                for (String word : line.split("\\W+")) {
                        if (word.length() > 0)
                                context.write(new Text(word), new IntWritable(1));
                }
        }
}
```

wordReducer:

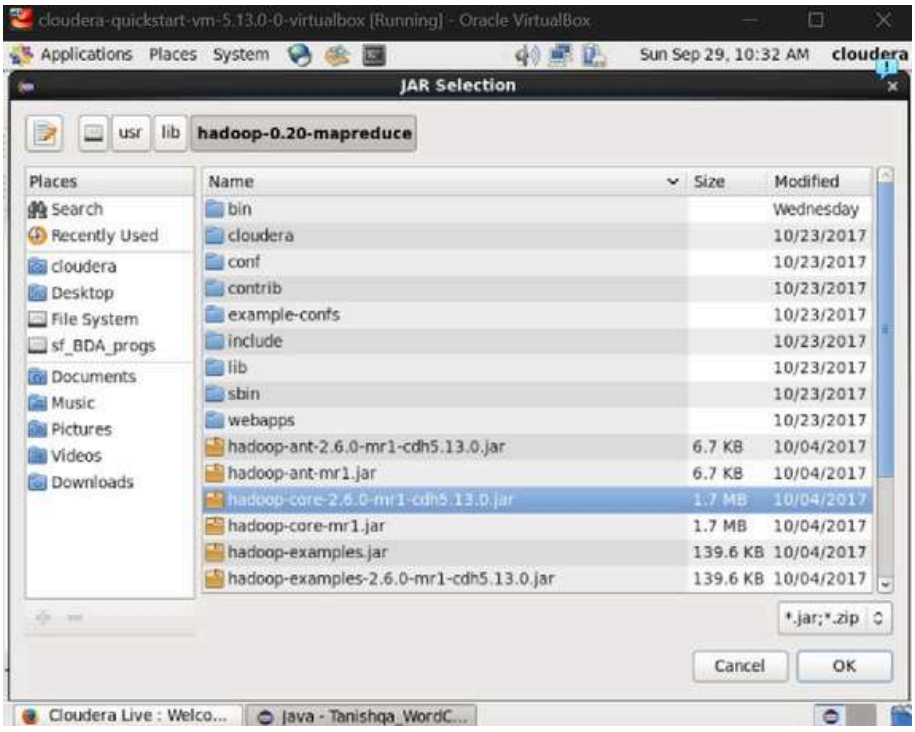
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class wordReducer extends Reducer {
        @Override
        public void reduce(Text key, Iterable values, Context context) throws
IOException, InterruptedException {
                int wordCount = 0;
                for (IntWritable value : values)
                        wordCount += value.get();
                context.write(key, new IntWritable(wordCount));
        }
```

```
}
```

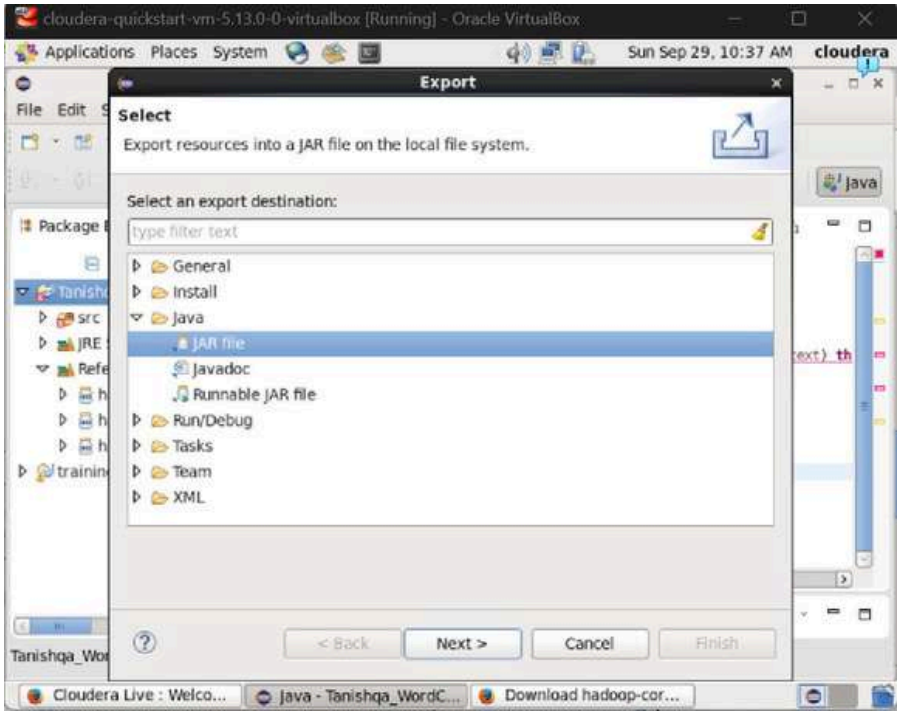
4. Now, right click on the project and navigate to Build Path and then Add External Archives. Now, navigate to **usr > lib > hadoop** and select the following JAR file.



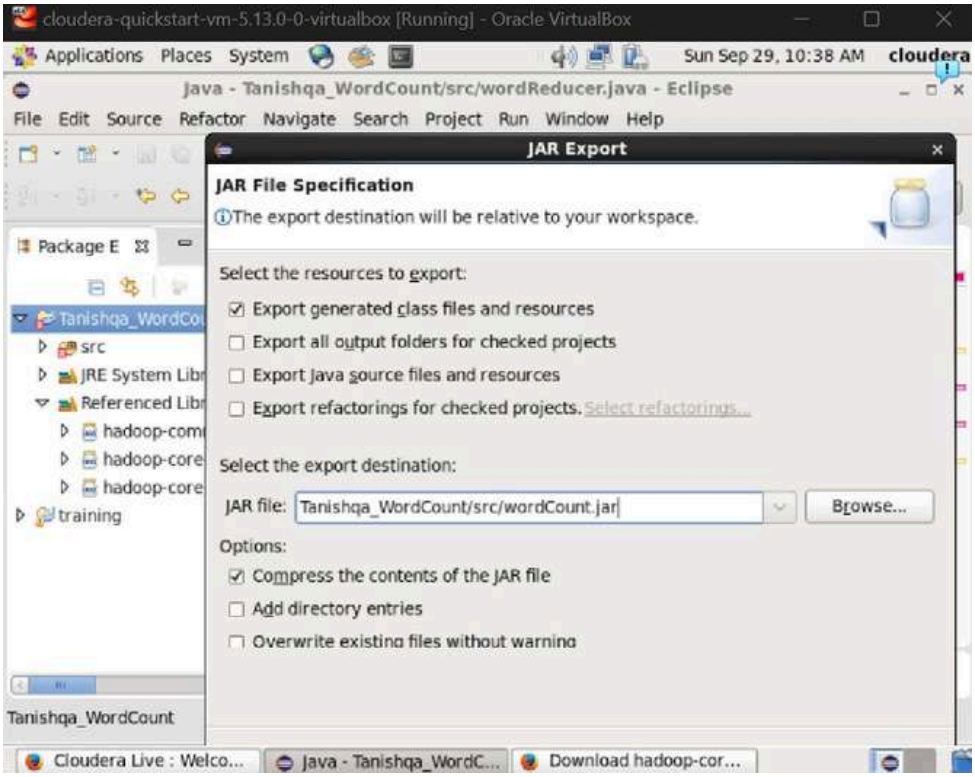
5. Similarly add the following JAR file as well



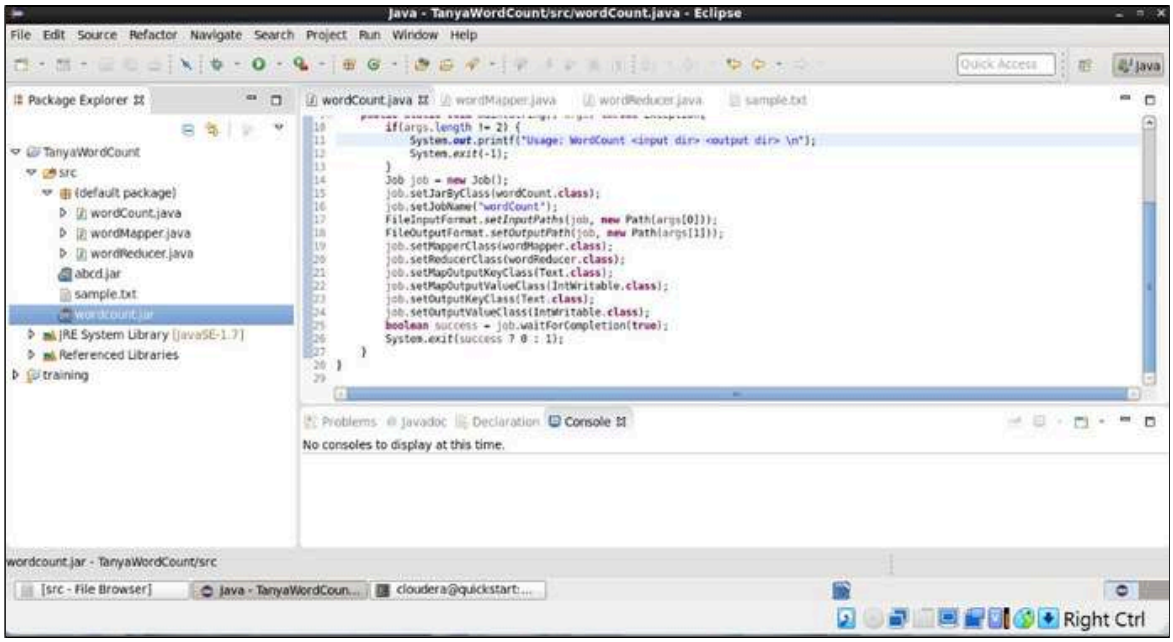
6. Download the file – **hadoop-core-1.2.1.jar** via Mozilla Firefox and add it to the project like shown above



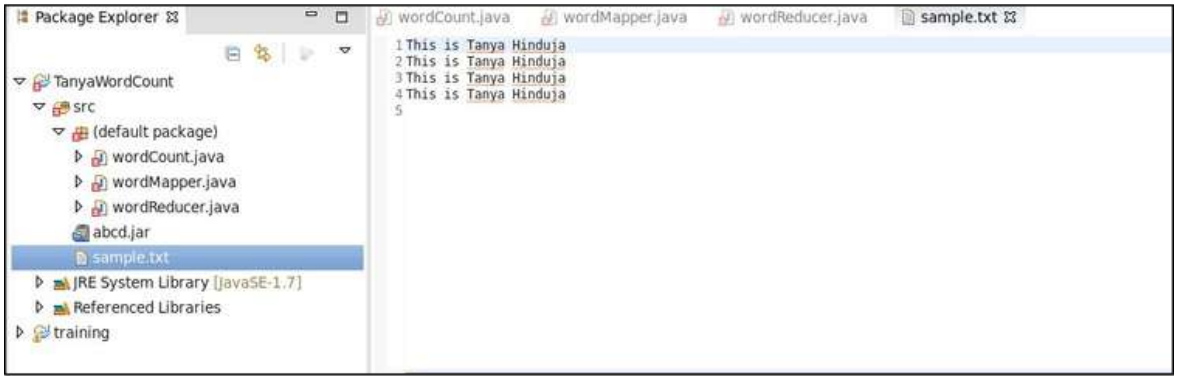
7. Now, right click on the Java project and export it as a JAR file to the same location as the wordcount, wordMapper and wordReducer classes



8. Now, right click on the Java project and export it as a JAR file to the same location as the wordcount, wordMapper and wordReducer classes



9. Set up a new txt file in the src folder, with content of your choice, as seen below



10. hdfs dfs -put /home/cloudera/workspace/TanyaWordCount/src/sample.txt /user/cloudera/ and then hdfs dfs -ls /user/cloudera

a. Next, open a terminal window, change current directory to the src folder and execute the following commands:

```
[cloudera@quickstart src]$ hdfs dfs -put /home/cloudera/workspace/TanyaWordCount/src/sample.txt /user/cloudera/
[cloudera@quickstart src]$ hdfs dfs -ls /user/cloudera/
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2024-09-18 15:15 /user/cloudera/apache_hadoop
-rw-r--r--   1 cloudera cloudera           88 2024-09-22 14:02 /user/cloudera/sample.txt
```

b. hadoop jar wordcount.jar wordCount sample.txt sampleoutdir

```
[cloudera@quickstart src]$ hdfs dfs -put /home/cloudera/workspace/TanyaWordCount/src/sample.txt /user/cloudera/
[cloudera@quickstart src]$ hdfs dfs -ls /user/cloudera/
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2024-09-18 15:15 /user/cloudera/apache_hadoop
-rw-r--r--   1 cloudera cloudera           88 2024-09-22 14:02 /user/cloudera/sample.txt
[cloudera@quickstart src]$ hadoop jar wordcount.jar wordCount sample.txt sampleoutdir
24/09/22 14:02:59 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/09/22 14:03:00 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
th ToolRunner to remedy this.
24/09/22 14:03:00 INFO input.FileInputFormat: Total input paths to process : 1
24/09/22 14:03:01 INFO mapreduce.JobSubmitter: number of splits:1
24/09/22 14:03:01 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1727037818973_0003
24/09/22 14:03:03 INFO impl.YarnClientImpl: Submitted application application_1727037818973_0003
24/09/22 14:03:03 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1727037818973_0003/
24/09/22 14:03:03 INFO mapreduce.Job: Running job: job_1727037818973_0003
24/09/22 14:03:21 INFO mapreduce.Job: Job job_1727037818973_0003 running in uber mode : false
24/09/22 14:03:21 INFO mapreduce.Job:  map 0% reduce 0%
24/09/22 14:03:33 INFO mapreduce.Job:  map 100% reduce 0%
24/09/22 14:03:41 INFO mapreduce.Job:  map 100% reduce 100%
24/09/22 14:03:41 INFO mapreduce.Job: Job job_1727037818973_0003 completed successfully
24/09/22 14:03:41 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=190
                FILE: Number of bytes written=287459
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=209
                HDFS: Number of bytes written=30
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
```


```
[cloudera@quickstart src]$ hdfs dfs -put /home/cloudera/workspace/TanyaWordCount/src/sample.txt /user/cloudera/
[cloudera@quickstart src]$ hdfs dfs -ls /user/cloudera/
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2024-09-18 15:15 /user/cloudera/apache_hadoop
-rw-r--r--   1 cloudera cloudera           88 2024-09-22 14:02 /user/cloudera/sample.txt
[cloudera@quickstart src]$ hadoop jar wordcount.jar wordCount sample.txt sampleoutdir
24/09/22 14:02:59 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/09/22 14:03:00 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
th ToolRunner to remedy this.
24/09/22 14:03:00 INFO input.FileInputFormat: Total input paths to process : 1
24/09/22 14:03:01 INFO mapreduce.JobSubmitter: number of splits:1
24/09/22 14:03:01 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1727037818973_0003
24/09/22 14:03:03 INFO impl.YarnClientImpl: Submitted application application_1727037818973_0003
24/09/22 14:03:03 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1727037818973_0003/
24/09/22 14:03:03 INFO mapreduce.Job: Running job: job_1727037818973_0003
24/09/22 14:03:21 INFO mapreduce.Job: Job job_1727037818973_0003 running in uber mode : false
24/09/22 14:03:21 INFO mapreduce.Job:  map 0% reduce 0%
24/09/22 14:03:33 INFO mapreduce.Job:  map 100% reduce 0%
24/09/22 14:03:41 INFO mapreduce.Job:  map 100% reduce 100%
24/09/22 14:03:41 INFO mapreduce.Job: Job job_1727037818973_0003 completed successfully
24/09/22 14:03:41 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=190
                FILE: Number of bytes written=287459
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=209
                HDFS: Number of bytes written=30
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
```


- hadoop fs -ls sampleoutdir

```
[cloudera@quickstart src]$ hadoop fs -ls sampleoutdir
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2024-09-22 14:03 sampleoutdir/_SUCCESS
-rw-r--r--   1 cloudera cloudera         30 2024-09-22 14:03 sampleoutdir/part-r-00000
[cloudera@quickstart src]$
```

- hadoop fs -cat sampleoutdir/part-r-00000

```
[cloudera@quickstart src]$ hadoop fs -cat sampleoutdir/part-r-00000
Hinduja 4
Tanya   4
This    4
is      4
[cloudera@quickstart src]$
```

[src - File Browser] | Java - TanyaWordCoun... | cloudera@quickstart:...

Experiment 05

Aim: Experiment on pig

Theory:

Pig Represents Big Data as data flows. Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes. First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally Pig Engine(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

Need of Pig: One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

It uses query approach which results in reducing the length of the code.
Pig Latin is SQL like language.
It provides many builtIn operators.
It provides nested data types (tuples, bags, map).

Commands:

1. **Load Data**

   The `LOAD` command is used to load data from the file system (HDFS or local) into a Pig relation.

```
data = LOAD '/user/hadoop/input/data.txt' USING PigStorage(',')
AS (id:int, name:chararray, age:int, city:chararray);
```

2. **Filter Data**
   The `FILTER` command selects rows from a dataset that meet a specific condition.

   adults = FILTER data BY age > 18;

3. **Group Data**
   The `GROUP` command groups data based on one or more fields.

   grouped_data = GROUP data BY city;

4. **Foreach - Generate**
   The `FOREACH` command is used to apply operations to each row of data and generate new columns or modify existing ones

   names_and_ages = FOREACH data GENERATE name, age;

5. **Join Data**

   The `JOIN` command merges two datasets based on a common field.

   joined_data = JOIN data BY id, other_data BY id;

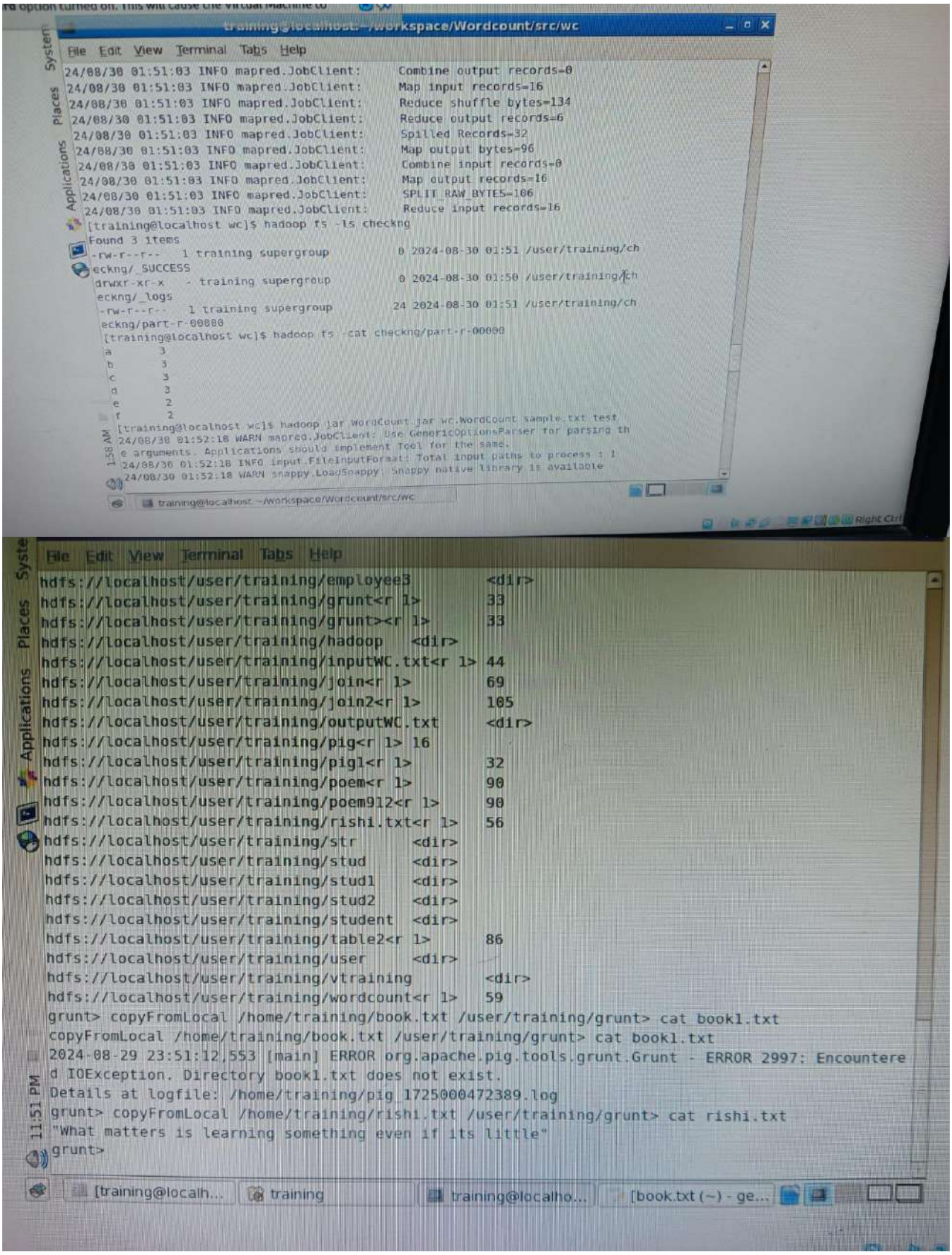
6. . **Store Data**

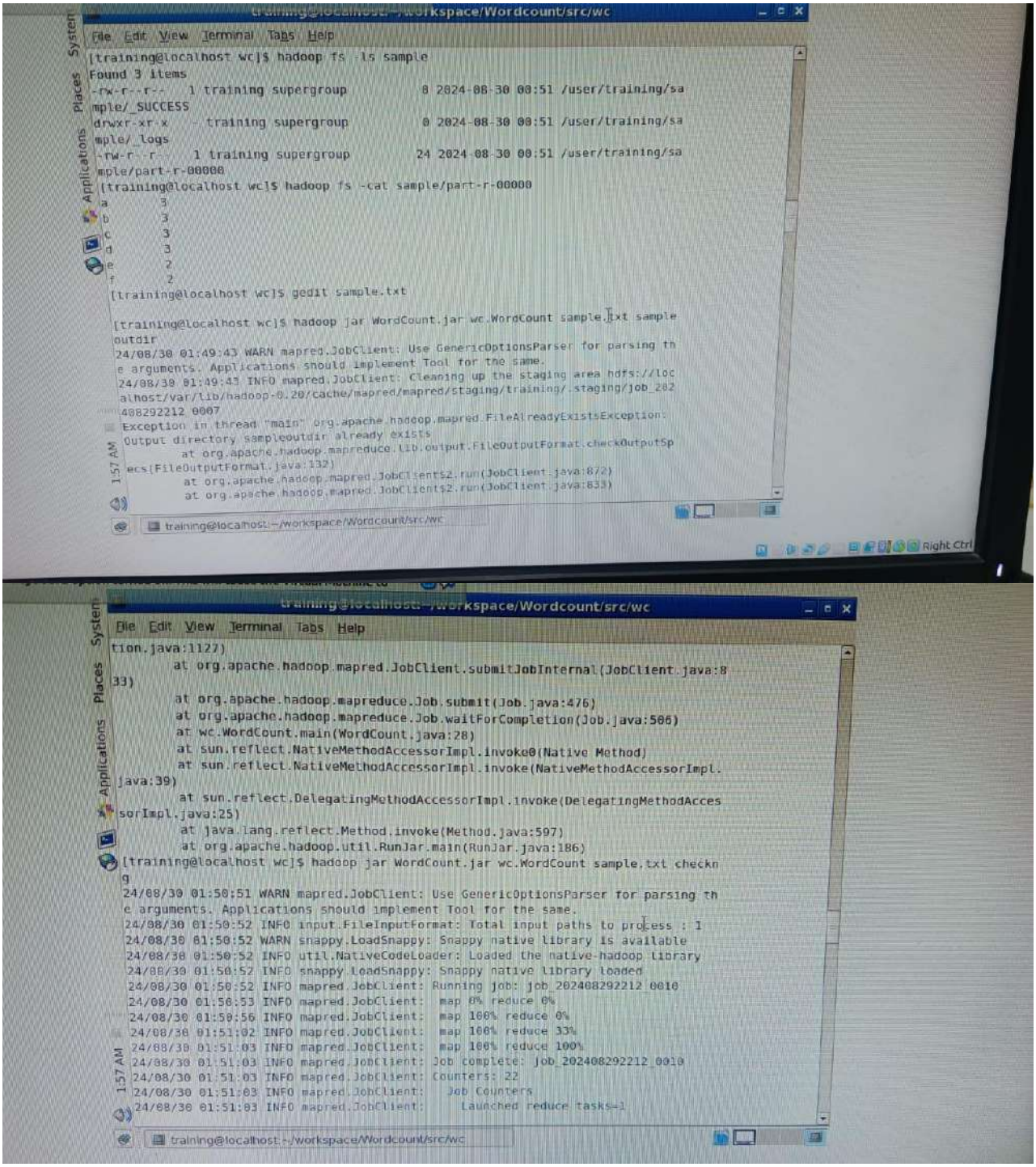
   The `STORE` command is used to save the results of a Pig script back to the file system (HDFS or local).

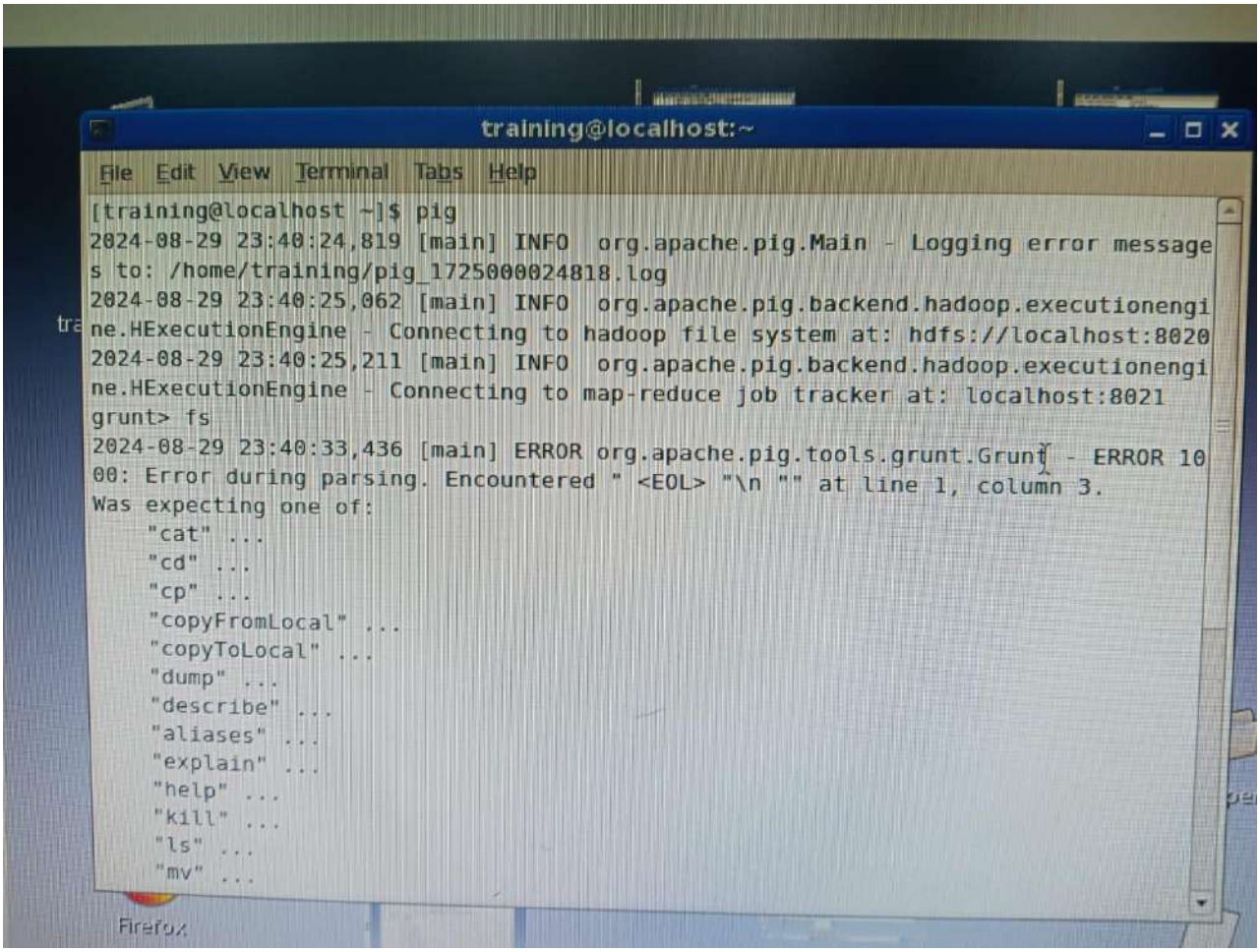
   STORE adults INTO '/user/hadoop/output/adults' USING PigStorage(',');

Apache Pig commands provide an efficient way to process large-scale datasets through a high-level abstraction, making data manipulation easier compared to traditional MapReduce.

Output:


training@localhost:~/workspace/Wordcount/src/wc
File Edit View Terminal Tabs Help
24/08/30 01:51:03 INFO mapred.JobClient: Combine output records=0
24/08/30 01:51:03 INFO mapred.JobClient: Map input records=16
24/08/30 01:51:03 INFO mapred.JobClient: Reduce shuffle bytes=134
24/08/30 01:51:03 INFO mapred.JobClient: Reduce output records=6
24/08/30 01:51:03 INFO mapred.JobClient: Spilled Records=32
24/08/30 01:51:03 INFO mapred.JobClient: Map output bytes=96
24/08/30 01:51:03 INFO mapred.JobClient: Combine input records=0
24/08/30 01:51:03 INFO mapred.JobClient: Map output records=16
24/08/30 01:51:03 INFO mapred.JobClient: SPLIT_RAW_BYTES=106
24/08/30 01:51:03 INFO mapred.JobClient: Reduce input records=16
[training@localhost wc]$ hadoop fs -ls checkng
Found 3 items
-rw-r--r-- 1 training supergroup 0 2024-08-30 01:51 /user/training/ch
eckng/_SUCCESS
drwxr-xr-x - training supergroup 0 2024-08-30 01:50 /user/training/ch
eckng/_logs
-rw-r--r-- 1 training supergroup 24 2024-08-30 01:51 /user/training/ch
eckng/part-r-00000
[training@localhost wc]$ hadoop fs -cat checkng/part-r-00000
a 3
b 3
c 3
d 3
e 2
f 2
[training@localhost wc]$ hadoop jar WordCount.jar wc.WordCount sample.txt test
24/08/30 01:52:18 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
24/08/30 01:52:18 INFO input.FileInputFormat: Total input paths to process : 1
24/08/30 01:52:18 WARN snappy.LoadSnappy: Snappy native library is available
training@localhost:~/workspace/Wordcount/src/wc
File Edit View Terminal Tabs Help
hdfs://localhost/user/training/employee3 <dir>
hdfs://localhost/user/training/grunt<r 1> 33
hdfs://localhost/user/training/grunt><r 1> 33
hdfs://localhost/user/training/hadoop <dir>
hdfs://localhost/user/training/inputWC.txt<r 1> 44
hdfs://localhost/user/training/join<r 1> 69
hdfs://localhost/user/training/join2<r 1> 105
hdfs://localhost/user/training/outputWC.txt <dir>
hdfs://localhost/user/training/pig<r 1> 16
hdfs://localhost/user/training/pig1<r 1> 32
hdfs://localhost/user/training/poem<r 1> 90
hdfs://localhost/user/training/poem912<r 1> 90
hdfs://localhost/user/training/rishi.txt<r 1> 56
hdfs://localhost/user/training/str <dir>
hdfs://localhost/user/training/stud <dir>
hdfs://localhost/user/training/stud1 <dir>
hdfs://localhost/user/training/stud2 <dir>
hdfs://localhost/user/training/student <dir>
hdfs://localhost/user/training/table2<r 1> 86
hdfs://localhost/user/training/user <dir>
hdfs://localhost/user/training/vtraining <dir>
hdfs://localhost/user/training/wordcount<r 1> 59
grunt> copyFromLocal /home/training/book.txt /user/training/grunt> cat book1.txt
copyFromLocal /home/training/book.txt /user/training/grunt> cat book1.txt
2024-08-29 23:51:12,553 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 2997: Encountere
d IOException. Directory book1.txt does not exist.
Details at logfile: /home/training/pig_1725000472389.log
grunt> copyFromLocal /home/training/rishi.txt /user/training/grunt> cat rishi.txt
"What matters is learning something even if its little"
grunt>
[training@localh... training training@localho... [book.txt (~) - ge...


File Edit View Terminal Tabs Help
[training@localhost wc]$ hadoop fs -ls sample
Found 3 items
-rw-r--r-- 1 training supergroup 8 2024-08-30 00:51 /user/training/sample/_SUCCESS
drwxr-xr-x - training supergroup 0 2024-08-30 00:51 /user/training/sample/_logs
-rw-r--r-- 1 training supergroup 24 2024-08-30 00:51 /user/training/sample/part-r-00000
[training@localhost wc]$ hadoop fs -cat sample/part-r-00000
a 3
b 3
c 3
d 3
e 2
f 2
[training@localhost wc]$ gedit sample.txt
[training@localhost wc]$ hadoop jar WordCount.jar wc.WordCount sample.txt sampleoutdir
24/08/30 01:49:43 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/08/30 01:49:43 INFO mapred.JobClient: Cleaning up the staging area hdfs://localhost/var/lib/hadoop-0.20/cache/mapred/mapred/staging/training/.staging/job_202408292212_0007
Exception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory sampleoutdir already exists
at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:132)
at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:872)
at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:833)
training@localhost:~/workspace/Wordcount/src/wc
Right Ctrl
File Edit View Terminal Tabs Help
tion.java:1127)
at org.apache.hadoop.mapred.JobClient.submitJobInternal(JobClient.java:833)
at org.apache.hadoop.mapreduce.Job.submit(Job.java:476)
at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:506)
at wc.WordCount.main(WordCount.java:28)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at org.apache.hadoop.util.RunJar.main(RunJar.java:186)
[training@localhost wc]$ hadoop jar WordCount.jar wc.WordCount sample.txt checkng
24/08/30 01:50:51 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/08/30 01:50:52 INFO input.FileInputFormat: Total input paths to process : 1
24/08/30 01:50:52 WARN snappy.LoadSnappy: Snappy native library is available
24/08/30 01:50:52 INFO util.NativeCodeLoader: Loaded the native-hadoop library
24/08/30 01:50:52 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/30 01:50:52 INFO mapred.JobClient: Running job: job_202408292212_0010
24/08/30 01:50:53 INFO mapred.JobClient: map 0% reduce 0%
24/08/30 01:50:56 INFO mapred.JobClient: map 100% reduce 0%
24/08/30 01:51:02 INFO mapred.JobClient: map 100% reduce 33%
24/08/30 01:51:03 INFO mapred.JobClient: map 100% reduce 100%
24/08/30 01:51:03 INFO mapred.JobClient: Job complete: job_202408292212_0010
24/08/30 01:51:03 INFO mapred.JobClient: Counters: 22
24/08/30 01:51:03 INFO mapred.JobClient: Job Counters
24/08/30 01:51:03 INFO mapred.JobClient: Launched reduce tasks=1
training@localhost:~/workspace/Wordcount/src/wc


training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ pig
2024-08-29 23:40:24,819 [main] INFO org.apache.pig.Main - Logging error messages to: /home/training/pig_1725000024818.log
2024-08-29 23:40:25,062 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2024-08-29 23:40:25,211 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
grunt> fs
2024-08-29 23:40:33,436 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1000: Error during parsing. Encountered " <EOL> "\n "" at line 1, column 3.
Was expecting one of:
"cat" ...
"cd" ...
"cp" ...
"copyFromLocal" ...
"copyToLocal" ...
"dump" ...
"describe" ...
"aliases" ...
"explain" ...
"help" ...
"kill" ...
"ls" ...
"mv" ...
Firefox

Experiment 6

Aim: Implementation of HIVE Commands.

Theory:

HIVE

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

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

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

HQL

Hive defines a simple SQL-like query language for querying and managing large datasets called Hive-QL (HQL). It's easy to use if you're familiar with SQL Language. Hive allows programmers who are familiar with the language to write the custom MapReduce framework to perform more sophisticated analysis.

Uses of Hive:

1. The Apache Hive distributed storage.

2. By using Hive, we can access files stored in Hadoop Distributed File System (HDFS is used for querying and managing large datasets residing in) or in other data storage systems such as Apache HBase.

Components of HIVE

1. **Metastore :**
   Hive stores the schema of the Hive tables in a Hive Metastore. Metastore is used to hold all the information about the tables and partitions that are in the warehouse. By default, the metastore is run in the same process as the Hive service and the default Metastore is DerBy Database.
2. **SerDe :**
   Serializer, Deserializer gives instructions to hive on how to process a record.

HIVE Organization

The data are organized in three different formats in HIVE.

Tables: They are very similar to RDBMS tables and contain rows and tables. Hive is just layered over the Hadoop File System (HDFS), hence tables are directly mapped to directories of the filesystems. It also supports tables stored in other native file systems.

Partitions: Hive tables can have more than one partition. They are mapped to subdirectories and file systems as well.

Buckets: In Hive data may be divided into buckets. Buckets are stored as files in a partition in the underlying file system.

Hive also has metastore which stores all the metadata. It is a relational database containing various information related to Hive Schema (column types, owners, key-value data, statistics, etc.). We can use MySQL database over here.

Limitations of HIVE

• Hive is not designed for Online transaction processing (OLTP), it is only used for the Online Analytical Processing.

• Hive supports overwriting or apprehending data, but not updates and deletes.

• In Hive, sub queries are not supported.

Hive, an alternate for Pig

The following are the reasons why Hive is used in spite of Pig's availability:

- Hive-QL is a declarative language line SQL, PigLatin is a data flow language.
- Pig: a data-flow language and environment for exploring very large datasets.
- Hive: a distributed data warehouse.

COMMANDS

1. To enter hive terminal

Command: hive

```
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
```

2. To check the databases

Command: show databases;

```
hive> show databases;
OK
default
Time taken: 0.841 seconds, Fetched: 1 row(s)
```

3. To check the tables

Command: show tables;

```
hive> show tables;
OK
Time taken: 0.257 seconds
```

4. To use a particular database

Command: use dbname;

```
hive> use bank;
FAILED: SemanticException [Error 10072]: Database does not exist: bank
```

5.

6.

7.

8. To create database *Command:*

create database retail;

```
hive> create database bank;
OK
Time taken: 2.565 seconds
hive> show databases;
OK
bank
default
Time taken: 0.018 seconds, Fetched: 2 row(s)
```

6. To create table emp in retail database *Command: create table <tablename>;* Output:

hive> use retail;

hive> create table emp(id INT,name STRING,sal DOUBLE) row format delimited fields terminated by ',' stored as textfile;

```
hive> use bank;
OK
Time taken: 0.058 seconds
```

```
hive> create table emp(id INT,name STRING,sal DOUBLE)row format delimited fields terminated by ',' stored as textfile
;
OK
Time taken: 0.288 seconds
```

7. Schema information of table *Command: describe <tablename>;* Output:

hive> use retail;

hive> describe emp;

```
hive> show tables;
OK
emp
Time taken: 0.022 seconds, Fetched: 1 row(s)
hive> describe emp;
OK
id                      int
name                    string
```

8.To create file in training folder and save
as demo.txt
1,abc,2000
2,pqr,4500
To view contents of demo.txt file
[training@localhost ~]$ cat /home/training/demo.txt
1,abc,2000
2,pqr,4500
To load data from local path
hive> load data local inpath '/home/training/demo.txt' into table emp;

```
[cloudera@quickstart ~]$ cat /home/cloudera/demo.txt
10,raj,1000
11,raunak,5000
12,sid,2000
13,tanay,4000
14,manav,3000
[cloudera@quickstart ~]$
```

```
hive> load data local inpath '/home/cloudera/demo.txt' into table emp;
Loading data to table bank.emp
Table bank.emp stats: [numFiles=1, totalSize=67]
OK
Time taken: 1.047 seconds
```

9.To view contents of table
*Command: select * from emp;*
Output:

```
hive> select * from emp;
OK
10      raj     1000.0
11      raunak  5000.0
12      sid     2000.0
13      tanay   4000.0
14      manav   3000.0
Time taken: 0.561 seconds, Fetched: 5 row(s)
hive>
```

10. To rename table name

Command: ALTER TABLE old_table_name RENAME TO new_table_name;

Output:

hive> use retail;

hive> alter table emp rename to emp_sal;

```
hive> alter table emp rename to emp_sal;
OK
Time taken: 0.298 seconds
```

11. Selecting data

*hive> select * from emp_sal where id=1;*

```
hive> select * from emp_sal where id =12;
OK
12      sid     2000.0
Time taken: 0.423 seconds, Fetched: 1 row(s)
hive>
```

12. To count number of records in table

hive> select count(*) from emp_sal;

```
hive> select count(*) from emp_sal;
Query ID = cloudera_20220827004040_0c678dd5-7203-497f-a7f9-e06e3ff6e37d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661580441152_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_16615
_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661580441152_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-27 00:40:23,876 Stage-1 map = 0%,  reduce = 0%
2022-08-27 00:40:33,868 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.87 sec
2022-08-27 00:40:42,440 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.13 sec
MapReduce Total cumulative CPU time: 4 seconds 130 msec
Ended Job = job_1661580441152_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.13 sec   HDFS Read: 6924 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 130 msec
OK
5
Time taken: 34.233 seconds, Fetched: 1 row(s)
hive cloudera@quickstart:~
```

13. Try using aggregate commands using HQL(Try creating tables with group by fields and
execute the aggregate commands)

hive > select AVG(sal) as avg_salary from emp_sal;

```
hive> select AVG(sal) as avg_salary from emp_sal;
Query ID = cloudera_20220827004242_46d651c4-549a-4018-bbb3-c6157ab53f1a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661580441152_0004, Tracking URL = http://quickstart.cloudera:8088/proxy
_0004/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661580441152_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-27 00:42:30,850 Stage-1 map = 0%,  reduce = 0%
2022-08-27 00:42:41,592 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.95 sec
2022-08-27 00:42:50,088 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.19 sec
MapReduce Total cumulative CPU time: 4 seconds 190 msec
Ended Job = job_1661580441152_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.19 sec   HDFS Read: 7272 HDFS Write: 7
Total MapReduce CPU Time Spent: 4 seconds 190 msec
OK
3000.0
Time taken: 29.07 seconds, Fetched: 1 row(s)
```

hive > select MAX(sal) as max_salary from emp_sal;

```
hive> select MAX(sal) as max_salary from emp_sal;
Query ID = cloudera_20220827004343_1c64146e-98f2-4f69-b281-f1b8efad0e3f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661580441152_0005, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661580441152
_0005/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661580441152_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-27 00:43:48,458 Stage-1 map = 0%,  reduce = 0%
2022-08-27 00:43:57,987 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.95 sec
2022-08-27 00:44:08,603 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.23 sec
MapReduce Total cumulative CPU time: 6 seconds 230 msec
Ended Job = job_1661580441152_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.23 sec   HDFS Read: 7080 HDFS Write: 7 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 230 msec
OK
5000.0
Time taken: 29.538 seconds, Fetched: 1 row(s)
hive>
```

14. To drop table

hive> drop table emp_sal;

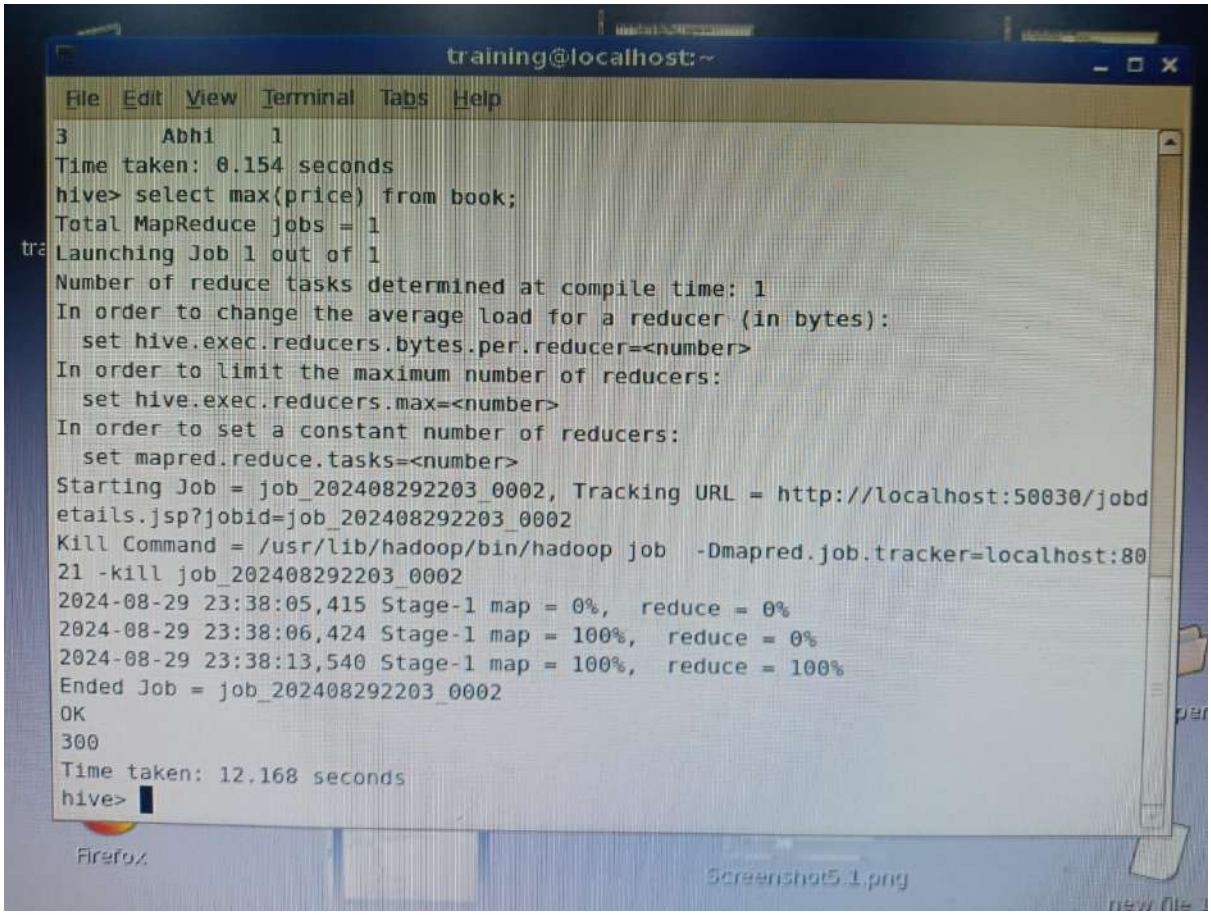
```
hive> drop table emp_sal;
OK
Time taken: 1.055 seconds
```

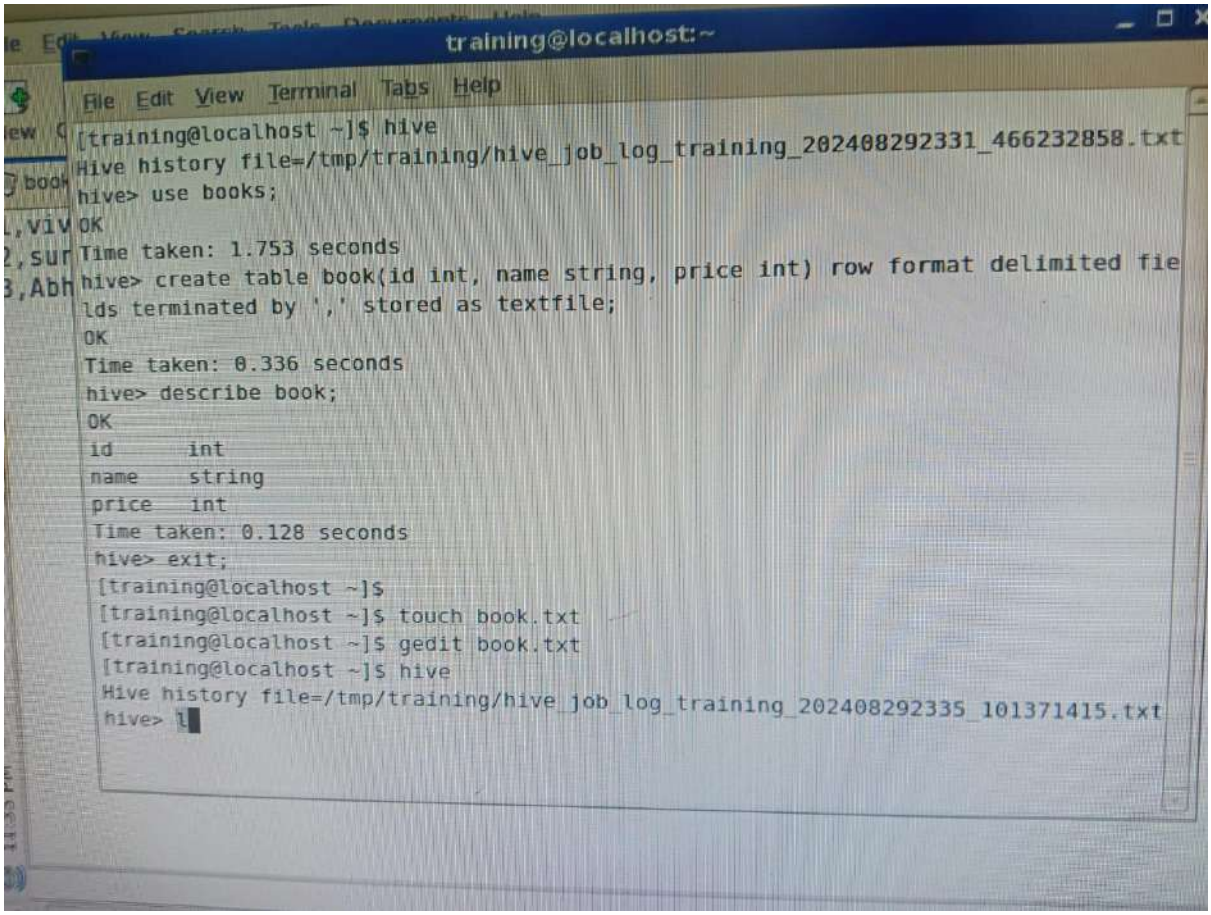
15. To exit from Hive

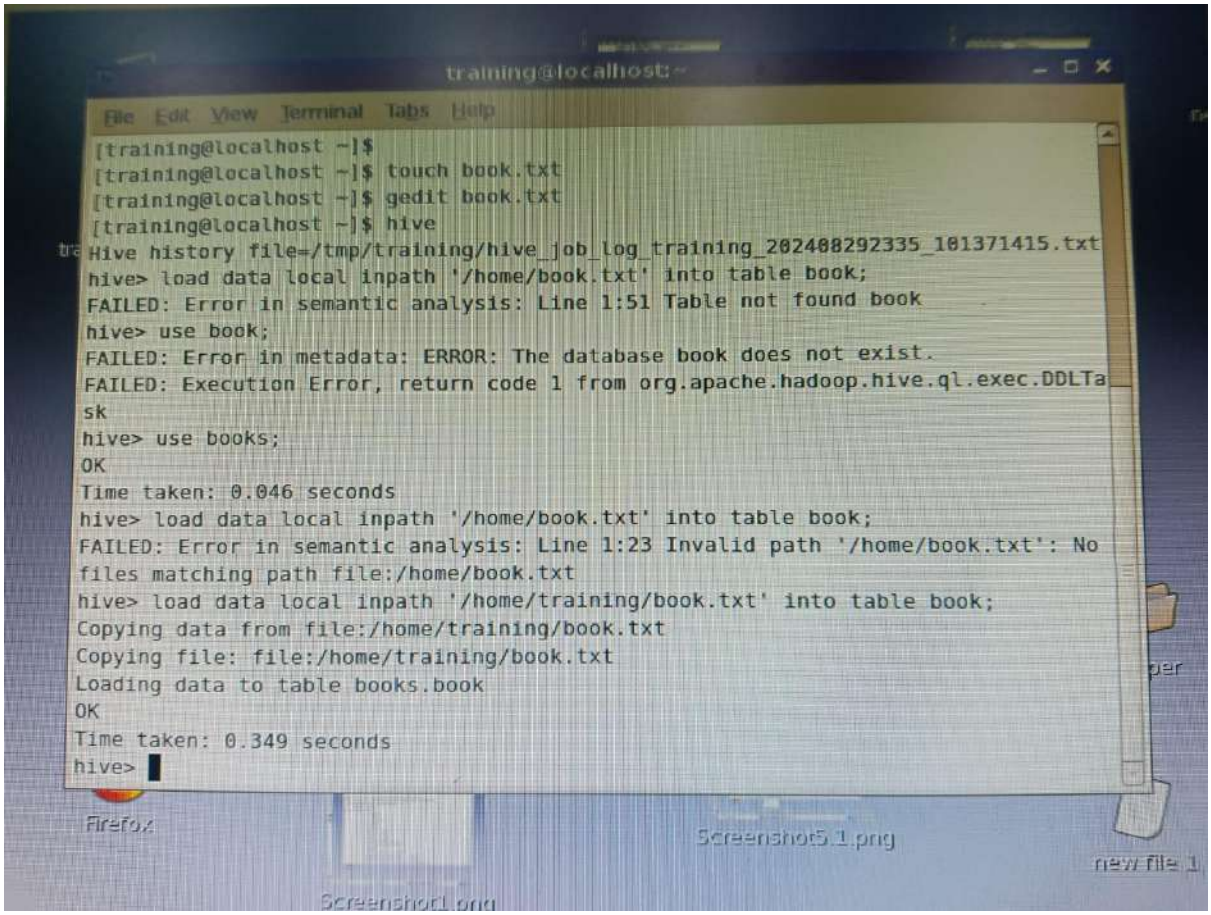
terminal hive> exit;

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

Output:


training@localhost:~
File Edit View Terminal Tabs Help
3 Abhi 1
Time taken: 0.154 seconds
hive> select max(price) from book;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
set mapred.reduce.tasks=<number>
Starting Job = job_202408292203_0002, Tracking URL = http://localhost:50030/jobd
etails.jsp?jobid=job_202408292203_0002
Kill Command = /usr/lib/hadoop/bin/hadoop job -Dmapred.job.tracker=localhost:80
21 -kill job_202408292203_0002
2024-08-29 23:38:05,415 Stage-1 map = 0%, reduce = 0%
2024-08-29 23:38:06,424 Stage-1 map = 100%, reduce = 0%
2024-08-29 23:38:13,540 Stage-1 map = 100%, reduce = 100%
Ended Job = job_202408292203_0002
OK
300
Time taken: 12.168 seconds
hive>
Firefox
Screenshot5.1.png


training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ hive
Hive history file=/tmp/training/hive_job_log_training_202408292331_466232858.txt
hive> use books;
OK
Time taken: 1.753 seconds
hive> create table book(id int, name string, price int) row format delimited fie
lds terminated by ',' stored as textfile;
OK
Time taken: 0.336 seconds
hive> describe book;
OK
id int
name string
price int
Time taken: 0.128 seconds
hive> exit;
[training@localhost ~]$
[training@localhost ~]$ touch book.txt
[training@localhost ~]$ gedit book.txt
[training@localhost ~]$ hive
Hive history file=/tmp/training/hive_job_log_training_202408292335_101371415.txt
hive>


training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$
[training@localhost ~]$ touch book.txt
[training@localhost ~]$ gedit book.txt
[training@localhost ~]$ hive
Hive history file=/tmp/training/hive_job_log_training_202408292335_101371415.txt
hive> load data local inpath '/home/book.txt' into table book;
FAILED: Error in semantic analysis: Line 1:51 Table not found book
hive> use book;
FAILED: Error in metadata: ERROR: The database book does not exist.
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTa
sk
hive> use books;
OK
Time taken: 0.046 seconds
hive> load data local inpath '/home/book.txt' into table book;
FAILED: Error in semantic analysis: Line 1:23 Invalid path '/home/book.txt': No
files matching path file:/home/book.txt
hive> load data local inpath '/home/training/book.txt' into table book;
Copying data from file:/home/training/book.txt
Copying file: file:/home/training/book.txt
Loading data to table books.book
OK
Time taken: 0.349 seconds
hive>
Firefox
Screenshot5_1.png
new file 1
Screenshot1.png

Experiment 7

Aim: Implement Bloom Filter using Python/R Programming.

Theory:

Bloom Filter is a data structure that can do this job. It is mainly a spaced optimized version of hashing where we may have false positives. The idea is to not store the actual key rather store only hash values. It is mainly a probabilistic and space optimized hashing where less than 10 bits per key are required for a 1% false positive probability and is not dependent on the size of individual keys.

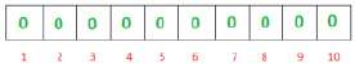
A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. For example, checking availability of username is set membership problem, where the set is the list of all registered username. The price we pay for efficiency is that it is probabilistic in nature that means, there might be some False Positive results. False positive means, it might tell that given username is already taken but actually it's not.

Properties of Bloom Filters:

Unlike a standard hash table, a Bloom filter of a fixed size can represent a set with an arbitrarily large number of elements.
Adding an element never fails. However, the false positive rate increases steadily as elements are added until all bits in the filter are set to 1, at which point all queries yield a positive result.
Bloom filters never generate false negative result, i.e., telling you that a username doesn't exist when it actually exists.
Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements. Example – if we delete "geeks" (in given example below) by clearing bit at 1, 4 and 7, we might end up deleting "nerd" also Because bit at index 4 becomes 0 and bloom filter claims that "nerd" is not present.

Working of Bloom Filter

A empty bloom filter is a bit array of m bits, all set to zero, like this –



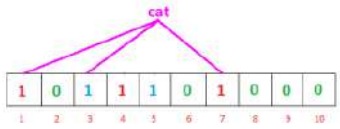
We need k number of hash functions to calculate the hashes for a given input. When we want to add an item in the filter, the bits at k indices h1(x), h2(x), … hk(x) are set, where indices are calculated using hash functions.
Example – Suppose we want to enter “geeks” in the filter, we are using 3 hash functions and a bit array of length 10, all set to 0 initially. First we’ll calculate the hashes as follows:

h1(“geeks”) % 10 = 1
h2(“geeks”) % 10 = 4
h3(“geeks”) % 10 = 7

False Positive in Bloom Filters

The question is why we said “probably present”, why this uncertainty. Let’s understand this with an example. Suppose we want to check whether “cat” is present or not. We’ll calculate hashes using h1, h2 and h3

h1(“cat”) % 10 = 1
h2(“cat”) % 10 = 3
h3(“cat”) % 10 = 7
If we check the bit array, bits at these indices are set to 1 but we know that “cat” was never added to the filter. Bit at index 1 and 7 was set when we added “geeks” and bit 3 was set we added “nerd”.



So, because bits at calculated indices are already set by some other item, bloom filter erroneously claims that “cat” is present and generating a false positive result. Depending on the application, it could be huge downside or relatively okay.
We can control the probability of getting a false positive by controlling the size of the Bloom filter. More space means fewer false positives. If we want to decrease probability of false positive result, we have to use more number of hash functions and larger bit array. This would add latency in addition to the item and checking membership.

Code:

```
import hashlib
import math
import random

def create_bloom_filter(size: int):
    """Create a new Bloom Filter."""
    bit_array = [False] * size  # Initialize bit array with False
    return bit_array

def _hashes(item: str, size: int, num_hashes: int):
    """Generate multiple hash values for the given item."""
    hash_values = []
    for i in range(num_hashes):
        # Generate a hash and mod it by the size of the bit array
        hash_value = int(hashlib.md5(item.encode()).hexdigest(), 16) + i
        hash_values.append(hash_value % size)
    return hash_values

def add(bloom_filter, item: str):
    """Add an item to the Bloom Filter."""
    for hash_value in _hashes(item, len(bloom_filter), num_hashes):
        bloom_filter[hash_value] = True  # Set the corresponding bits to True

def contains(bloom_filter, item: str) -> str:
    """Check if an item is in the Bloom Filter."""
    # Check if all bits at hashed positions are True
    if all(bloom_filter[hash_value] for hash_value in _hashes(item, len(bloom_filter),
num_hashes)):
        # Randomly simulate a false positive scenario
        if random.random() < false_positive_probability:
            return f"'{item}' is a false positive!"
        return f"'{item}' is probably present!"
    return f"'{item}' is definitely not present!"

# Parameters
size = 124
num_hashes = 4
false_positive_probability = 0.05

# Create a Bloom Filter
bloom_filter = create_bloom_filter(size)
```

```
# Adding items to the Bloom Filter
items_to_add = ["abundant", "bloom", "coherent", "cohesive", "bonus", "abounds",
"genial", "generosity", "abundance"]
for item in items_to_add:
    add(bloom_filter, item)

# Items to check
items_to_check = ["war", "gloomy", "humanity", "abundant", "bloom", "coherent",
                  "cohesive", "bluff", "bolster", "hate", "racism", "bonus",
                  "abounds", "genial", "geeksforgeeks", "nuke", "hurt",
                  "twitter", "cheater", "facebook"]

# Check membership and print results
for item in items_to_check:
    print(contains(bloom_filter, item))
```

Output:

```
'war' is definitely not present!
'gloomy' is definitely not present!
'humanity' is definitely not present!
'abundant' is probably present!
'bloom' is probably present!
'coherent' is probably present!
'cohesive' is probably present!
'bluff' is definitely not present!
'bolster' is definitely not present!
'hate' is definitely not present!
'racism' is definitely not present!
'bonus' is probably present!
'abounds' is probably present!
'genial' is probably present!
'geeksforgeeks' is definitely not present!
'nuke' is definitely not present!
'hurt' is definitely not present!
'twitter' is definitely not present!
'cheater' is definitely not present!
'facebook' is definitely not present!


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Experiment 8

Aim: Implement FM algorithm using Python/R Programming.

Theory:

The Flajolet-Martin algorithm is also known as probabilistic algorithm which is mainly used to count the number of unique elements in a stream or database

The basic idea to which Flajolet-Martin algorithm is based on is to use a hash function to map the elements in the given dataset to a binary string, and to make use of the length of the longest null sequence in the binary string as an estimator for the number of unique elements to use as a value element.

The steps for the Flajolet-Martin algorithm are:

- First step is to choose a hash function that can be used to map the elements in the database to fixed-length binary strings. The length of the binary string can be chosen based on the accuracy desired.
- Next step is to apply the hash function to each data item in the dataset to get its binary string representation.
- Next step includes determinig the position of the rightmost zero in each binary string.
- Next we compute the maximum position of the rightmost zero for all binary strings.
- Now we estimate the number of distinct elements in the dataset as 2 to the power of the maximum position of the rightmost zero which we calculated in previous step.

The accuracy of Flajolet Martin Algorithm is determined by the length of the binary strings and the number of hash functions it uses. Generally, with increse in the length of the binary strings or using more hash functions in algorithm can often increase the algorithm’s accuracy.

The Flajolet Martin Algorithm is especially used for big datasets that cannot be kept in memory or analysed with regular methods. This algorithm , by using good probabilistic techniques, can provide a precise estimate of the number of unique elements in the data set by using less computing.

Code:

```
from collections import defaultdict

def create_fp_tree(transactions, min_support):
    """Create an FP-tree from transactions."""
    # Count item frequencies
    item_count = defaultdict(int)
    for transaction in transactions:
        for item in transaction:
            item_count[item] += 1

    # Filter out items that do not meet the minimum support
    item_count = {item: count for item, count in item_count.items() if count >=
min_support}

    # Create the FP-tree structure
    fp_tree = {}
    header_table = {item: [] for item in item_count}  # To hold links to the nodes

    for transaction in transactions:
        # Filter and sort items in the transaction by frequency
        filtered_items = [item for item in transaction if item in item_count]
        filtered_items.sort(key=lambda x: item_count[x], reverse=True)

        if not filtered_items:
            continue

        current_node = fp_tree  # Start at the root of the FP-tree
        for item in filtered_items:
            if item not in current_node:
                current_node[item] = {'count': 1, 'children': {}}
                header_table[item].append(current_node[item])  # Link to the header table
            else:
                current_node[item]['count'] += 1  # Update the count

            current_node = current_node[item]['children']  # Move to the child node

    return fp_tree, header_table

def find_frequent_patterns(fp_tree, header_table, min_support):
    """Extract frequent patterns from the FP-tree."""
    patterns = {}
```

```
    for item, nodes in header_table.items():
        if nodes:  # Only process if there are nodes for the item
            count = sum(node['count'] for node in nodes)
            if count >= min_support:
                patterns[item] = count

    return patterns

def fp_growth(transactions, min_support):
    """Main function to perform the FP-Growth algorithm."""
    # Step 1: Create the FP-tree
    fp_tree, header_table = create_fp_tree(transactions, min_support)

    # Step 2: Find frequent patterns
    return find_frequent_patterns(fp_tree, header_table, min_support)

# Example usage
if __name__ == "__main__":
    transactions = [
        ['milk', 'bread', 'diaper'],
        ['milk', 'diaper', 'beer', 'bread'],
        ['milk', 'bread'],
        ['diaper', 'beer'],
        ['milk', 'diaper', 'bread']
    ]

    min_support = 2
    frequent_patterns = fp_growth(transactions, min_support)

    print("Frequent Patterns:")
    for item, count in frequent_patterns.items():
        print(f"Item: {item}, Count: {count}")
```

Output:

```
Frequent Patterns:
Item: milk, Count: 4
Item: bread, Count: 4
Item: diaper, Count: 4
Item: beer, Count: 2


** Process exited - Return Code: 0 **
```

Experiment 9

Aim: Data Visualisation using R

Theory:

Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

Data Visualization in R Programming Language:

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts, Infogram, and Kibana. The various data visualization platforms have different capabilities, functionality, and use cases. They also require a different skill set. This article discusses the use of R for data visualization.

R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

Types of Data Visualizations
Some of the various types of visualizations offered by R are:

Bar Plot

There are two types of bar plots- horizontal and vertical which represent data points as horizontal or vertical bars of certain lengths proportional to the value of the data item. They are generally used for continuous and categorical variable plotting. By setting the horiz parameter to true and false, we can get horizontal and vertical bar plots respectively

Histogram

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
sns.set_theme()
sns.set(rc = {"figure.figsize":(10,6), "figure.dpi":300})

!pip install country_converter -q
import country_converter as coco


df=pd.read_csv('/kaggle/input/data-science-salaries-2023/ds_salaries.csv')
df.head()
```

	work_year	experience_level	employment_type	job_title	salary	salary_curren
0	2023	SE	FT	Principal Data Scientist	80000	EU
1	2023	MI	CT	ML Engineer	30000	US
2	2023	MI	CT	ML Engineer	25500	US
3	2023	SE	FT	Data Scientist	175000	US
4	2023	SE	FT	Data Scientist	120000	US

```
print("Number of rows and columns in the dataset:",df.shape)
```

```
Number of rows and columns in the dataset: (3755, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           3755 non-null   int64 
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64 
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64 
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64 
```

```
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
```

```
print("Number of missing data in the dataset:",df.isnull().sum().sum())
```

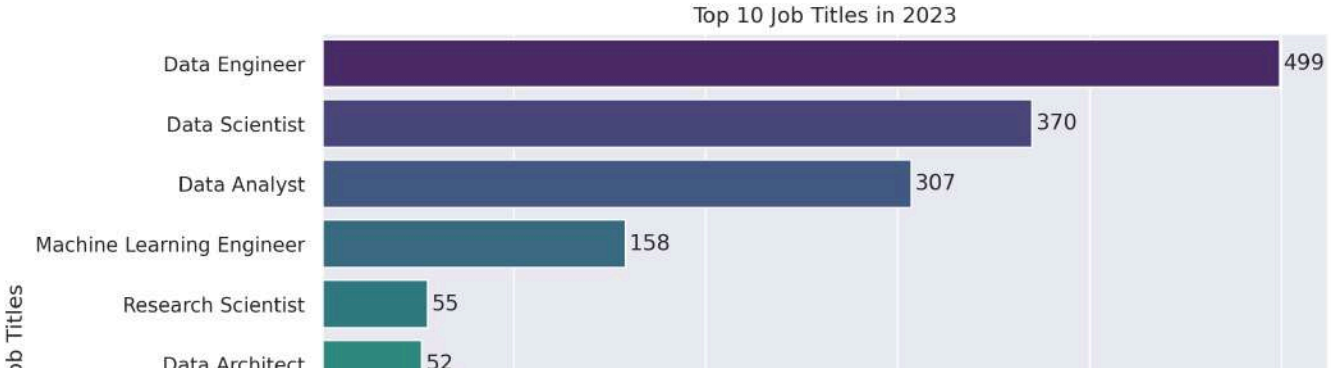
```
Number of missing data in the dataset: 0
```

```
print("Number of unique values in columns:\n\n", df.nunique())
```

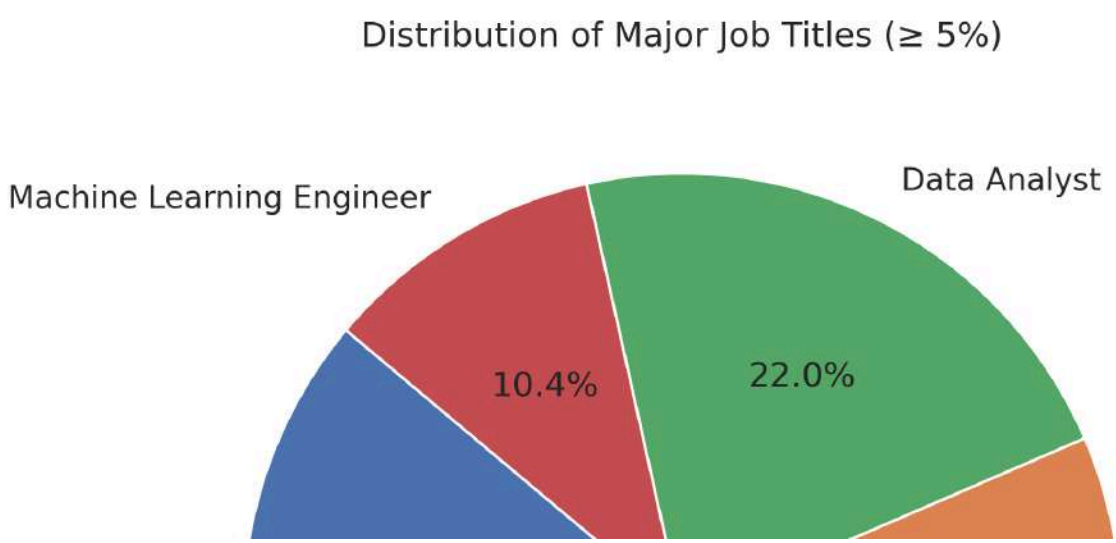
```
Number of unique values in columns:

 work_year                4
experience_level          4
employment_type           4
job_title                93
salary                  815
salary_currency          20
salary_in_usd          1035
employee_residence       78
remote_ratio              3
company_location         72
company_size              3
dtype: int64
```

```
jobs = df[df['work_year'] == 2023]['job_title'].value_counts().nlargest(10).reset_index()
jobs.columns = ['Job Title', 'Count']

fig, ax = plt.subplots()
sns.barplot(ax=ax, data=jobs, y='Job Title', x='Count', palette='viridis')
ax.set(ylabel='Job Titles', xlabel='Counts', title='Top 10 Job Titles in 2023')

for container in ax.containers:
    ax.bar_label(container, padding=2)

plt.show()
```

```
<ipython-input-86-d7846ccb267c>:5: FutureWarning:

    Passing `palette` without assigning `hue` is deprecated and will be removed in v
```




```
import pandas as pd
import matplotlib.pyplot as plt


job_title_counts = df['job_title'].value_counts()

job_title_percentages = job_title_counts / job_title_counts.sum() * 100

major_job_titles = job_title_percentages[job_title_percentages >= threshold_percentage]

remaining_count = job_title_counts[job_title_percentages < threshold_percentage].sum()

plt.figure(figsize=(10, 7))
plt.pie(major_job_titles, labels=major_job_titles.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Major Job Titles (≥ 5%)')
plt.show()
```



As you can see, there are the most data engineers, followed by data scientists.

First, let's look at the unique values in the `experience_level` column.

```
df['experience_level'].unique()
```

```
array(['SE', 'MI', 'EN', 'EX'], dtype=object)
```

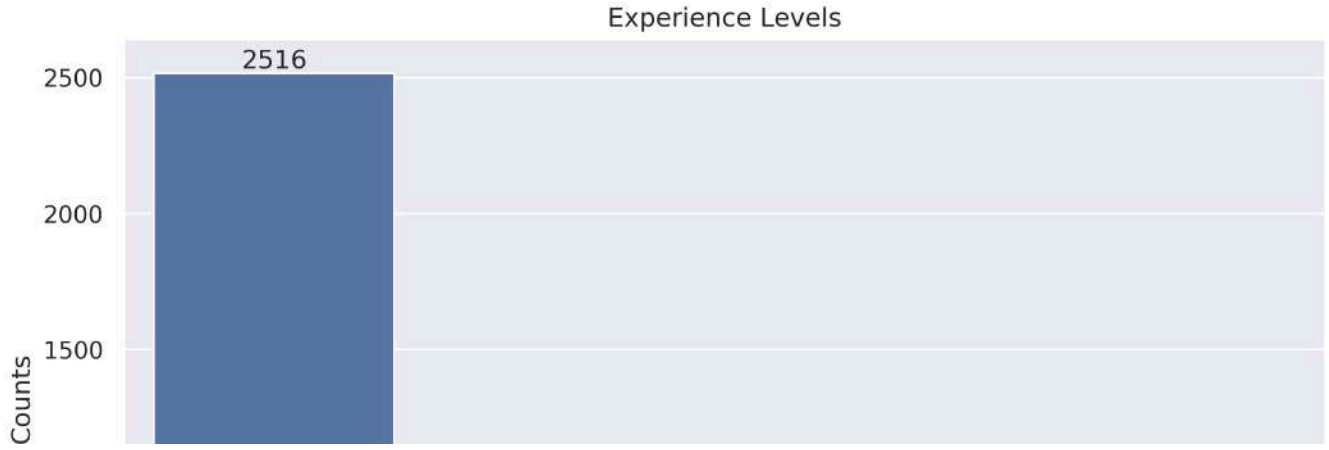
As you can see, there are 4 unique values which are SE(Senior level/expert) , MI(medium level/intermediate) , EN (Entry level) and EX(Executive level). Lets rename these values with the `rename` method.

```
df['experience_level'] = df['experience_level'].replace('EN','Entry-level/Junior')
df['experience_level'] = df['experience_level'].replace('MI','Mid-level/Intermediate')
df['experience_level'] = df['experience_level'].replace('SE','Senior-level/Expert')
df['experience_level'] = df['experience_level'].replace('EX','Executive-level/Director')
```

Let's draw a bar plot of experience levels.

```
fig, ax = plt.subplots()
sns.countplot(ax = ax, data = df, x = df.experience_level)
ax.set(xlabel='', ylabel='Counts', title='Experience Levels')
ax.bar_label(ax.containers[0])
```

```
[Text(0, 0, '2516'), Text(0, 0, '805'), Text(0, 0, '320'), Text(0, 0, '114')]
```



As you can see, the senior-level positions have the highest count, followed by mid-level and junior positions. There are fewer director-level positions compared to other levels.

Now, let's look at the unique values in the employment_type column.

```
df['employment_type'].unique()
```

```
array(['FT', 'CT', 'FL', 'PT'], dtype=object)
```

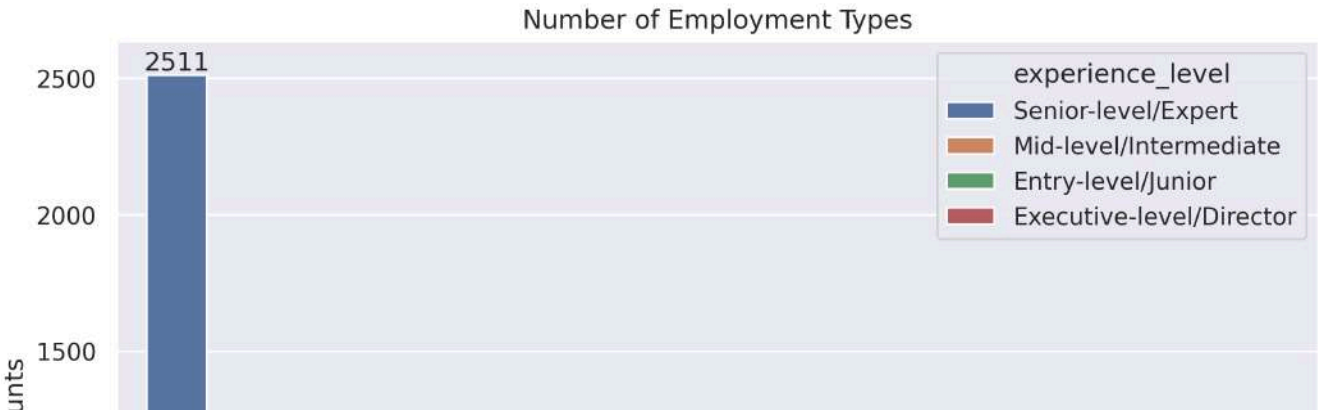
As you can see, there are 4 unique values which are FT(Full-Time), PT(Part-Time), CT(Contract) and FL(Freelance). Lets rename these values with the rename method.

```
df['employment_type'] = df['employment_type'].replace('FT','Full-Time')
df['employment_type'] = df['employment_type'].replace('PT','Part-Time')
df['employment_type'] = df['employment_type'].replace('CT','Contract')
df['employment_type'] = df['employment_type'].replace('FL','Freelance')
```

Let's draw a bar plot of experience types.

```
fig, ax = plt.subplots()
sns.countplot(ax = ax, data = df, x = df.employment_type, hue = 'experience_level')
ax.set(xlabel='', ylabel='Counts', title='Number of Employment Types')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
ax.bar_label(ax.containers[2])
ax.bar_label(ax.containers[3])
```

```
[Text(0, 0, '113'), Text(0, 0, '1')]
```

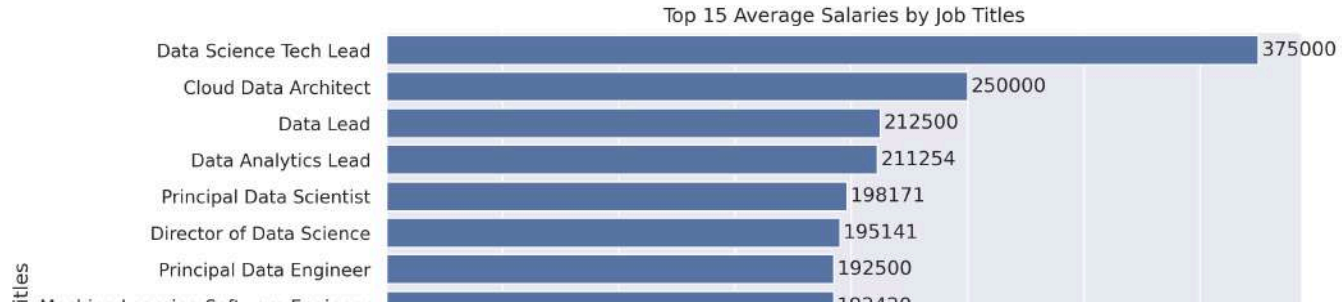


As you can see, a considerable number of people are employed here on a full-time basis. Among the full-time employees, the majority of them are senior. We observe that freelancing is less prevalent these days

```
job_title_salary = df['salary_in_usd'].groupby(df['job_title']).mean().round(0).nlargest(15
plt.figure(figsize=(25,9))
fig, ax = plt.subplots()
ax = sns.barplot(ax = ax, data = job_title_salary , y = job_title_salary.job_title, x = job
```

```
ax.set(ylabel='Job titles',xlabel='Salary in usd', title='Top 15 Average Salaries by Job Ti
ax.bar_label(ax.containers[0], padding = 2)
```

```
[Text(2, 0, '375000'),
 Text(2, 0, '250000'),
 Text(2, 0, '212500'),
 Text(2, 0, '211254'),
 Text(2, 0, '198171'),
 Text(2, 0, '195141'),
 Text(2, 0, '192500'),
 Text(2, 0, '192420'),
 Text(2, 0, '191279'),
 Text(2, 0, '190264'),
 Text(2, 0, '190000'),
 Text(2, 0, '183858'),
 Text(2, 0, '175052'),
 Text(2, 0, '174150'),
 Text(2, 0, '163220')]
<Figure size 7500x2700 with 0 Axes>
```



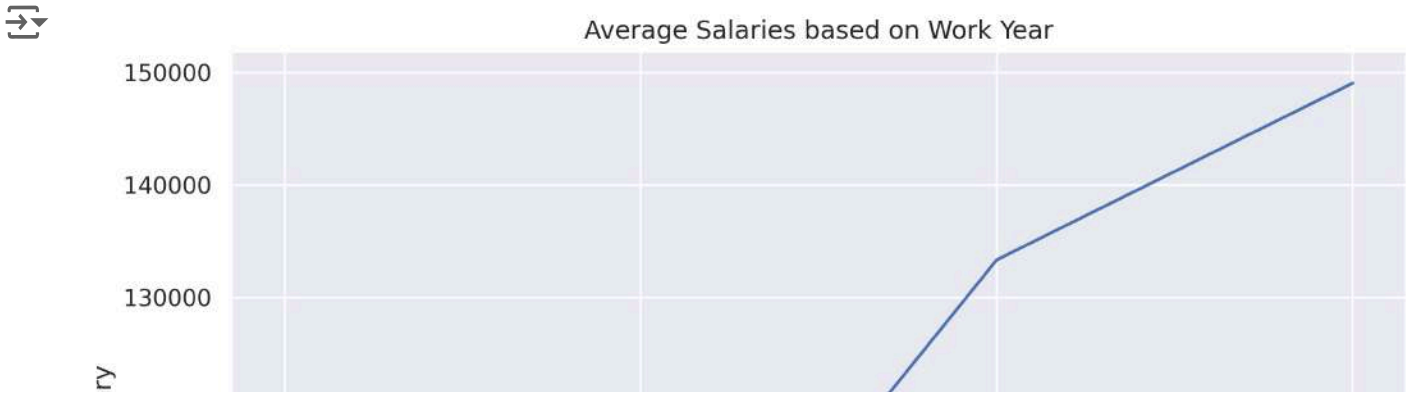
As we expected, the average salaries of those generally employed at the executive level are higher. Due to the trend of cloud computing, cloud data architect is the second highest paid profession.

```
avg_salaries = df.groupby('employment_type')['salary_in_usd'].mean().round(0).sort_values(a
fig, ax = plt.subplots()
sns.barplot(ax =ax,data = df , x = 'employment_type', y = 'salary_in_usd',errorbar = None,
ax.set(xlabel='', ylabel='Dollars', title='Average Salaries in Dollars Per Year')
ax.bar_label(ax.containers[3], padding = 2)
```

```
[Text(0, 2, '149367'),
 Text(0, 2, '27750'),
 Text(0, 2, '50000'),
 Text(0, 2, '17779')]
```

As you can see, average salaries for full-time have increased over the years. It shows that companies care about data science. The second-highest salaries on the plot belong to freelancers, which is a clear indication of the growing trend in freelance work.

```
year_based_salary=df['salary_in_usd'].groupby(df['work_year']).mean()
plt.title("Average Salaries based on Work Year")
plt.xlabel('Work Year')
plt.ylabel('Salary')
sns.lineplot(x=['2020', '2021', '2022','2023'],y=year_based_salary)
plt.show()
```



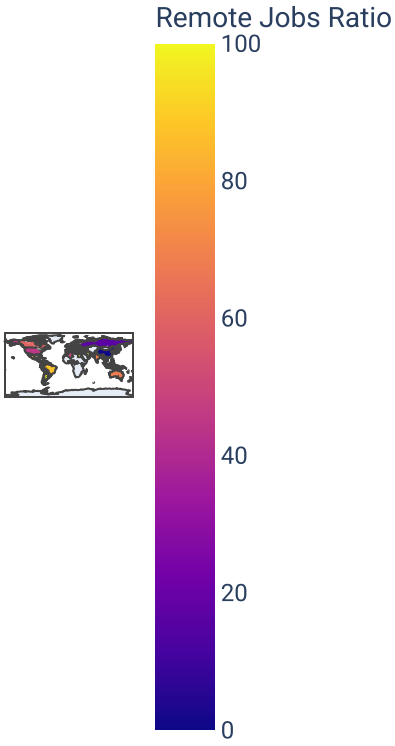
As you can see, the average salary for data-driven jobs is increasing every year, with a particularly significant jump observed between 2021 and 2022. This trend underscores the growing demand for skilled professionals in this field.

```
rr = df.groupby('company_location')['remote_ratio'].mean().reset_index()
rr['company_location'] =  coco.convert(names = rr['company_location'], to = "ISO3")
rr.head()
```

	company_location	remote_ratio
0	ARE	66.666667
1	ALB	50.000000
2	ARM	0.000000
3	ARG	100.000000
4	ASM	66.666667

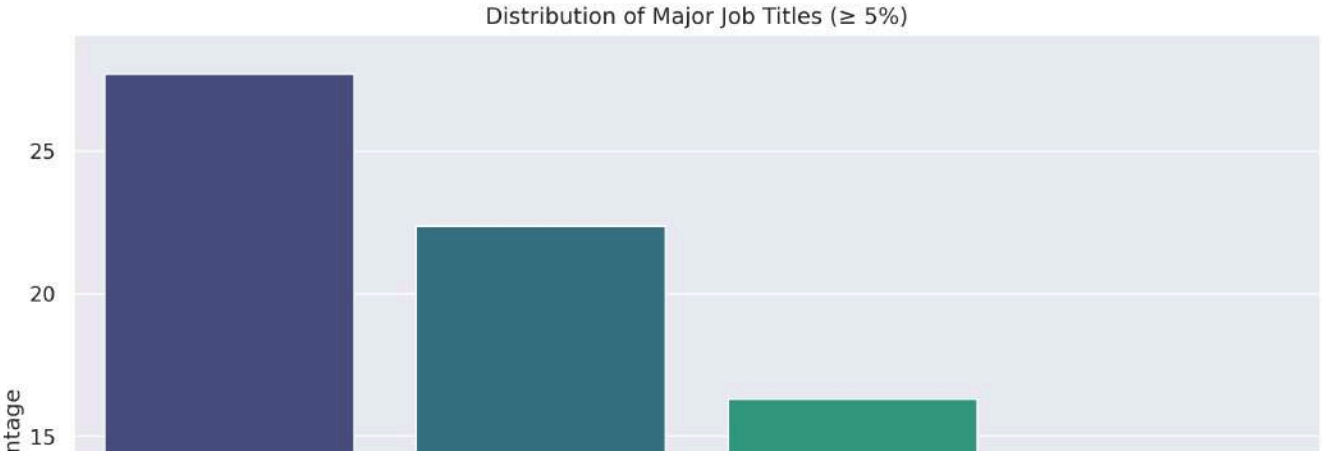
```
fig = px.choropleth(rr,
                    locations = rr.company_location,
                    color = rr.remote_ratio,
                    labels={'company_location':'Country','remote_ratio':'Remote Jobs Ratio'

fig.update_layout(title = "Remote Jobs Locations")
fig.show()
```

Remote Jobs Locations




```
job_title_counts = df['job_title'].value_counts()

job_title_percentages = job_title_counts / job_title_counts.sum() * 100

major_job_titles = job_title_percentages[job_title_percentages >= threshold_percentage]

histogram_data = pd.DataFrame({
    'Job Title': major_job_titles.index,
    'Percentage': major_job_titles.values
})

plt.figure(figsize=(12, 8))
sns.barplot(data=histogram_data, x='Job Title', y='Percentage', palette='viridis')
plt.title('Distribution of Major Job Titles (≥ 5%)')
plt.xlabel('Job Title')
plt.ylabel('Percentage')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-99-f8359f18730d>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
```



Thadomal Shahani Engineering College, Bandra, Mumbai
Department of Computer Engineering
Big Data Analytics (Mini Project) SEM VII

Report on Mini Project

Subject: Big Data Analytics (CSC702)

AY: 2024-25

IPL DATA ANALYSIS

Kanav Bhatia: 2103020

Parth Dabholkar: 2103032

Shirish Shetty: 2103164

Vedant Devkar: 2103034

Guided By

Ms. Sonali Jadhav

CHAPTER 1: INTRODUCTION

Analyzing IPL (Indian Premier League) data is crucial for gaining insights into team performance, player statistics, and overall game dynamics. By building a data pipeline and using Apache Spark on Databricks, you can handle vast amounts of data efficiently and perform real-time transformations. This type of analysis allows teams, analysts, and fans to understand patterns, make data-driven decisions, and even predict future outcomes. Additionally, IPL data analysis can help identify key performance indicators, strategize for upcoming matches, and enhance fan engagement through deeper insights.

This project aims to delve into the depths of IPL's data ecosystem, leveraging Big Data engineering techniques to extract valuable insights. By constructing a comprehensive pipeline from data ingestion to visualization, we aim to unravel patterns, trends, and user behaviors within IPL's vast score catalog.

Utilizing Amazon Web Services (AWS) cloud infrastructure, this project employs cutting-edge technologies such as Spark Streaming for real-time data streaming, Apache SPARK for data transformation, and Databricks Platform for querying and analysis. Through the seamless integration of these tools, we unlock the potential of IPL scores data, empowering decision-makers with actionable insights.

From the initial ingestion of raw data to the creation of a refined and integrated data lake, this project showcases the power of Big Data engineering in unraveling the complexities of the modern Cricket culture's landscape. Join us on this journey as we explore the rhythms and melodies hidden within IPL's vast sea of data, transforming information into intelligence, and insights into innovation.

CHAPTER 2: DATA DESCRIPTION AND ANALYSIS

This data set has the ball-by-ball data of all the Indian Premier League (IPL) matches till 2017 season.

Source: http://cricsheet.org/

Research scope: Predicting the winner of the next season of IPL based on past data, Visualizations, Perspectives, etc.

The dataset utilized in this project encompasses 5 primary CSV files: Ball_by_Ball.csv, Player_Match.csv, Match.csv, Player.csv, Team.csv. Each dataset contributes unique attributes crucial for understanding the IPL ecosystem comprehensively.

- Ball_by_Ball.csv: Info about each and every ball.
- Player_Match.csv: Info about players in each match
- Match.csv: Info about the matches played
- Player.csv: Info about each player from each team.
- Team.csv: Info about the teams that participated in the IPL.

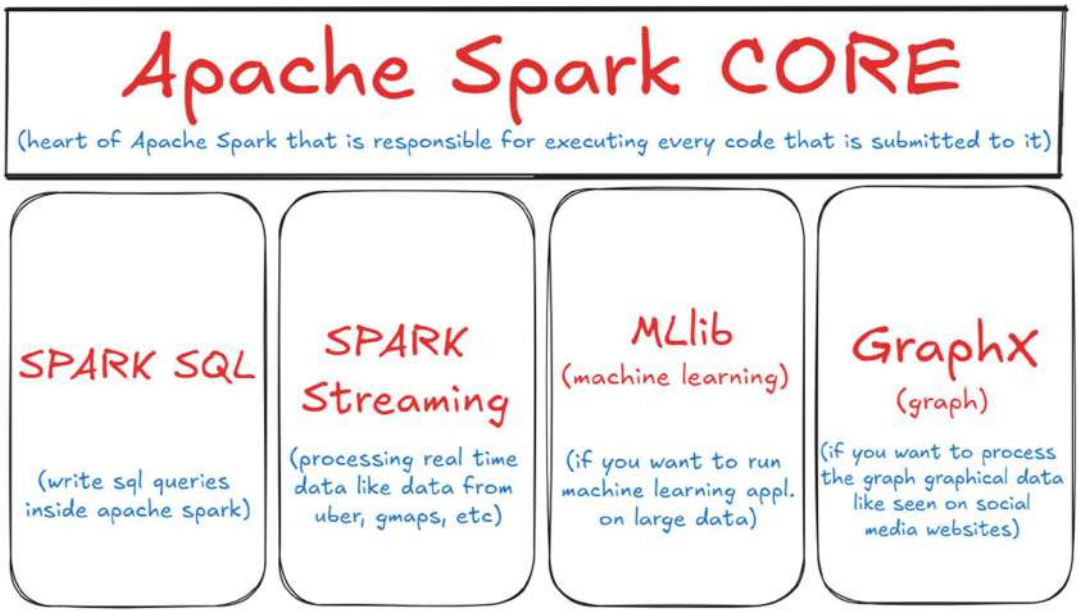
Analysis:

By joining the all these datasets, we can gain insights intovarious aspects of the Spotify ecosystem.

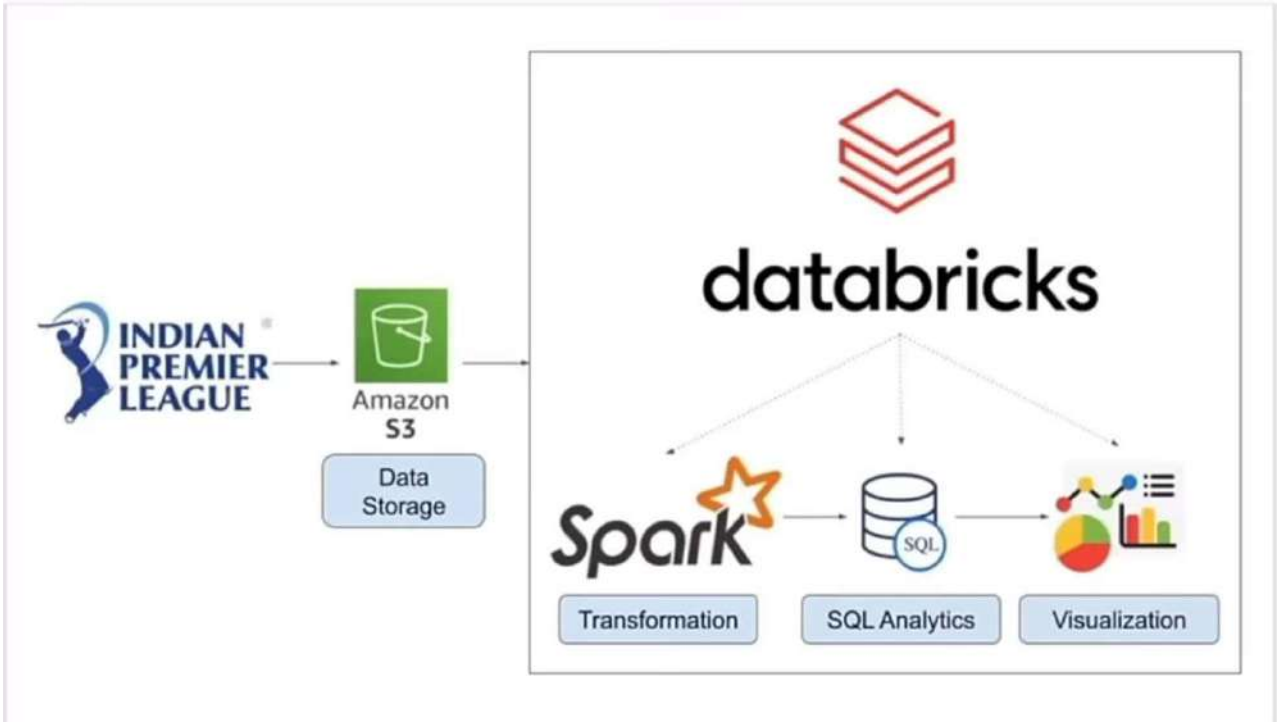
Analysis can include:

- Filter to include only valid deliveries (excluding extras like wides and no balls for specific analyses)
- Aggregation: Calculate the total and average runs scored in each match and inning
- Window Function: Calculate running total of runs in each match for each over
- Exploration of temporal trends in album releases and track popularity over time.
- Conditional Column: Flag for high impact balls (either a wicket or more than 6 runs including extras).

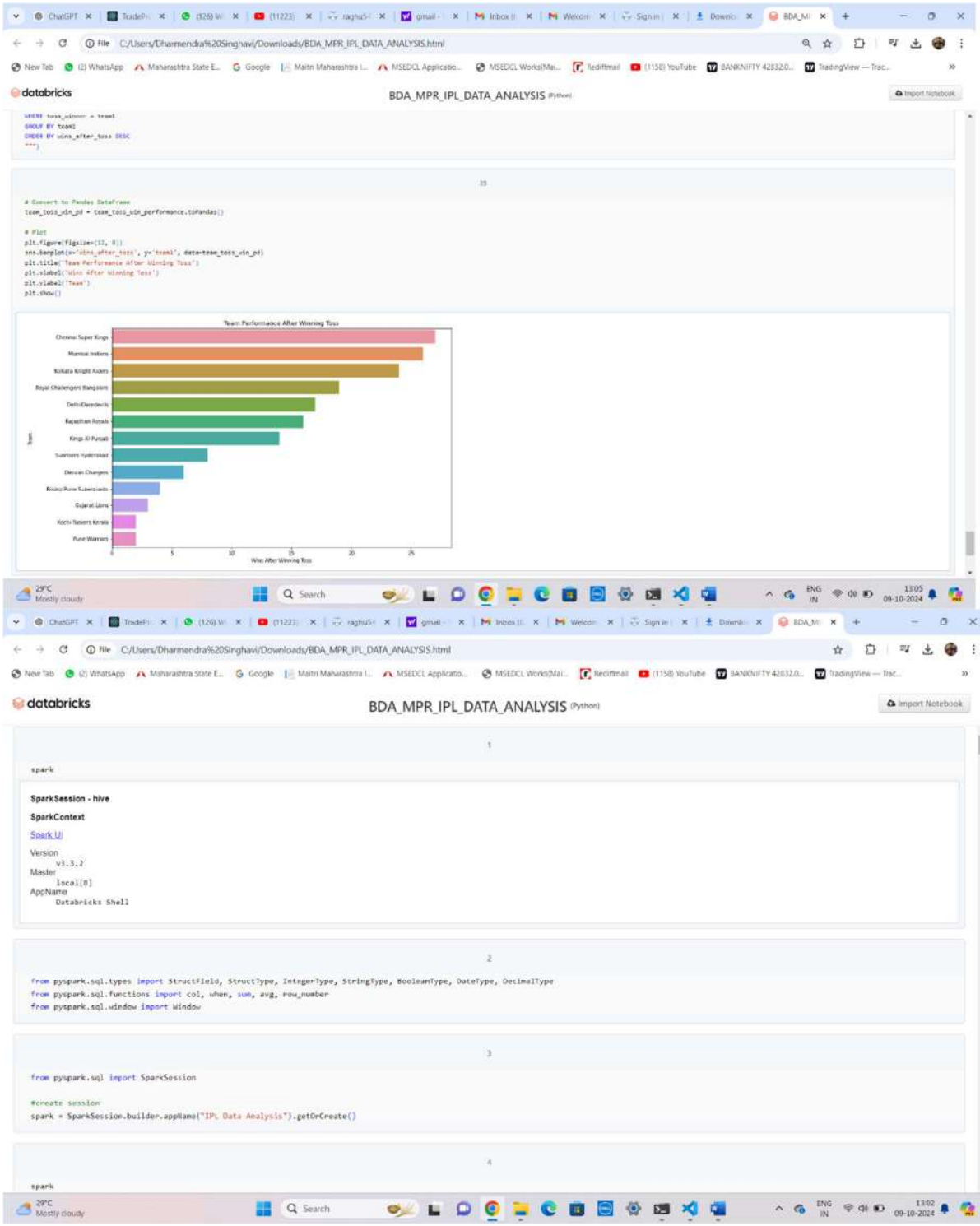
Through comprehensive querying and visualization techniques building a data pipeline and using Apache Spark on Databricks, you can handle vast amounts of data efficiently and perform real-time transformations.

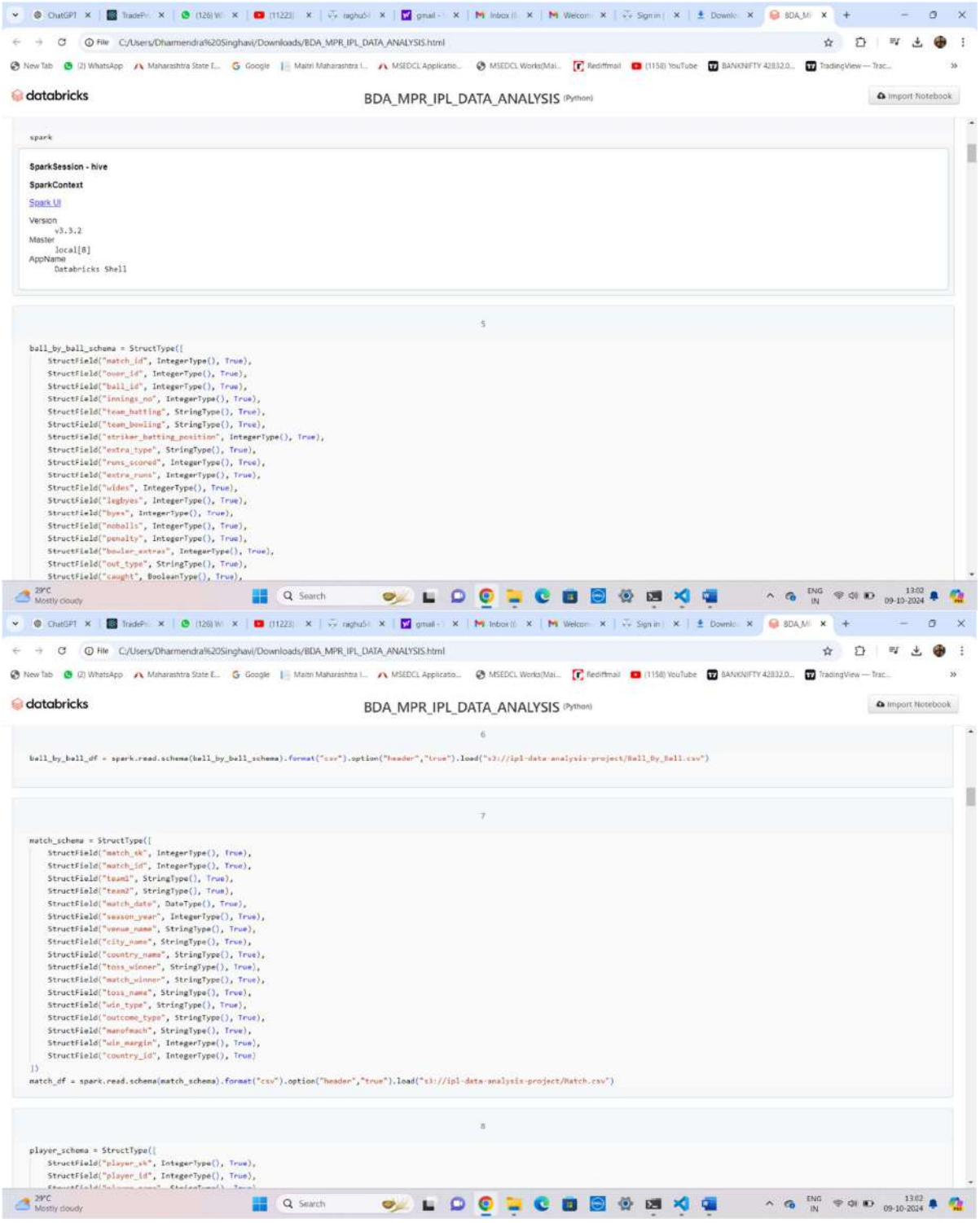


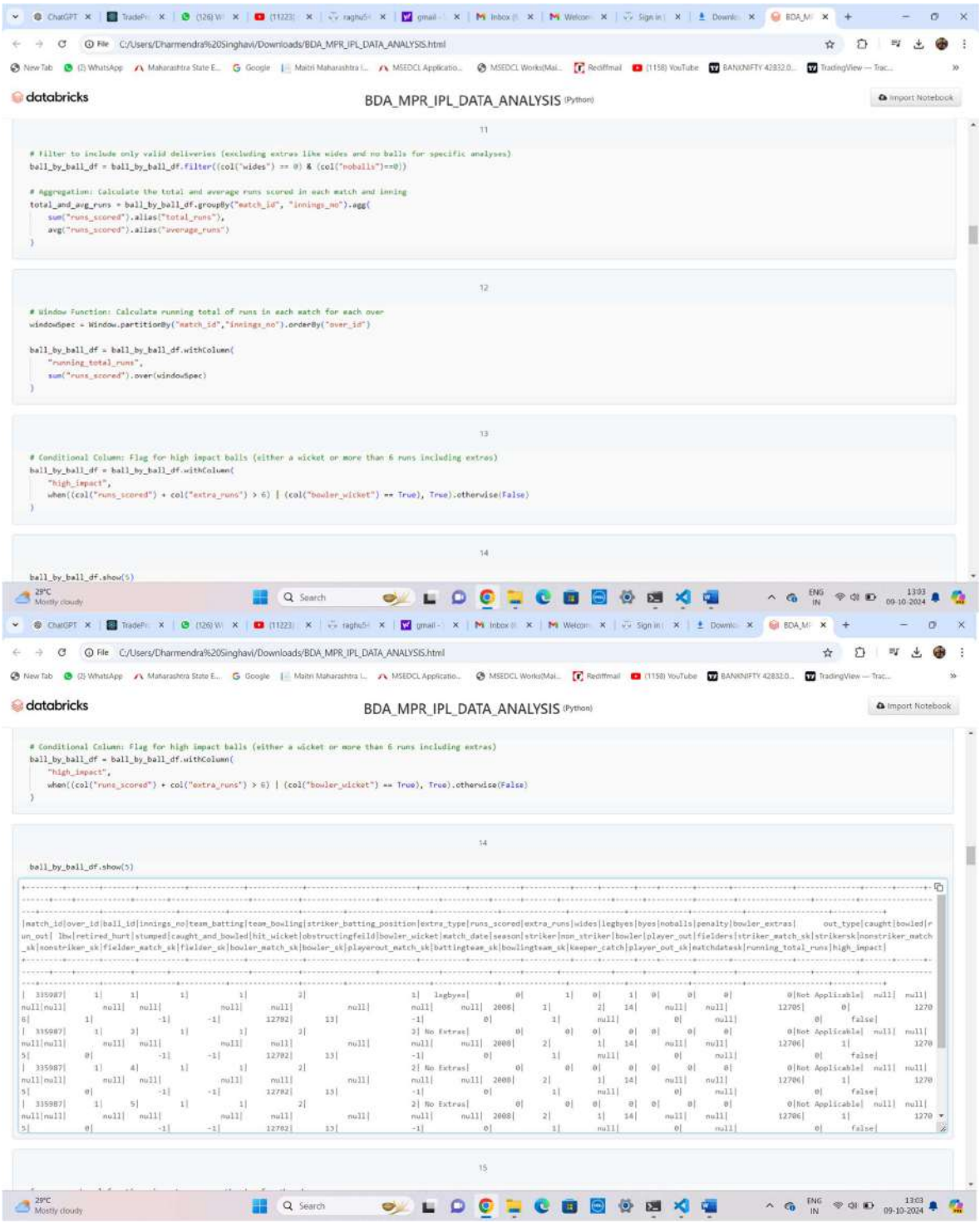
CHAPTER 3: DESIGN OF DATA PIPELINE

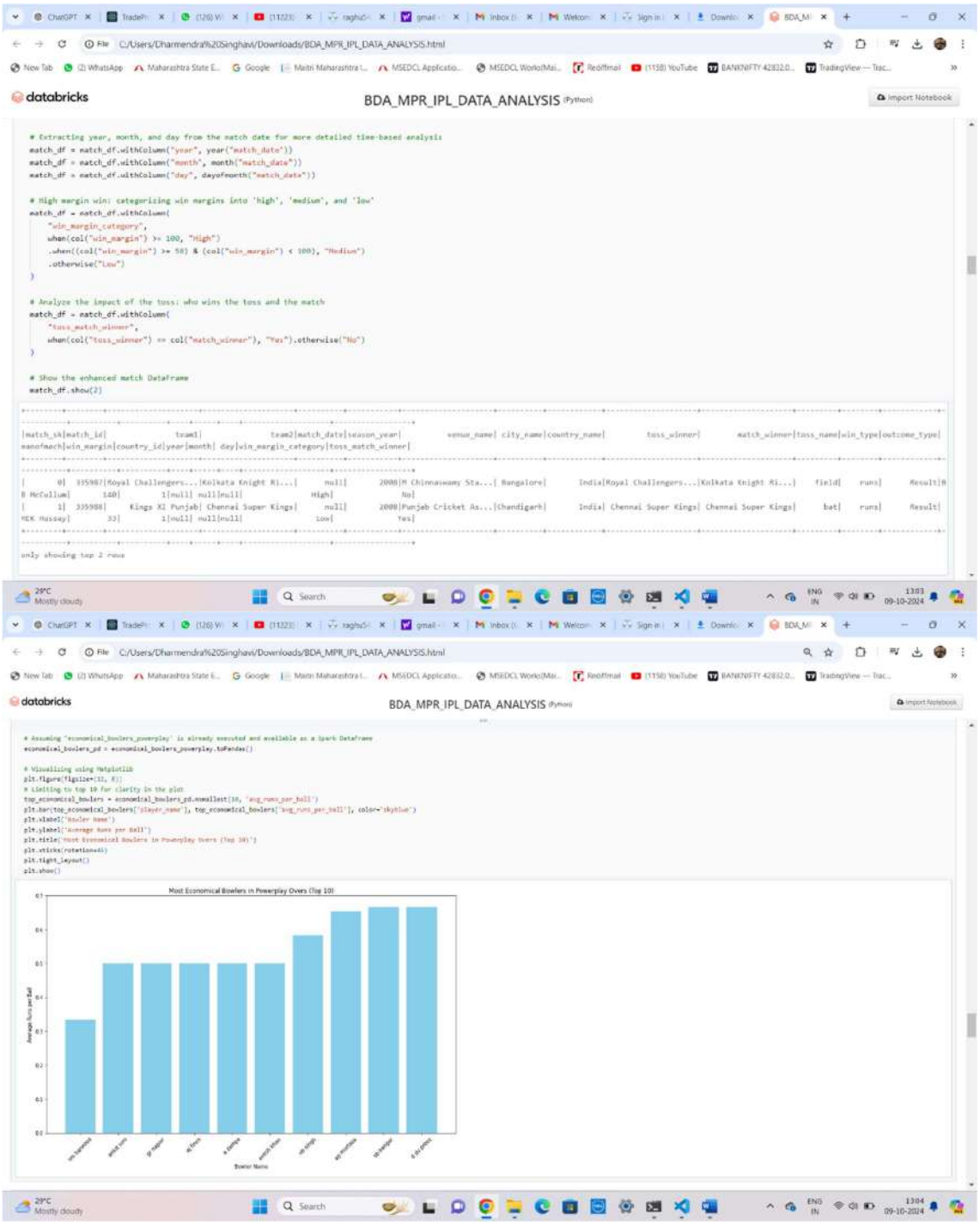


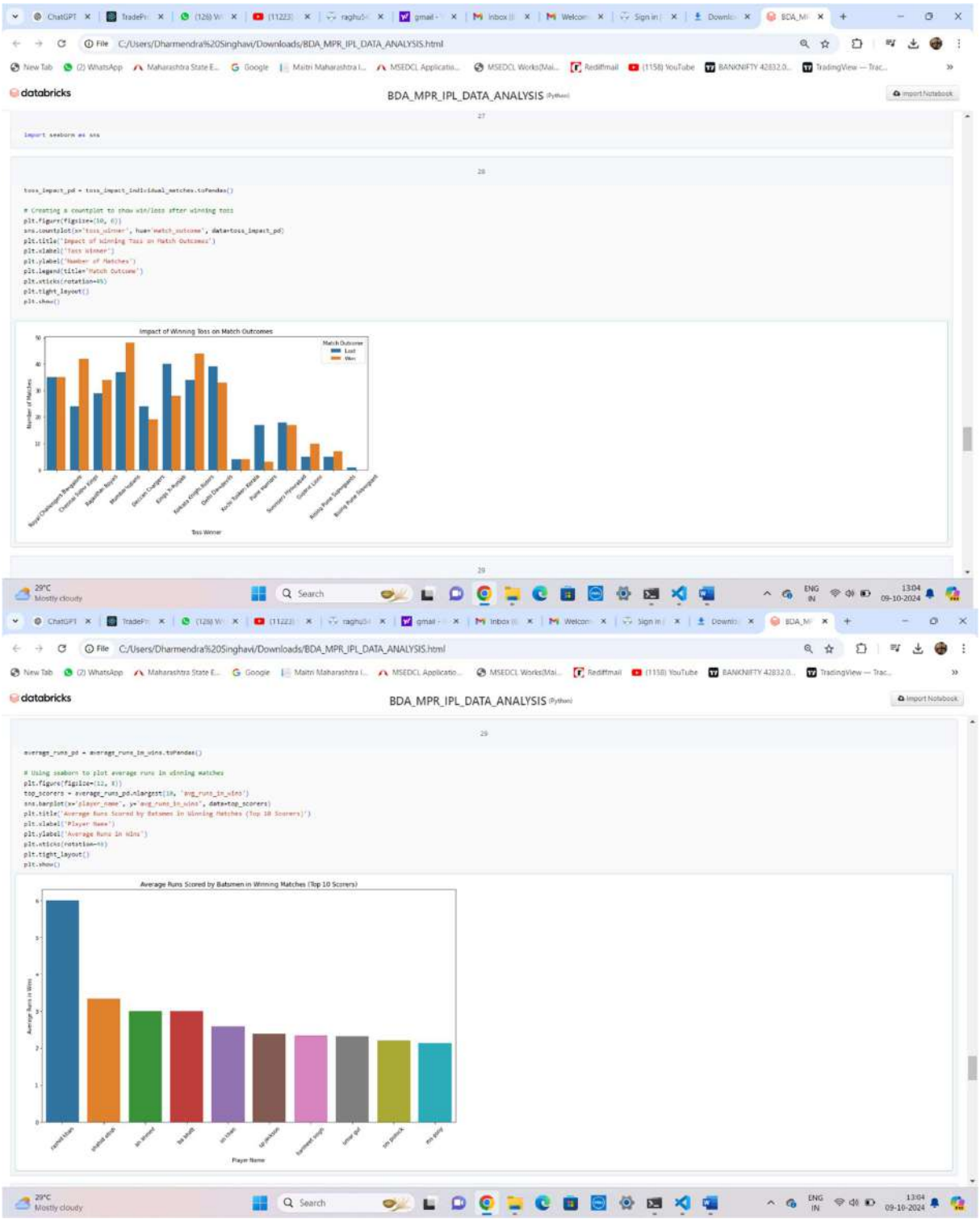
CHAPTER 4: RESULT ANALYSIS

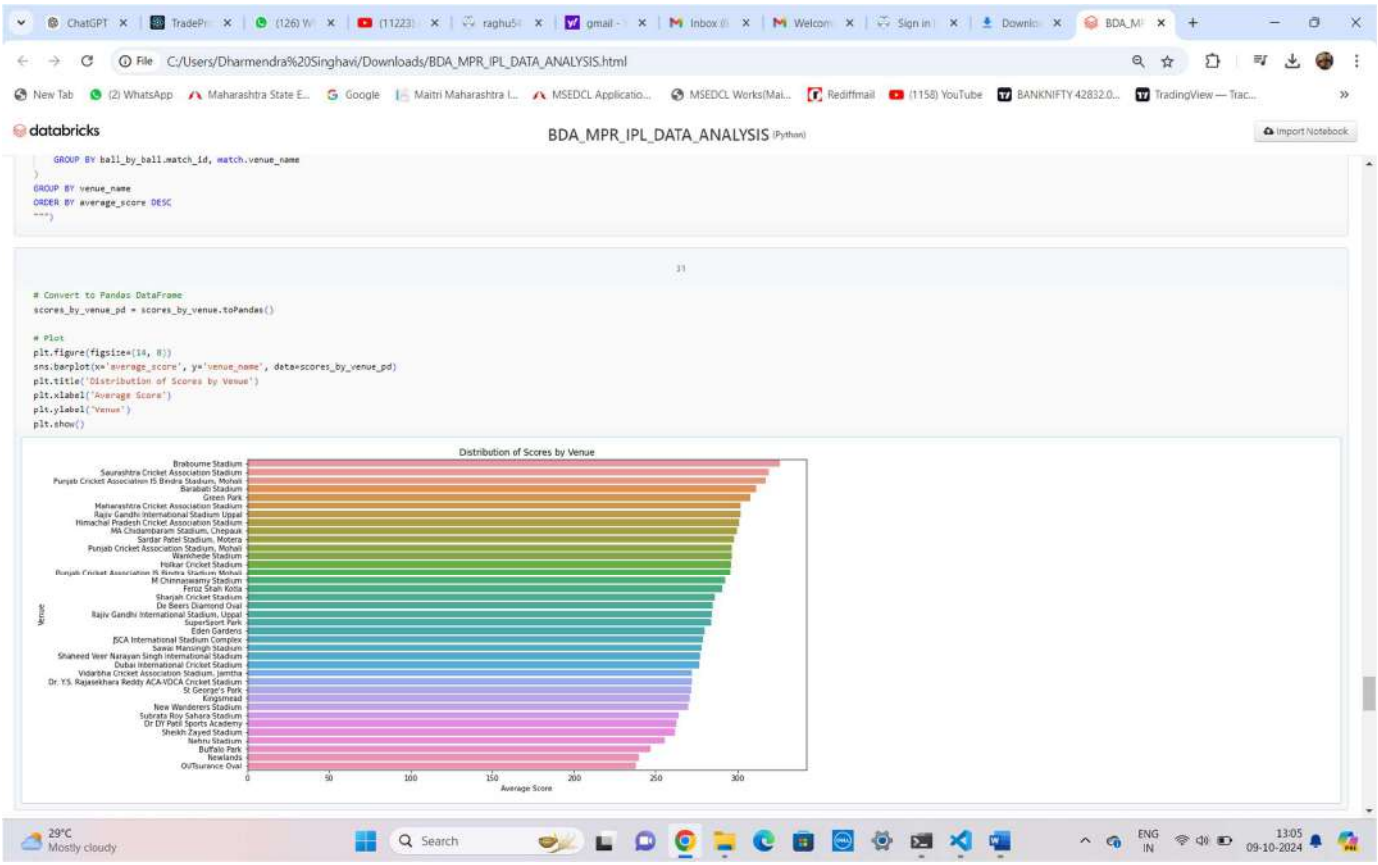



databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
spark
SparkSession - hive
SparkContext
Spark UI
Version
v3.3.2
Master
local[8]
AppName
Databricks Shell
5
ball_by_ball_schema = StructType([
StructField("match_id", IntegerType(), True),
StructField("over_id", IntegerType(), True),
StructField("ball_id", IntegerType(), True),
StructField("innings_no", IntegerType(), True),
StructField("team_batting", StringType(), True),
StructField("team_bowling", StringType(), True),
StructField("striker_batting_position", IntegerType(), True),
StructField("extra_type", StringType(), True),
StructField("runs_scored", IntegerType(), True),
StructField("extra_runs", IntegerType(), True),
StructField("wides", IntegerType(), True),
StructField("legbyes", IntegerType(), True),
StructField("byes", IntegerType(), True),
StructField("noballs", IntegerType(), True),
StructField("penalty", IntegerType(), True),
StructField("bowler_extras", IntegerType(), True),
StructField("out_type", StringType(), True),
StructField("caught", BooleanType(), True),
databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
6
ball_by_ball_df = spark.read.schema(ball_by_ball_schema).format("csv").option("header","true").load("s3://ipl-data-analysis-project/Ball_By_Ball.csv")
7
match_schema = StructType([
StructField("match_sk", IntegerType(), True),
StructField("match_id", IntegerType(), True),
StructField("team1", StringType(), True),
StructField("team2", StringType(), True),
StructField("match_date", DateType(), True),
StructField("season_year", IntegerType(), True),
StructField("venue_name", StringType(), True),
StructField("city_name", StringType(), True),
StructField("country_name", StringType(), True),
StructField("toss_winner", StringType(), True),
StructField("match_winner", StringType(), True),
StructField("toss_name", StringType(), True),
StructField("win_type", StringType(), True),
StructField("outcome_type", StringType(), True),
StructField("manofmach", StringType(), True),
StructField("win_margin", IntegerType(), True),
StructField("country_id", IntegerType(), True)
])
match_df = spark.read.schema(match_schema).format("csv").option("header","true").load("s3://ipl-data-analysis-project/Match.csv")
8
player_schema = StructType([
StructField("player_sk", IntegerType(), True),
StructField("player_id", IntegerType(), True),

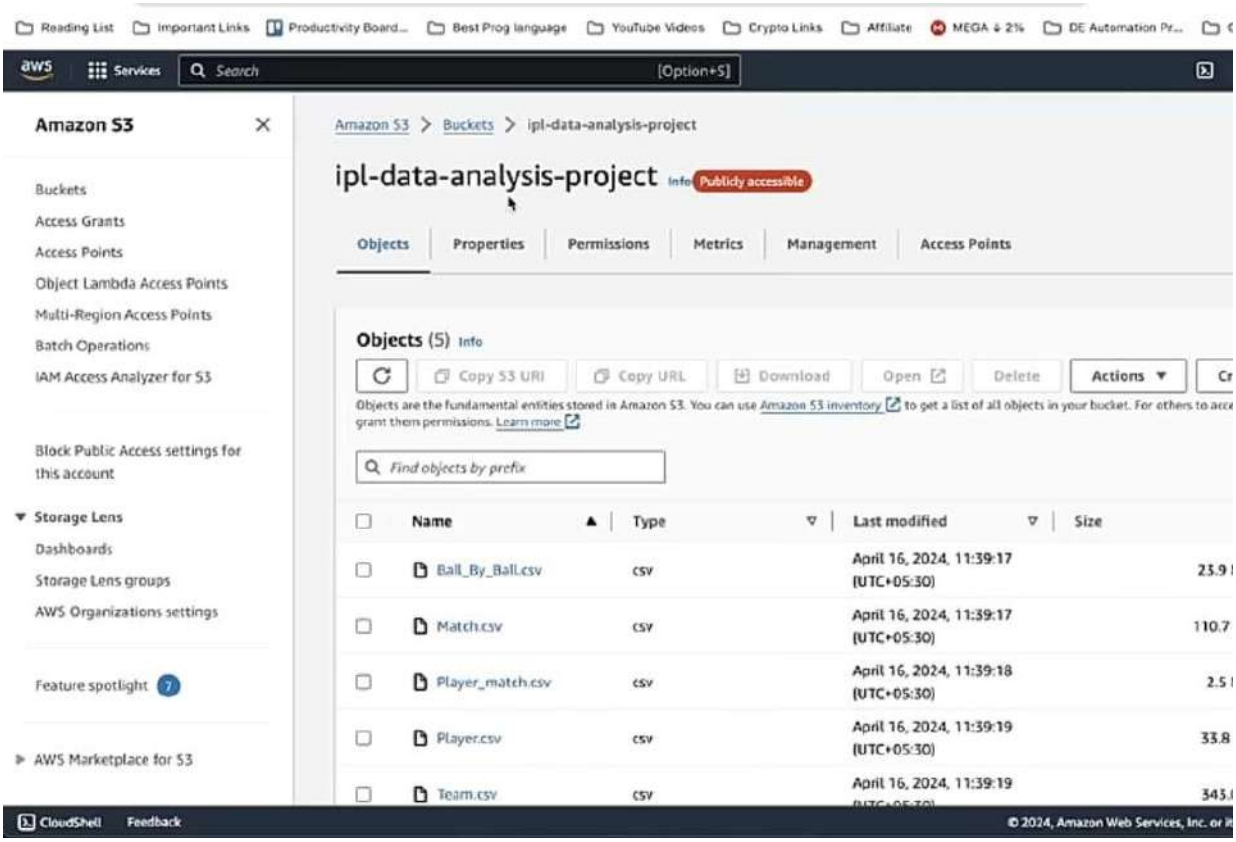

databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
11
# Filter to include only valid deliveries (excluding extras like wides and no balls for specific analyses)
ball_by_ball_df = ball_by_ball_df.filter((col("wides") == 0) & (col("noballs")==0))
# Aggregation: Calculate the total and average runs scored in each match and inning
total_and_avg_runs = ball_by_ball_df.groupBy("match_id", "innings_no").agg(
sum("runs_scored").alias("total_runs"),
avg("runs_scored").alias("average_runs")
)
12
# Window Function: Calculate running total of runs in each match for each over
windowSpec = Window.partitionBy("match_id","innings_no").orderBy("over_id")
ball_by_ball_df = ball_by_ball_df.withColumn(
"running_total_runs",
sum("runs_scored").over(windowSpec)
)
13
# Conditional Column: Flag for high impact balls (either a wicket or more than 6 runs including extras)
ball_by_ball_df = ball_by_ball_df.withColumn(
"high_impact",
when((col("runs_scored") + col("extra_runs") > 6) | (col("bowler_wicket") == True), True).otherwise(False)
)
14
ball_by_ball_df.show(5)
29°C
Mostly cloudy
Search
ENG
IN
13:03
09-10-2024
databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
# Conditional Column: Flag for high impact balls (either a wicket or more than 6 runs including extras)
ball_by_ball_df = ball_by_ball_df.withColumn(
"high_impact",
when((col("runs_scored") + col("extra_runs") > 6) | (col("bowler_wicket") == True), True).otherwise(False)
)
14
ball_by_ball_df.show(5)
15
29°C
Mostly cloudy
Search
ENG
IN
13:03
09-10-2024


databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
# Extracting year, month, and day from the match date for more detailed time-based analysis
match_df = match_df.withColumn("year", year("match_date"))
match_df = match_df.withColumn("month", month("match_date"))
match_df = match_df.withColumn("day", dayofmonth("match_date"))
# High margin win: categorizing win margins into 'high', 'medium', and 'low'
match_df = match_df.withColumn(
"win_margin_category",
when(col("win_margin") >= 100, "High")
.when((col("win_margin") >= 50) & (col("win_margin") < 100), "Medium")
.otherwise("Low")
)
# Analyze the impact of the toss: who wins the toss and the match
match_df = match_df.withColumn(
"toss_match_winner",
when(col("toss_winner") == col("match_winner"), "Yes").otherwise("No")
)
# Show the enhanced match DataFrame
match_df.show(2)
only showing top 2 rows
databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
# Assuming 'economical_bowlers_powerplay' is already executed and available as a Spark DataFrame
economical_bowlers_pd = economical_bowlers_powerplay.toPandas()
# Visualizing using Matplotlib
plt.figure(figsize=(12, 8))
# Limiting to top 10 for clarity in the plot
top_economical_bowlers = economical_bowlers_pd.nsmallest(10, 'avg_runs_per_ball')
plt.bar(top_economical_bowlers['player_name'], top_economical_bowlers['avg_runs_per_ball'], color='skyblue')
plt.xlabel('Bowler Name')
plt.ylabel('Average Runs per Ball')
plt.title('Most Economical Bowlers in Powerplay Overs (Top 10)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
Most Economical Bowlers in Powerplay Overs (Top 10)
Average Runs per Ball
Bowler Name


BDA_MPR_IPL_DATA_ANALYSIS
Impact of Winning Toss on Match Outcomes
Match Outcome
Lost
Won
Number of Matches
Toss Winner
BDA_MPR_IPL_DATA_ANALYSIS
Average Runs Scored by Batsmen in Winning Matches (Top 10 Scorers)
Average Runs in Wins
Player Name


databricks
BDA_MPR_IPL_DATA_ANALYSIS (Python)
Import Notebook
GROUP BY ball_by_ball.match_id, match.venue_name
)
GROUP BY venue_name
ORDER BY average_score DESC
"""
)
31
# Convert to Pandas DataFrame
scores_by_venue_pd = scores_by_venue.toPandas()
# Plot
plt.figure(figsize=(14, 8))
sns.barplot(x='average_score', y='venue_name', data=scores_by_venue_pd)
plt.title('Distribution of Scores by Venue')
plt.xlabel('Average Score')
plt.ylabel('Venue')
plt.show()
Distribution of Scores by Venue
Brabourne Stadium
Saurashtra Cricket Association Stadium
Punjab Cricket Association IS Bindra Stadium, Mohali
Barabati Stadium
Green Park
Maharashtra Cricket Association Stadium
Rajiv Gandhi International Stadium Uppal
Himachal Pradesh Cricket Association Stadium
MA Chidambaram Stadium, Chepauk
Sardar Patel Stadium, Motera
Punjab Cricket Association Stadium, Mohali
Wankhede Stadium
Holkar Cricket Stadium
Punjab Cricket Association IS Bindra Stadium, Mohali
M Chinnaswamy Stadium
Feroz Shah Kotla
Sharjah Cricket Stadium
De Beers Diamond Oval
Rajiv Gandhi International Stadium, Uppal
SuperSport Park
Eden Gardens
JSCA International Stadium Complex
Sawai Mansingh Stadium
Shaheed Veer Narayan Singh International Stadium
Dubai International Cricket Stadium
Vidarbha Cricket Association Stadium, Jamtha
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium
St George's Park
Kingsmead
New Wanderers Stadium
Subrata Roy Sahara Stadium
Dr DY Patil Sports Academy
Sheikh Zayed Stadium
Nehru Stadium
Buffalo Park
Newlands
OUTsurance Oval
Venue
0
50
100
150
200
250
300
Average Score
29°C
Mostly cloudy
Search
ENG
IN
13:05
09-10-2024

Amazon S3:


Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > ipl-data-analysis-project
ipl-data-analysis-project
Publicly accessible
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (5)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Find objects by prefix
Name
Type
Last modified
Size
Ball_By_Ball.csv
csv
April 16, 2024, 11:39:17 (UTC+05:30)
Match.csv
csv
April 16, 2024, 11:39:17 (UTC+05:30)
Player_match.csv
csv
April 16, 2024, 11:39:18 (UTC+05:30)
Player.csv
csv
April 16, 2024, 11:39:19 (UTC+05:30)
Team.csv
csv
CloudShell
Feedback

Visualizations:
The three graphs you provided appear to represent the following:

1. Distribution of Scores by Venue (First Graph):
   - This bar plot shows the average score distribution across various IPL venues. Each bar represents a different stadium or venue, and the length of the bar indicates the average score achieved in matches held at that venue. Higher bars correspond to venues where teams generally score more, and vice versa.

2. Impact of Winning Toss on Match Outcomes (Second Graph):
   - This is a count plot representing the relationship between winning the toss and match outcomes. The x-axis lists the toss winners (teams), and the y-axis shows the number of matches won or lost after winning the toss. The two colors (orange and blue) represent matches won and lost, respectively. This analysis helps understand how much impact winning the toss has on winning the match.

3. Average Runs Scored by Batsmen in Winning Matches (Third Graph):
   - This bar plot shows the top 10 batsmen who have scored the highest average runs in winning matches. The x-axis lists the players' names, and the y-axis represents the average number of runs they scored in games that their team won. The higher the bar, the better the performance of the player in winning matches.

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

Conclusion:

Through this comprehensive analysis of IPL data using Apache Spark on Databricks, we gained key insights into factors that influence match outcomes and performance trends across different dimensions. The analysis provided valuable findings:

1. **Venue Analysis**: The distribution of scores by venue revealed significant variations in average scores, suggesting that pitch conditions, weather, and ground dimensions play a crucial role in determining scoring trends at different stadiums. This insight is beneficial for teams in preparing strategies tailored to specific venues.
2. **Toss Impact**: Our investigation into the effect of winning the toss on match outcomes indicated a noticeable correlation between toss results and match wins for several teams. Understanding this relationship can help teams make informed decisions when they win the toss, potentially increasing their chances of securing a victory.
3. **Player Performance**: The analysis of individual player contributions highlighted the top-performing batsmen who consistently scored runs in winning matches. These players are key to their teams' success, and such insights can guide team selection and strategy formulation for future matches.

In conclusion, the use of big data tools like Apache Spark and Databricks enabled efficient handling of large IPL datasets, providing real-time and actionable insights. This type of analysis can benefit teams, analysts, and fans by deepening their understanding of the factors that influence match results and player performance. Such data-driven approaches can further enhance strategic planning in future IPL seasons.

Future Scope:

1. Advanced Analytics: Explore advanced analytics techniques such as machine learning and predictive modeling to forecast trends, identify emerging players, and personalize user recommendations.

2. Real-time Monitoring: Implement real-time monitoring and analytics capabilities to capture performance trends, player performance, and match-wise performance, enabling proactive

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment 1

Q1) Write brief on Big data characteristics, Hadoop architecture, Hadoop Ecosystem.

=) Big data contains a large amount of data that is not being processed by traditional data storage or processing unit.

=) There are five V's of Big data that explain the characteristics :-

1.) Volume

- The name Big Data itself is related to enormous size.
- Big data is vast volumes of data generated from many sources daily, such as business, machines etc.

2.) Variety

- Big data can be stucted, unstructed and semi-structured that are being collected from different sources.
- for eg :- Web server logs i.e the log file is created and maintained by some server that contains a list of activities.

3.) Veracity

- Veracity means how much the data is reliable. It has many ways to filter or translate the data.
- Veracity is procen of being able to handle and manage data efficiently
- for eg :- Facebook posts with hashtags.

4.) Value

- Value is an intrinsic characteristics of big data.
- It is not data that we procen or store. It is valuable and reliable data that we store procen and also analyze.

5.) Velocity

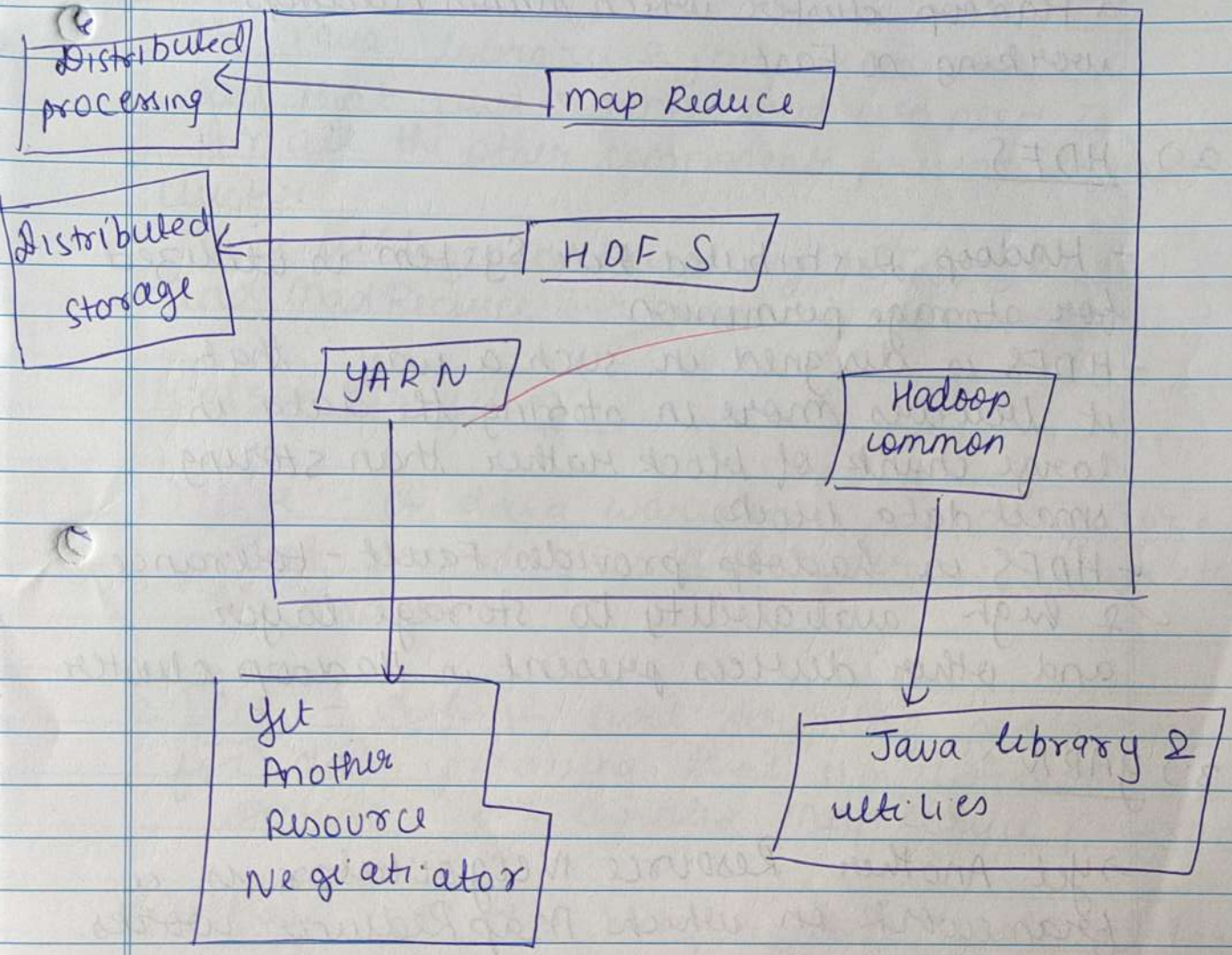
- Velocity plays an important role compared to other. Velocity creates speed by which data is created in real time.
- It contains the linking of incoming data sets, speeds, rate of change and activity burst.

- Hadoop architecture :-

- Hadoop is framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data

It consist of 4 components :-

1) Map Reduce
2) HDFS
3) YARN
4) Hadoop common



THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

1.) MapReduce

⇒ MapReduce nothing but just like Algorithm or data structure that is based on the YARN framework.
⇒ major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which makes Hadoop working so Fast.

2.) HDFS

- Hadoop Distributed File System is utilized for storage permission
- HDFS is designed in such a way that it believes more in storing the data in large chunk of block rather than storing small data blocks.
- HDFS in hadoop provides Fault-tolerance & high availability to storage layer and other devices present in Hadoop cluster.

3.) YARN

⇒ Yet Another Resource Negotiator is a framework on which MapReduce works.
⇒ YARN performs 2 operation that are Job scheduling and Resource management.

i) Multi - Tenancy
ii) Scalability
iii) Cluster - utilization
iv) Compatability

4.) Hadoop common

- It java library & java files or we say that java scripts that we need to for all the other components present in Hadoop cluster.
- These utilizes are used by HDFS, YARN, and MapReduce for running the cluster.

- Hadoop Ecosystem

1.) Hive :- A data warehouse infrastructure that provide SQL-like querying capabilities on Hadoop

2.) Pig :- A high level scripting platform for data prousing that simplifies the writing for complex MapReduce program.

3.) HBase :- A distributed scalable, NoSQL database that runs on top of HDFS and suitable for real-time read/write access

4.) Flume :- A service of effeciently collecting, aggregating and moving large amount of log data

5.) Oozie :- A workflow schedular system to manage Hadoop jobs.

7.) Tookoper :- A coordination service for distributed applications, providing mechanism for synchronization & configuration management

8.) Spark :- An open source, distributed computing system for big data procesing, known for its speed and the ease of use, often integrated with Hadoop.

7/10/24 (A)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment 2

Q1.) Mining :: Social - Network Graphs.

- Clustering of social graph using Nirvan Newman algorithm
- Direct discovery of communities in social graph using Clique percolation method.

- Social network graphs are key component of modern social platform, representing relationship between individual or group.
- Two common or clustering social network graph are the Nirvan Newman algorithm and Clique Percolation method. (CPM)

- **Clustering of Social Graph using Nirvan Newman Algorithm :-**

- The Nirvan Newman algorithm, also known as Girvan Newman algorithm is hierarchial clustering algorithm used for detecting communities within a social graph

- Key steps of Algorithm :-

i) For each edge in graph, compute the edge between which measure how many shortest path between pairs of nodes pass through that edge.

2.) After removing edge, recalculate the between of the remaining edges in the graph.

3.) Continue removing the edge with the highest betweenes score until the graph break down into disconnected cluster or communities.

for eg :-
In a social network graph, this algorithm would first identify the most "central" connections and gradually removes them, allowing more tightly connected subgroup.

- Direct discovery of communities in social graph using Clique Percolation Method (CPM)

- The CPM is another approach of community detection.

- Key concept in CPM :-

1) Clique :- A clique in graph is a subset of nodes such that every two distinct nodes are connected by edge. A K-clique is clique with K-nodes.

2) Percolation :- This idea is community can be discovered by finding cliques that percolate through the graph.

- Key step in CPM

1) Identify all K-cliques in social network graph. These are group of K nodes where node is connected to every other node in group.

2) Two K cliques are said to adajacent if they share K-1 nodes. CPM identifies communities by grouping together adajacent K-clique

3) By connecting adajacent K-Clique, the CPM method form communities.

for eg:-
In a social network, a clique could represent a tightly knit Friend group, if group above share several members the CPM will treat them as part of larger overlapping community.

Sab 11/10/24 (A)