Roll No:-2103109
Group No:-44
Batch :- C32

Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss Aarav Malvia

of Computer Engg Department, Semester VII with Roll No. 2103109 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

AB 18/10/24

Teacher In-Charge

Head of the Department

Date 18/10/2024

Principal

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

Experiment 1

Aim: Installation of Hadoop and Experiment on HDFS commands

Theory:

Hadoop is an open-source framework developed by the Apache Software Foundation that allows for the distributed storage and processing of large datasets across clusters of computers using simple programming models. It is designed to scale from a single server to thousands of machines, each offering local computation and storage. Hadoop is especially useful for big data tasks where conventional processing would be inefficient or impossible due to data size.

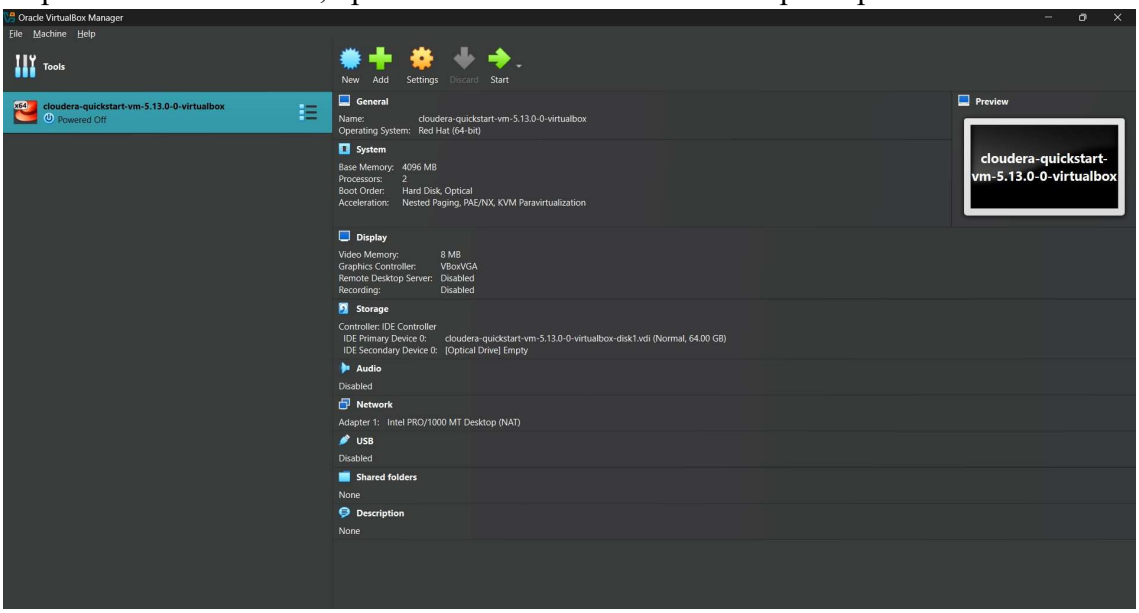
Hadoop operates through two core components:

1. HDFS (Hadoop Distributed File System) – handles data storage.
2. MapReduce – manages data processing.

Installation of Hadoop:

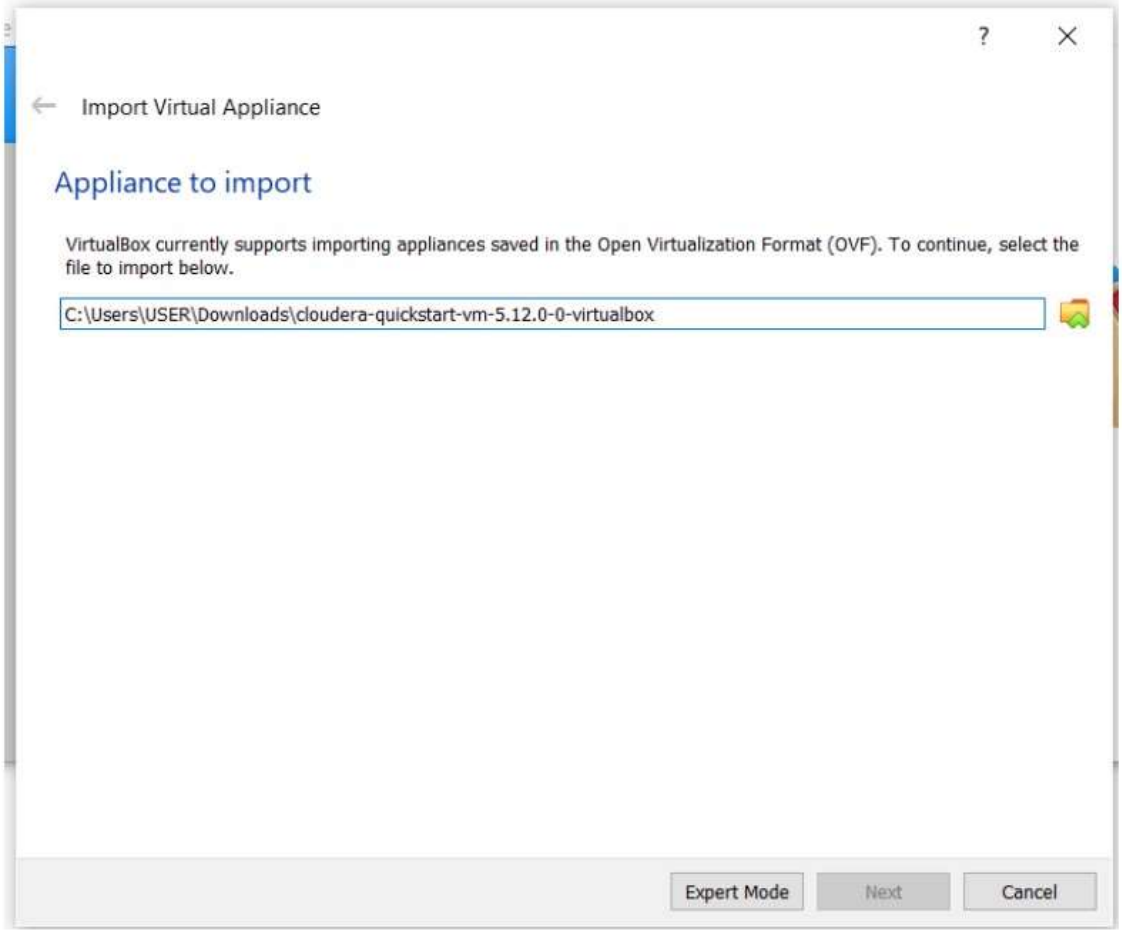
Step 1: Download Cloudera and Virtual Box

Step 2: After installation, open VirtualBox and select the Import option

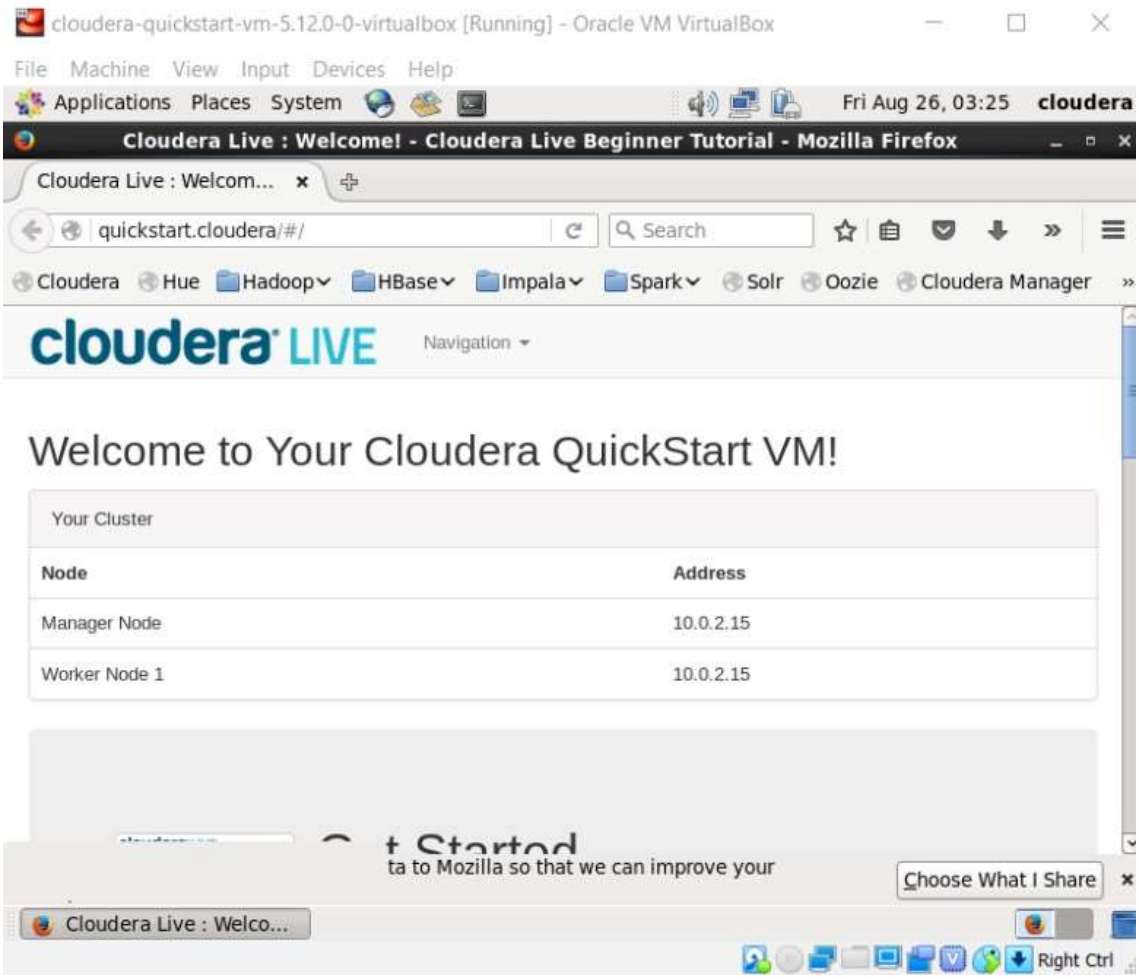


Step 3: After selecting import, include the path of the previously downloaded

Cloudera software



Step 4: Now click on the Start button and wait for a few minutes. Initially, your window will look like this.



Step 5: Once loading is completed, this window will appear.

HDFS Commands:

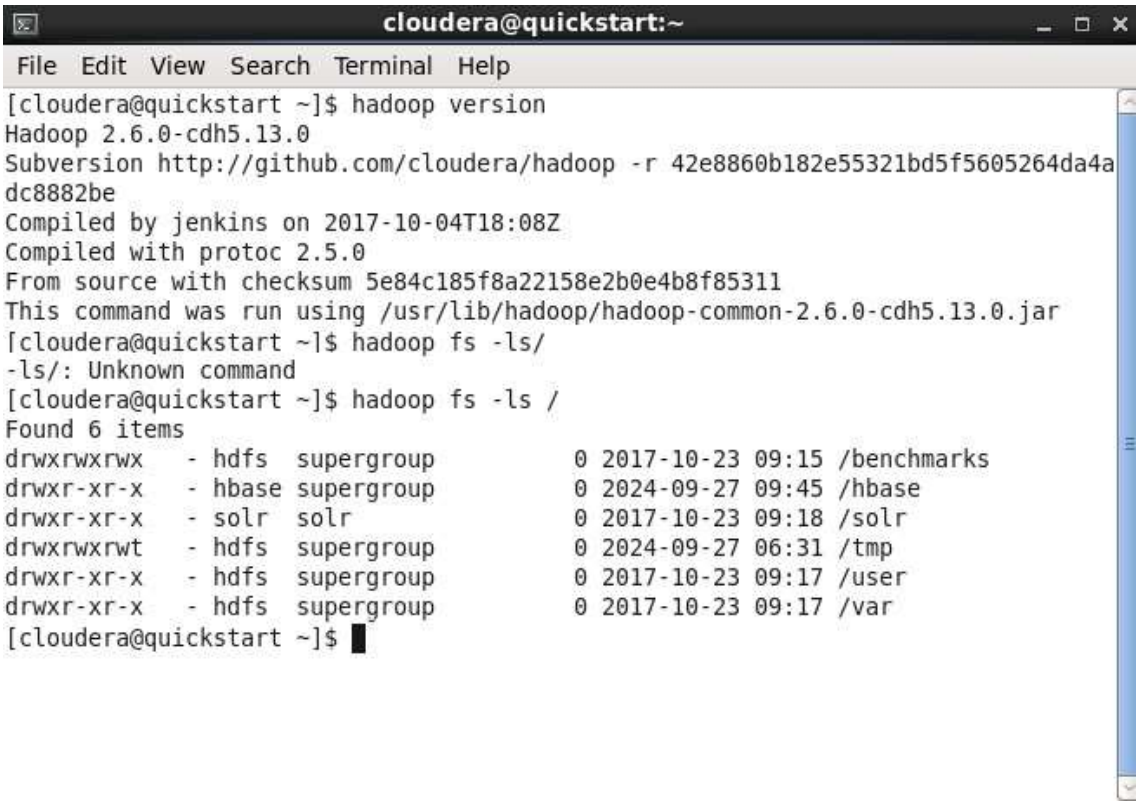
HDFS is the storage system of Hadoop and is inspired by the Google File System (GFS). It is designed to store large files in a distributed manner across a cluster, ensuring fault tolerance and high throughput.

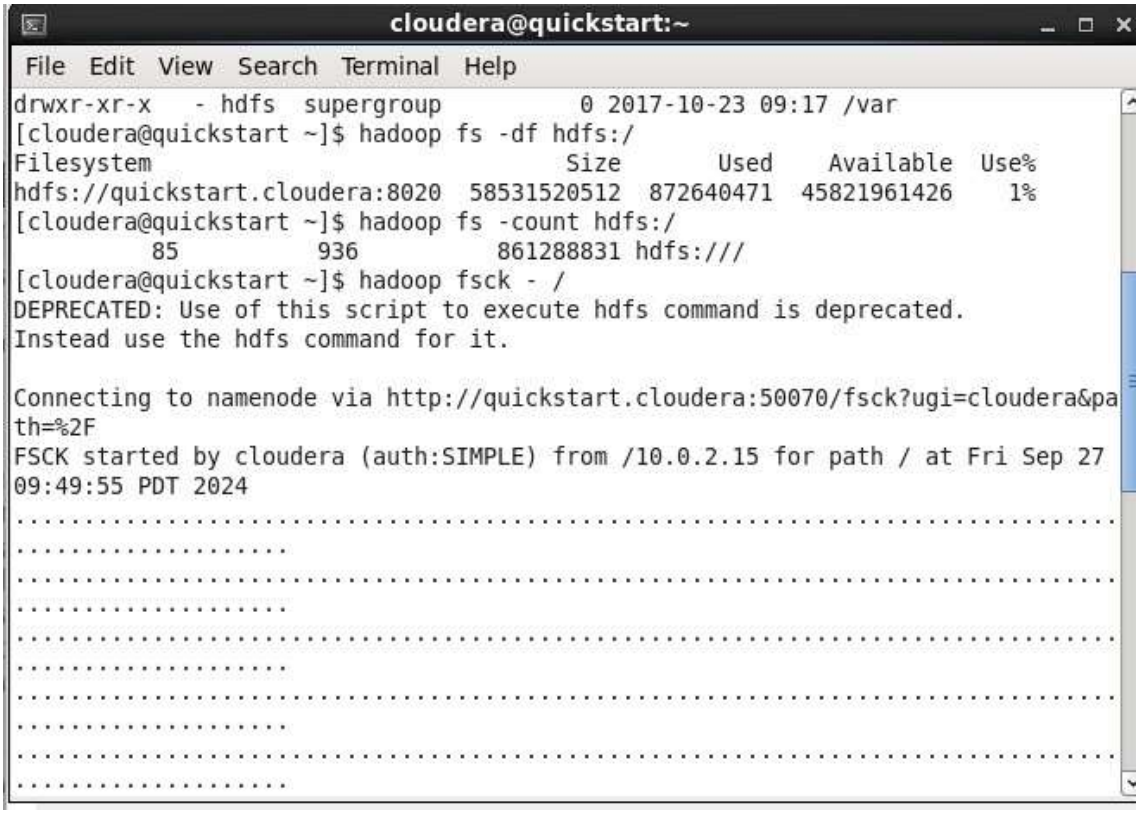
Key Concepts and Features of HDFS:

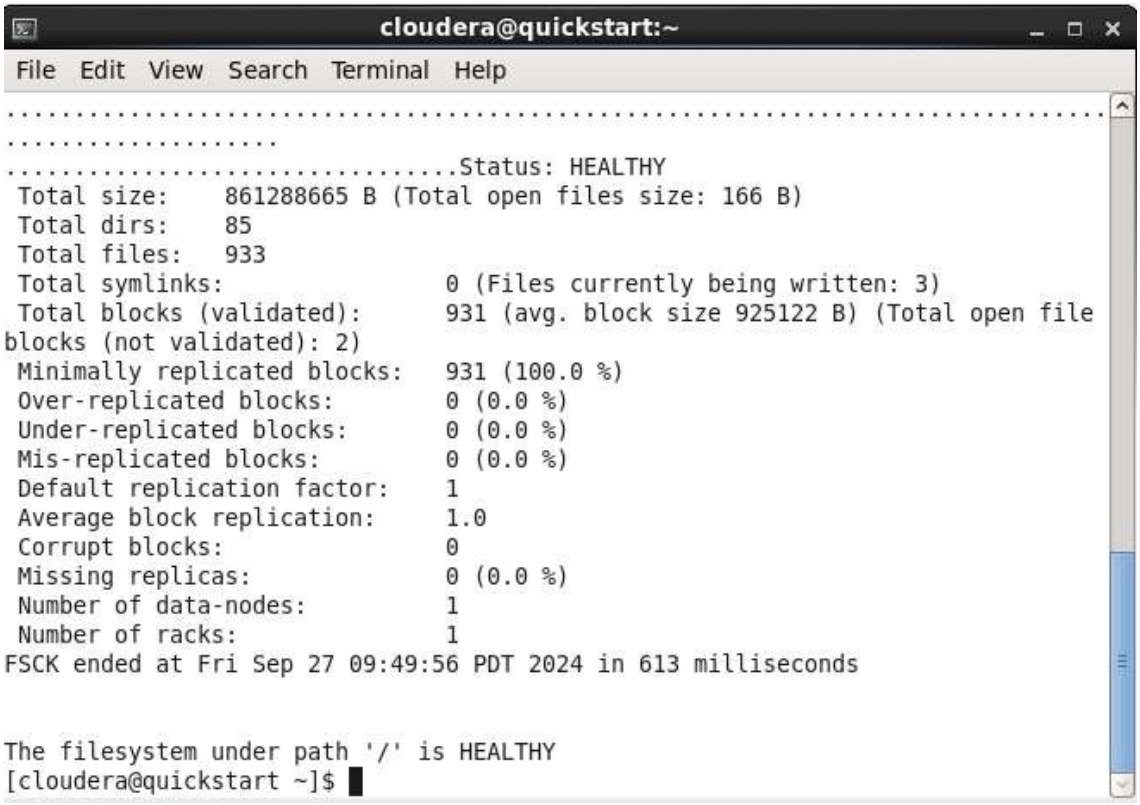
1. Blocks:
    - In HDFS, files are split into large blocks (typically 128 MB or 64 MB).
    - Each block is stored across multiple nodes in the cluster.
    - This block-level division enables efficient storage and parallel processing.

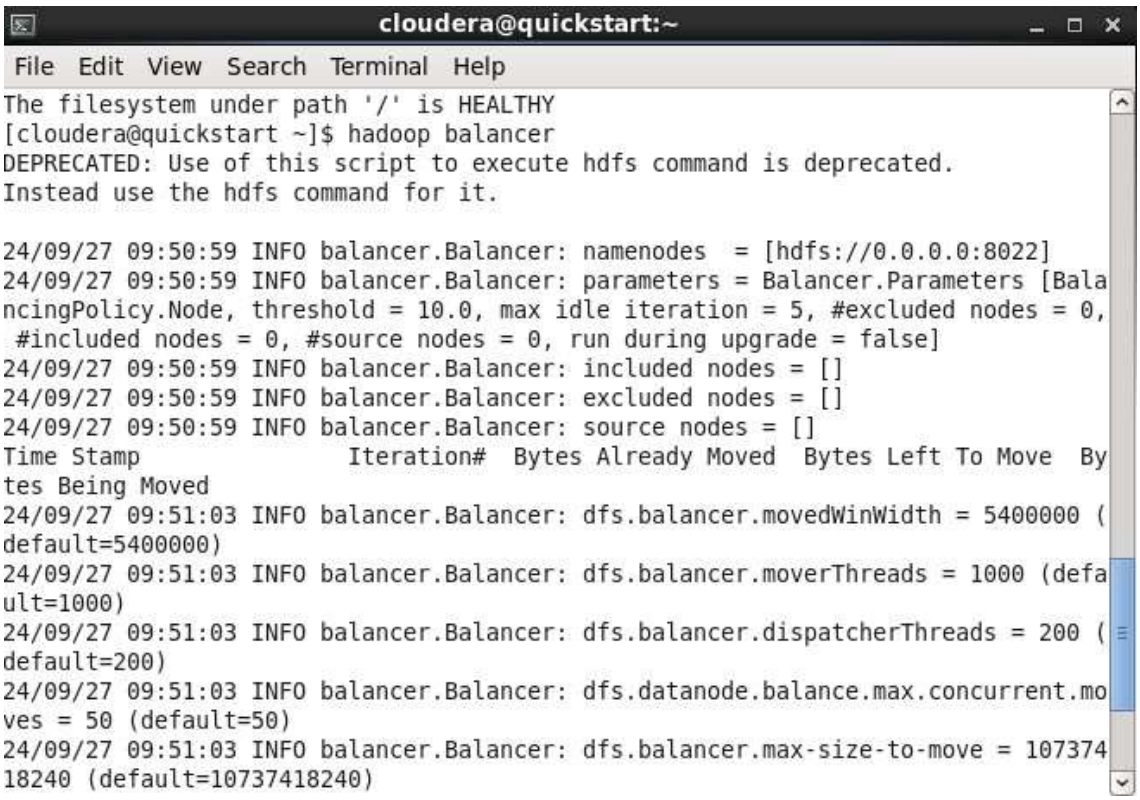
2. Replication:
    - Each block is replicated across several nodes to ensure fault tolerance. The default replication factor is three, meaning each block is stored on three different nodes.
    - If one node fails, the system automatically retrieves data from another node that holds the replica, ensuring high availability.
3. NameNode and DataNode Architecture:
    - NameNode: The master node that manages the metadata of the file system, such as the directory structure and the location of data blocks.
    - DataNodes: The worker nodes where the actual data blocks are stored. DataNodes regularly report back to the NameNode regarding the health of the system and the blocks stored on them.
4. Fault Tolerance:
    - HDFS is highly fault-tolerant. If a DataNode fails, the NameNode detects it via heartbeat signals and initiates a block replication to ensure data redundancy.
    - Data replication across multiple nodes provides fault tolerance and reliability.
5. High Throughput:
    - HDFS is optimized for high throughput rather than low latency. It is ideal for applications that need to process large volumes of data in a batch-oriented fashion, such as log processing, data mining, or machine learning on large datasets.
    - The design focus is to provide a system that can read/write data in bulk efficiently rather than focusing on real-time operations.
6. Write Once, Read Many Model:
    - HDFS follows a "write once, read many" pattern, meaning data can only be written once and then read multiple times. This simplification helps streamline the file system's design and allows for efficient data processing.
7. Scalability:
    - HDFS is designed to scale to hundreds or thousands of nodes, making it suitable for handling petabytes of data. It can scale horizontally by adding more nodes to the cluster.
8. Data Integrity:

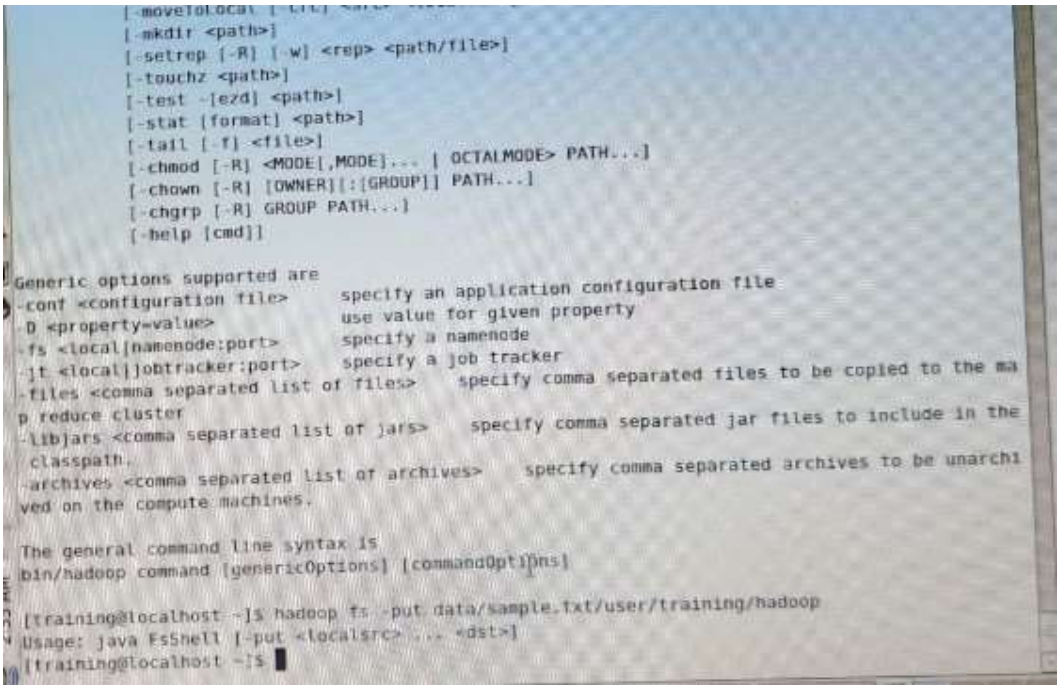
- HDFS ensures data integrity using checksum verification. When data is written, HDFS generates a checksum for each block, and during reading, it verifies the checksum to detect data corruption.

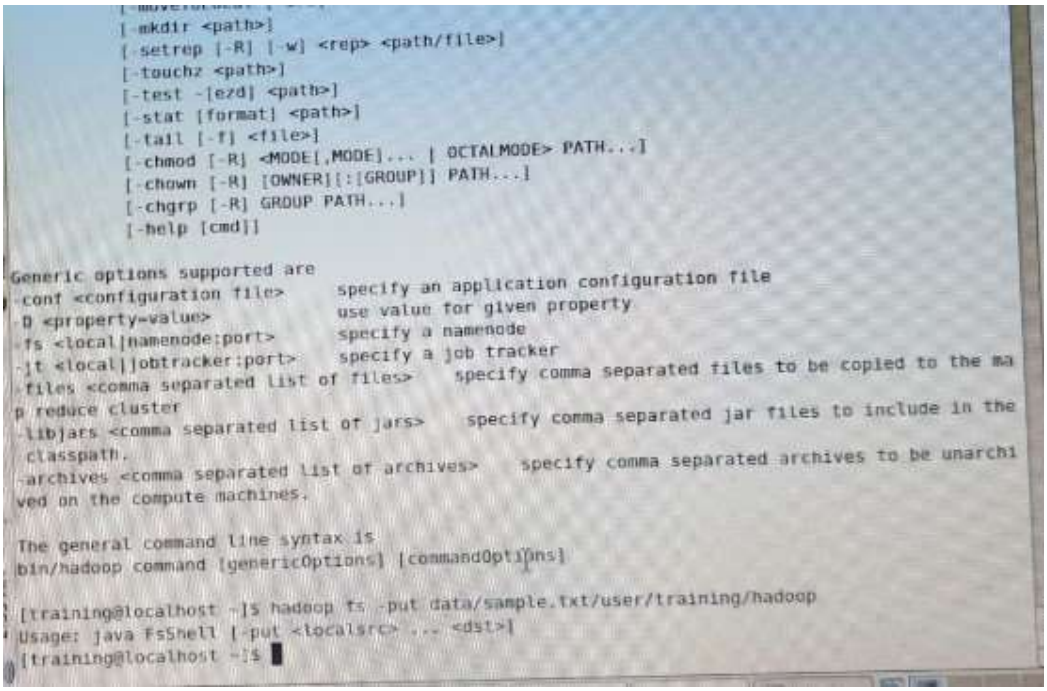

cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ hadoop version
Hadoop 2.6.0-cdh5.13.0
Subversion http://github.com/cloudera/hadoop -r 42e8860b182e55321bd5f5605264da4a
dc8882be
Compiled by jenkins on 2017-10-04T18:08Z
Compiled with protoc 2.5.0
From source with checksum 5e84c185f8a22158e2b0e4b8f85311
This command was run using /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar
[cloudera@quickstart ~]$ hadoop fs -ls/
-ls/: Unknown command
[cloudera@quickstart ~]$ hadoop fs -ls /
Found 6 items
drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks
drwxr-xr-x - hbase supergroup 0 2024-09-27 09:45 /hbase
drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr
drwxrwxrwt - hdfs supergroup 0 2024-09-27 06:31 /tmp
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /user
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$


cloudera@quickstart:~
File Edit View Search Terminal Help
drwxr-xr-x - hdfs supergroup 0 2017-10-23 09:17 /var
[cloudera@quickstart ~]$ hadoop fs -df hdfs:/
Filesystem Size Used Available Use%
hdfs://quickstart.cloudera:8020 58531520512 872640471 45821961426 1%
[cloudera@quickstart ~]$ hadoop fs -count hdfs:/
85 936 861288831 hdfs:///
[cloudera@quickstart ~]$ hadoop fsck - /
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Connecting to namenode via http://quickstart.cloudera:50070/fsck?ugi=cloudera&pa
th=%2F
FSCK started by cloudera (auth:SIMPLE) from /10.0.2.15 for path / at Fri Sep 27
09:49:55 PDT 2024


cloudera@quickstart:~
File Edit View Search Terminal Help
.....
.....
.....Status: HEALTHY
 Total size: 861288665 B (Total open files size: 166 B)
 Total dirs: 85
 Total files: 933
 Total symlinks: 0 (Files currently being written: 3)
 Total blocks (validated):   931 (avg. block size 925122 B) (Total open file
blocks (not validated): 2)
 Minimally replicated blocks: 931 (100.0 %)
 Over-replicated blocks: 0 (0.0 %)
 Under-replicated blocks: 0 (0.0 %)
 Mis-replicated blocks: 0 (0.0 %)
 Default replication factor: 1
 Average block replication: 1.0
 Corrupt blocks: 0
 Missing replicas: 0 (0.0 %)
 Number of data-nodes: 1
 Number of racks: 1
FSCK ended at Fri Sep 27 09:49:56 PDT 2024 in 613 milliseconds
The filesystem under path '/' is HEALTHY
[cloudera@quickstart ~]$


cloudera@quickstart:~
File Edit View Search Terminal Help
The filesystem under path '/' is HEALTHY
[cloudera@quickstart ~]$ hadoop balancer
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
24/09/27 09:50:59 INFO balancer.Balancer: namenodes = [hdfs://0.0.0.0:8022]
24/09/27 09:50:59 INFO balancer.Balancer: parameters = Balancer.Parameters [Bala
ncingPolicy.Node, threshold = 10.0, max idle iteration = 5, #excluded nodes = 0,
 #included nodes = 0, #source nodes = 0, run during upgrade = false]
24/09/27 09:50:59 INFO balancer.Balancer: included nodes = []
24/09/27 09:50:59 INFO balancer.Balancer: excluded nodes = []
24/09/27 09:50:59 INFO balancer.Balancer: source nodes = []
Time Stamp   Iteration# Bytes Already Moved Bytes Left To Move By
tes Being Moved
24/09/27 09:51:03 INFO balancer.Balancer: dfs.balancer.movedWinWidth = 5400000 (
default=5400000)
24/09/27 09:51:03 INFO balancer.Balancer: dfs.balancer.moverThreads = 1000 (defa
ult=1000)
24/09/27 09:51:03 INFO balancer.Balancer: dfs.balancer.dispatcherThreads = 200 (
default=200)
24/09/27 09:51:03 INFO balancer.Balancer: dfs.datanode.balance.max.concurrent.mo
ves = 50 (default=50)
24/09/27 09:51:03 INFO balancer.Balancer: dfs.balancer.max-size-to-move = 107374
18240 (default=10737418240)


[-mkdir <path>]
[-setrep [-R] [-w] <rep> <path/file>]
[-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 ~]$ hadoop fs -put data/sample.txt/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$


[-mkdir <path>]
[-setrep [-R] [-w] <rep> <path/file>]
[-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 ~]$ hadoop fs -put data/sample.txt/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$

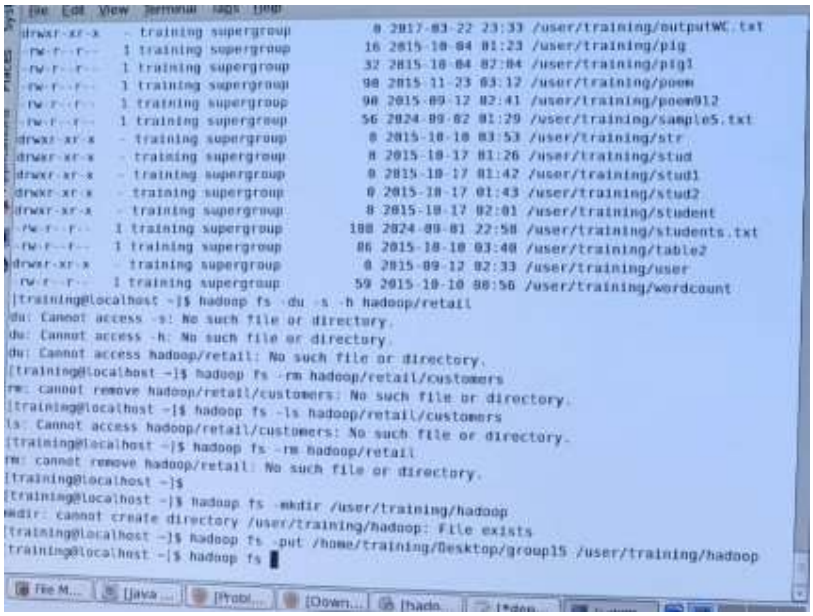
```
[training@localhost ~]$ hadoop balancer
Time Stamp               Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Move
d
24/09/02 02:33:34 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:50010
24/09/02 02:33:34 INFO balancer.Balancer: 0 over utilized nodes:
24/09/02 02:33:34 INFO balancer.Balancer: 1 under utilized nodes:  127.0.0.1:50010
The cluster is balanced. Exiting...
Balancing took 351.0 milliseconds
[training@localhost ~]$ hadoop fs -mkdir/user/training/hadoop
mkdir/user/training/hadoop: Unknown command
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>]
```

```
-rw-r--r--   1 training supergroup         24 2024-08-14 01:17 /user/training/dumA.txt
-rw-r--r--   1 training supergroup         12 2024-08-14 01:28 /user/training/dumB.txt
drwxr-xr-x   - training supergroup          0 2015-10-17 02:14 /user/training/emp
drwxr-xr-x   - training supergroup          0 2015-10-17 02:34 /user/training/emp2
drwxr-xr-x   - training supergroup          0 2015-10-10 23:18 /user/training/emp_dir
drwxr-xr-x   - training supergroup          0 2015-10-17 02:26 /user/training/employee
drwxr-xr-x   - training supergroup          0 2015-10-17 23:39 /user/training/employee2
drwxr-xr-x   - training supergroup          0 2015-10-17 23:42 /user/training/employee3
-rw-r--r--   1 training supergroup        109 2024-09-02 01:38 /user/training/group15.txt
drwxr-xr-x   - training supergroup          0 2024-07-15 22:44 /user/training/hadoop
drwxr-xr-x   - training supergroup          0 2024-07-15 23:08 /user/training/hadoop1
-rw-r--r--   1 training supergroup         44 2017-03-22 23:31 /user/training/inputWC.txt
-rw-r--r--   1 training supergroup         69 2015-10-10 02:35 /user/training/join
-rw-r--r--   1 training supergroup        105 2015-10-10 02:38 /user/training/join2
drwxr-xr-x   - training supergroup          0 2017-03-22 23:33 /user/training/outputWC.txt
-rw-r--r--   1 training supergroup         16 2015-10-04 01:23 /user/training/pig
-rw-r--r--   1 training supergroup         32 2015-10-04 02:04 /user/training/pig1
-rw-r--r--   1 training supergroup         90 2015-11-23 03:12 /user/training/poem
-rw-r--r--   1 training supergroup         90 2015-09-12 02:41 /user/training/poem912
-rw-r--r--   1 training supergroup         56 2024-09-02 01:29 /user/training/sample5.txt
drwxr-xr-x   - training supergroup          0 2015-10-10 03:53 /user/training/str
drwxr-xr-x   - training supergroup          0 2015-10-17 01:26 /user/training/stud
drwxr-xr-x   - training supergroup          0 2015-10-17 01:42 /user/training/stud1
drwxr-xr-x   - training supergroup          0 2015-10-17 01:43 /user/training/stud2
drwxr-xr-x   - training supergroup          0 2015-10-17 02:01 /user/training/student
-rw-r--r--   1 training supergroup        108 2024-09-01 22:50 /user/training/students.txt
-rw-r--r--   1 training supergroup         86 2015-10-10 03:40 /user/training/table2
drwxr-xr-x   - training supergroup          0 2015-09-12 02:33 /user/training/user
-rw-r--r--   1 training supergroup         59 2015-10-10 00:56 /user/training/wordcount
training@localhost ~]$ l2 next
```

```
          [-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 ~]$ hadoop fs -put data/sample.txt/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$ hadoop fs -ls /user/training/hadoop
Found 2 items
drwxr-xr-x   - training supergroup          0 2024-07-15 22:44 /user/training/hadoop/retail1
-rw-r--r--   1 training supergroup         32 2024-07-15 22:40 /user/training/hadoop/sample.tx
t
[training@localhost ~]$ hadoop fs -put data/retail /user/training/hadoop
put: File data/retail does not exist.
```

```
          [-moveToLocal [-crc] <src> <localdst>]
          [-mkdir <path>]
          [-setrep [-R] [-w] <rep> <path/file>]
          [-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 ~]$ hadoop fs -put data/sample.txt/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$
```

```
-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 ~]$ hadoop fs -put data/sample.txt/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$ hadoop fs -ls /user/training/hadoop
Found 2 items
drwxr-xr-x   - training supergroup          0 2024-07-15 22:44 /user/training/hadoop/retail1
-rw-r--r--   1 training supergroup         32 2024-07-15 22:48 /user/training/hadoop/sample.tx
t
[training@localhost ~]$ hadoop fs -put data/retail /user/training/hadoop
put: File data/retail does not exist.
[training@localhost ~]$ hadoop fs -put data/retail/user/training/hadoop
Usage: java FsShell [-put <localsrc> ... <dst>]
[training@localhost ~]$ hadoop fs -ls
Found 55 items
-rw-r--r--   1 training supergroup       1390 2015-10-02 20:20 /user/training/Books
drwxr-xr-x   - training supergroup          0 2024-07-16 00:26 /user/training/Saarthak
drwxr-xr-x   - training supergroup          0 2014-08-17 03:50 /user/training/WeatherData
drwxr-xr-x   - training supergroup          0 2015-10-17 02:34 /user/training/_sqoop
-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/apache_hadoop
-rw-r--r--   1 training supergroup         43 2024-08-14 00:24 /user/training/b.txt
-rw-r--r--   1 training supergroup         23 2024-09-02 01:34 /user/training/bdaExp5.txt
-rw-r--r--   1 training supergroup       2944 2015-09-26 02:37 /user/training/bookinfo
drwxr-xr-x   - training supergroup          0 2015-09-20 01:38 /user/training/class2009_dir1
-rw-r--r--   1 training supergroup        112 2024-08-14 00:57 /user/training/department.txt
```



Experiment 2

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

Theory:

Apache Sqoop is a powerful command-line tool designed to efficiently transfer large volumes of data between Hadoop and relational database servers. It enables seamless data exchange between the Hadoop ecosystem (HDFS, Hive, HBase) and popular relational databases such as MySQL, Oracle, PostgreSQL, SQL Server, and others. Sqoop plays a crucial role in Big Data environments where data movement is required for analytics, data warehousing, and ETL (Extract, Transform, Load) processes.

Key Features of Apache Sqoop

1. Efficient Bulk Data Transfer:
    - Sqoop is designed to handle bulk data efficiently, allowing it to import/export large datasets between Hadoop and relational databases without significant performance overhead.
2. Data Import:
    - Sqoop enables importing data from relational databases into Hadoop's HDFS, Apache Hive, or HBase tables. This data can then be processed using MapReduce, Hive queries, or other Big Data tools in the Hadoop ecosystem.
3. Data Export:
    - Sqoop also supports exporting data from Hadoop (HDFS, Hive, HBase) back into relational databases. This feature is useful for reporting, further processing, or archiving data back into relational systems.
4. Parallelism:
    - Sqoop can perform parallel data transfer to enhance the speed of the data import/export process. It splits the input data into multiple parts and transfers them in parallel, leveraging the distributed nature of Hadoop.

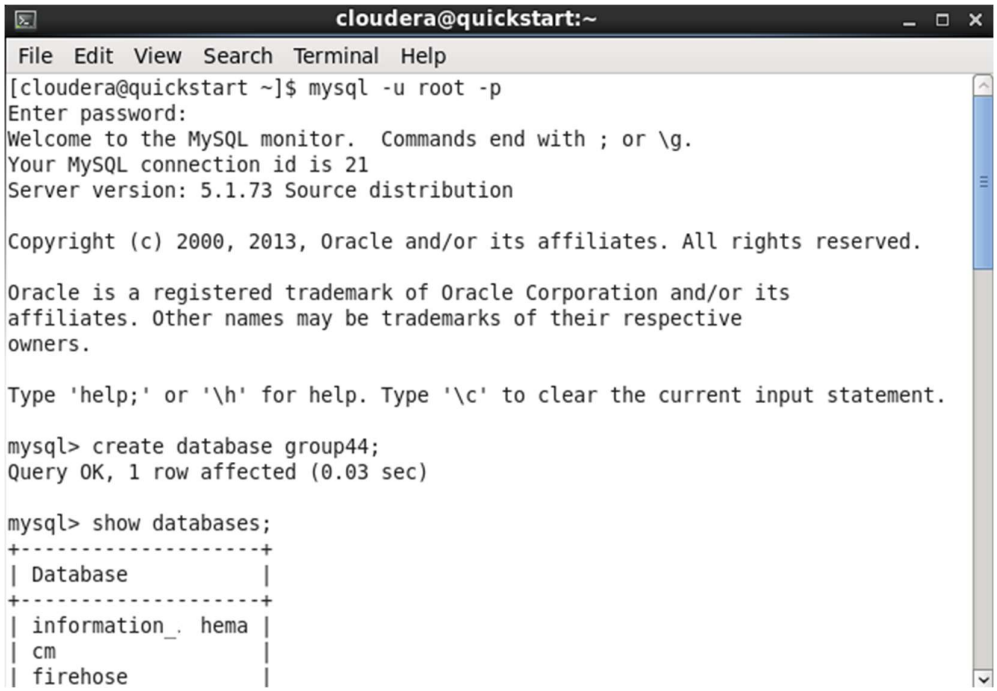
    5. Data Transformation:

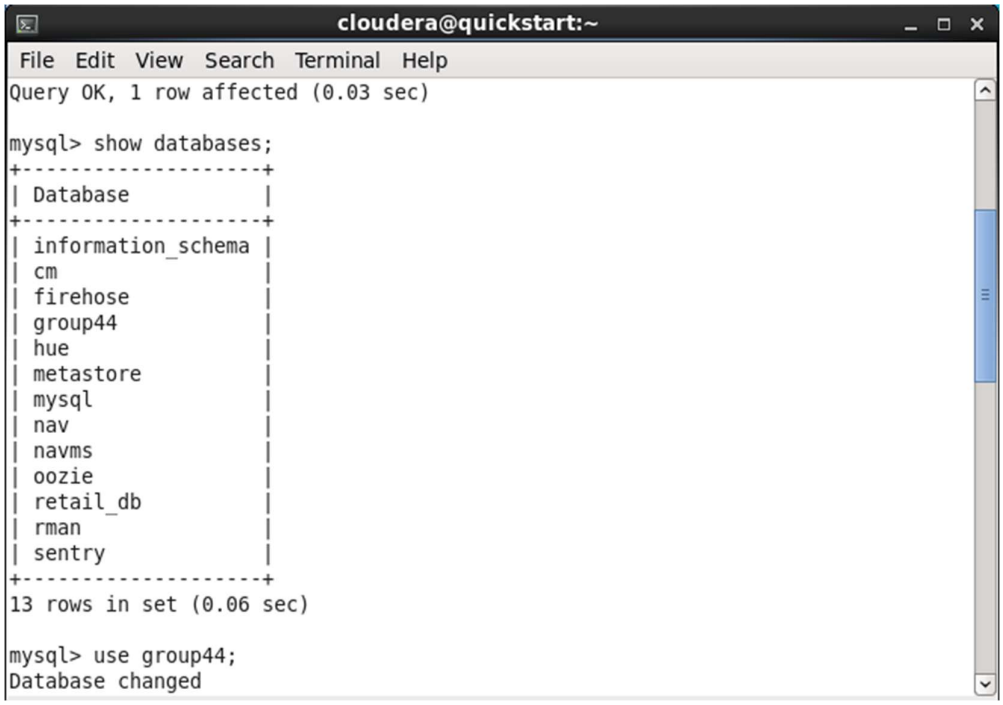
    - During data transfer, Sqoop supports lightweight transformation of the data, such as selecting specific columns, filtering rows, or applying SQL queries.
6. Incremental Data Load:

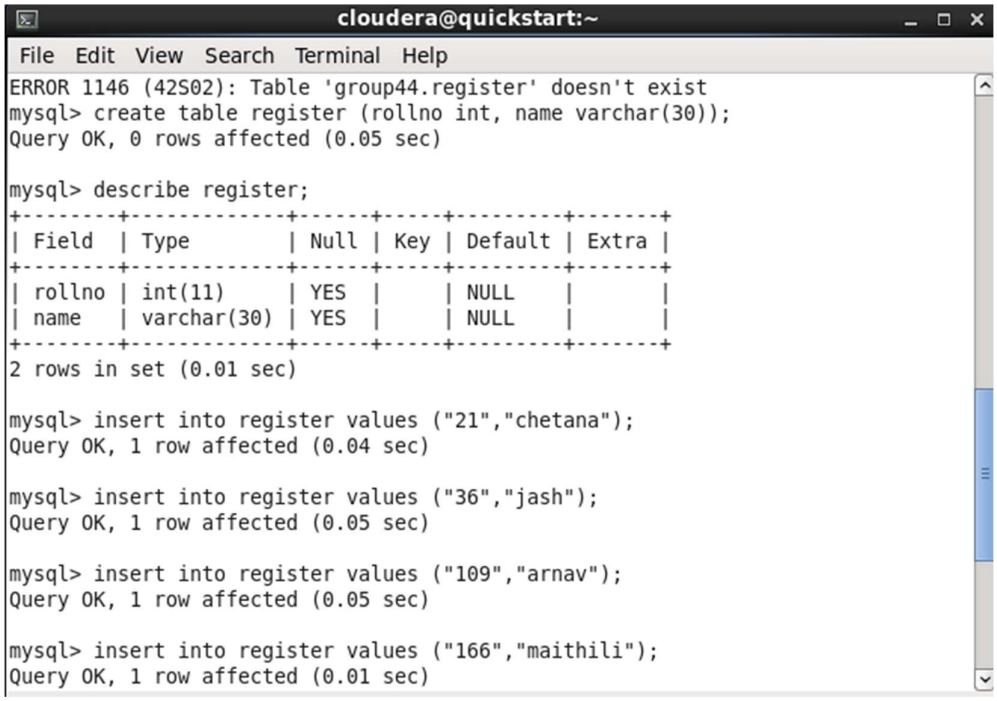
- Sqoop provides the capability to perform incremental data imports. This means it can import only the newly added or updated data from relational databases into Hadoop, reducing redundancy and improving efficiency.

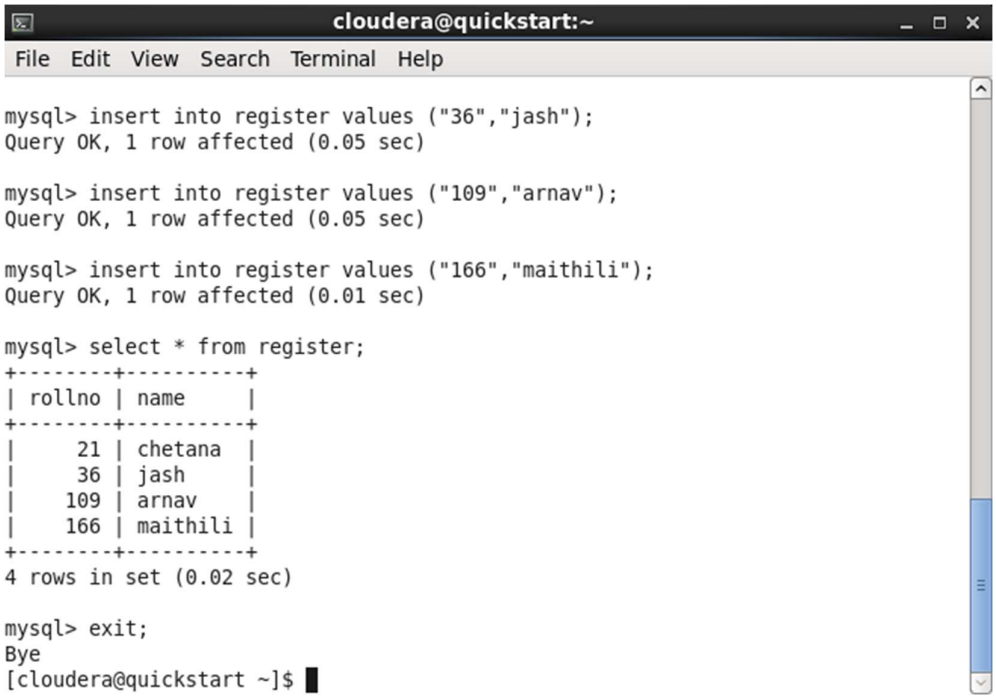
7. Compression Support:
   - Sqoop supports compression of data during transfer, which reduces storage requirements in HDFS and speeds up data transmission.
8. Direct Mode:
   - For certain databases (like MySQL and PostgreSQL), Sqoop provides a direct mode to bypass the MapReduce framework and use the database's native API for faster data transfer.

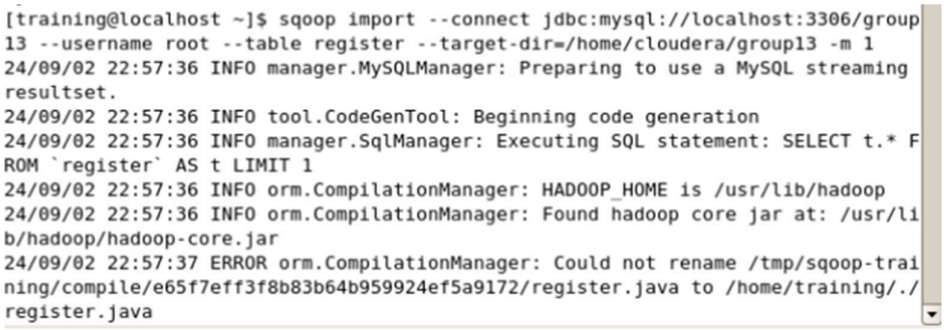
Code & Output:

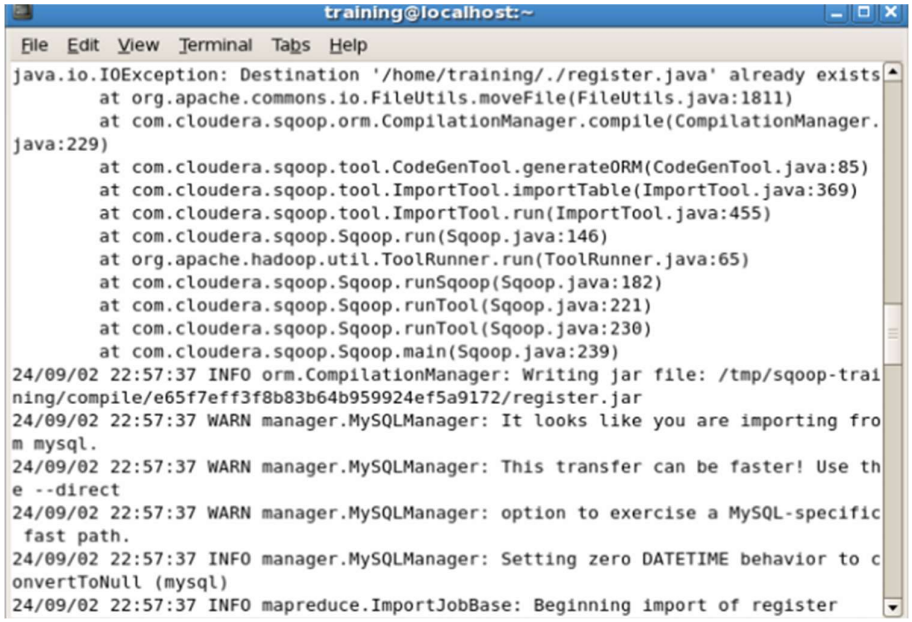


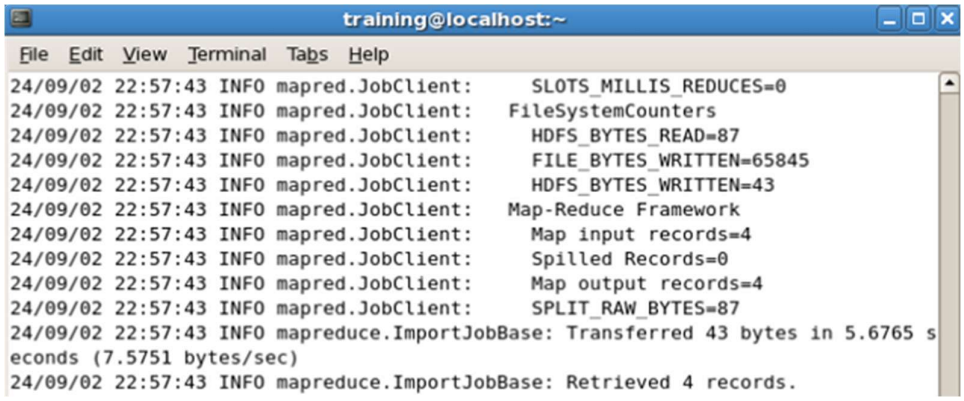

cloudera@quickstart:~
File Edit View Search Terminal Help
Query OK, 1 row affected (0.03 sec)
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| cm |
| firehose |
| group44 |
| hue |
| metastore |
| mysql |
| nav |
| navms |
| oozie |
| retail_db |
| rman |
| sentry |
+--------------------+
13 rows in set (0.06 sec)
mysql> use group44;
Database changed

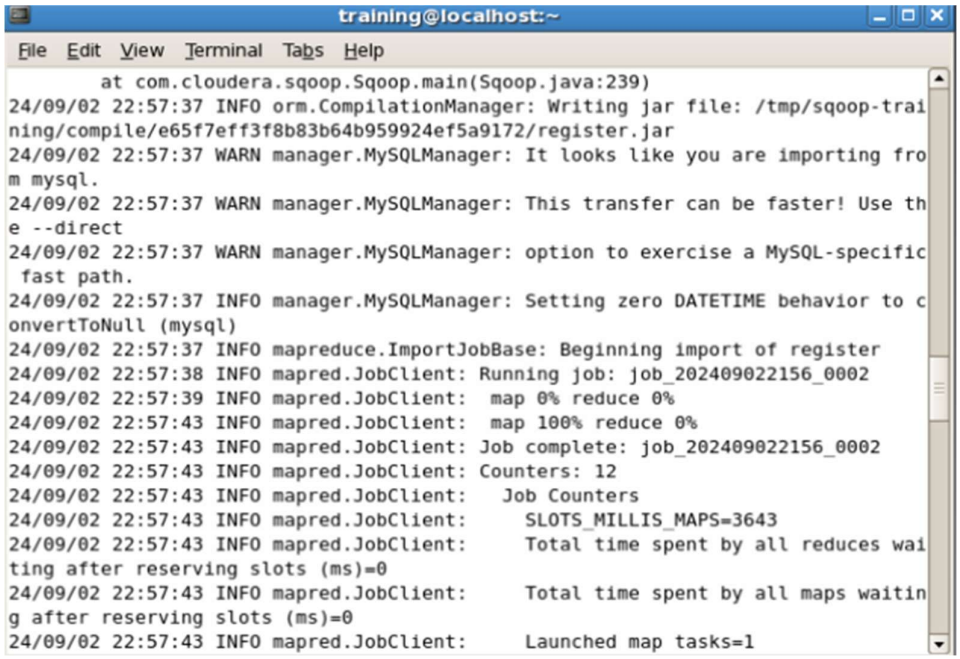

cloudera@quickstart:~
File Edit View Search Terminal Help
ERROR 1146 (42S02): Table 'group44.register' doesn't exist
mysql> create table register (rollno int, name varchar(30));
Query OK, 0 rows affected (0.05 sec)
mysql> describe register;
+--------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| rollno | int(11) | YES | | NULL | |
| name | varchar(30) | YES | | NULL | |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
mysql> insert into register values ("21","chetana");
Query OK, 1 row affected (0.04 sec)
mysql> insert into register values ("36","jash");
Query OK, 1 row affected (0.05 sec)
mysql> insert into register values ("109","arnav");
Query OK, 1 row affected (0.05 sec)
mysql> insert into register values ("166","maithili");
Query OK, 1 row affected (0.01 sec)

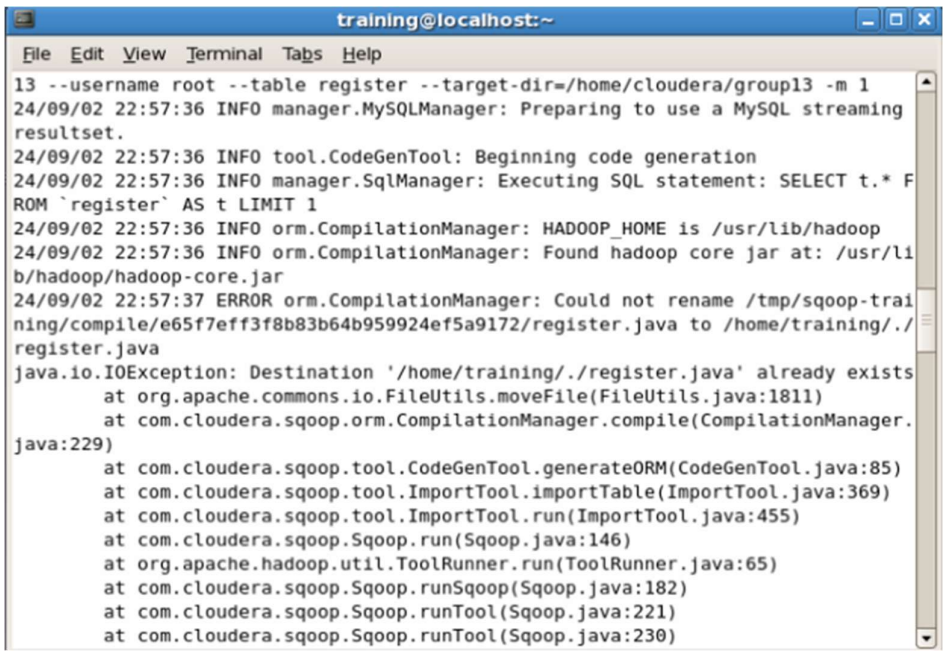

cloudera@quickstart:~
File Edit View Search Terminal Help
mysql> insert into register values ("36","jash");
Query OK, 1 row affected (0.05 sec)
mysql> insert into register values ("109","arnav");
Query OK, 1 row affected (0.05 sec)
mysql> insert into register values ("166","maithili");
Query OK, 1 row affected (0.01 sec)
mysql> select * from register;
+--------+----------+
| rollno | name |
+--------+----------+
21	chetana
36	jash
109	arnav
166	maithili
+--------+----------+
4 rows in set (0.02 sec)
mysql> exit;
Bye
[cloudera@quickstart ~]$

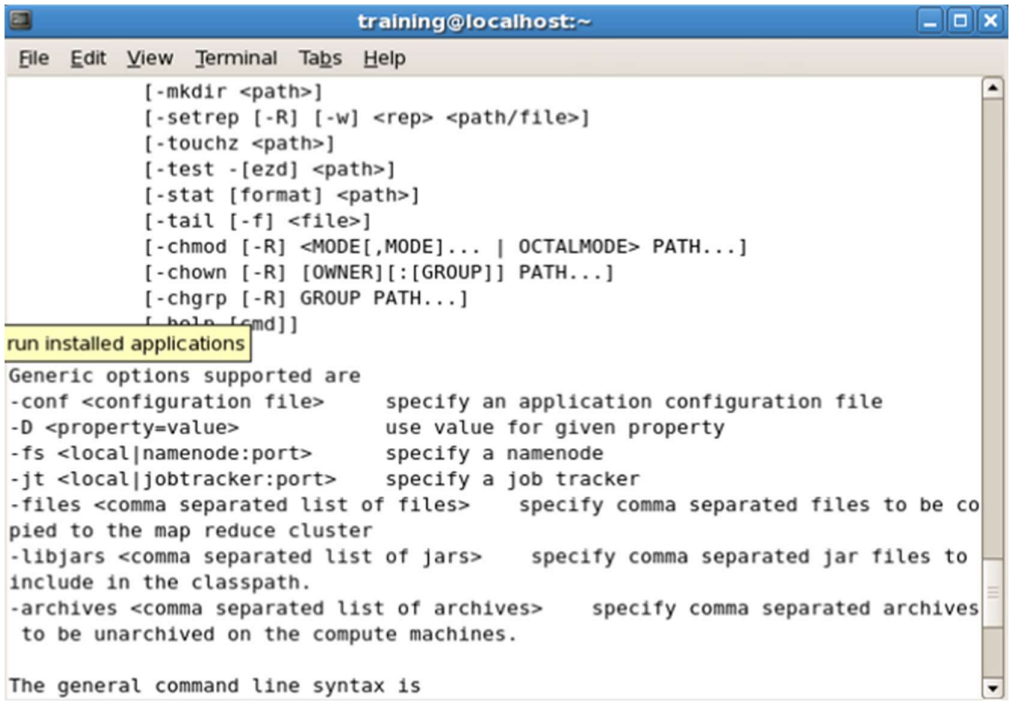

[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost:3306/group
13 --username root --table register --target-dir=/home/cloudera/group13 -m 1
24/09/02 22:57:36 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
24/09/02 22:57:36 INFO tool.CodeGenTool: Beginning code generation
24/09/02 22:57:36 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `register` AS t LIMIT 1
24/09/02 22:57:36 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
24/09/02 22:57:36 INFO orm.CompilationManager: Found hadoop core jar at: /usr/li
b/hadoop/hadoop-core.jar
24/09/02 22:57:37 ERROR orm.CompilationManager: Could not rename /tmp/sqoop-trai
ning/compile/e65f7eff3f8b83b64b959924ef5a9172/register.java to /home/training/./
register.java

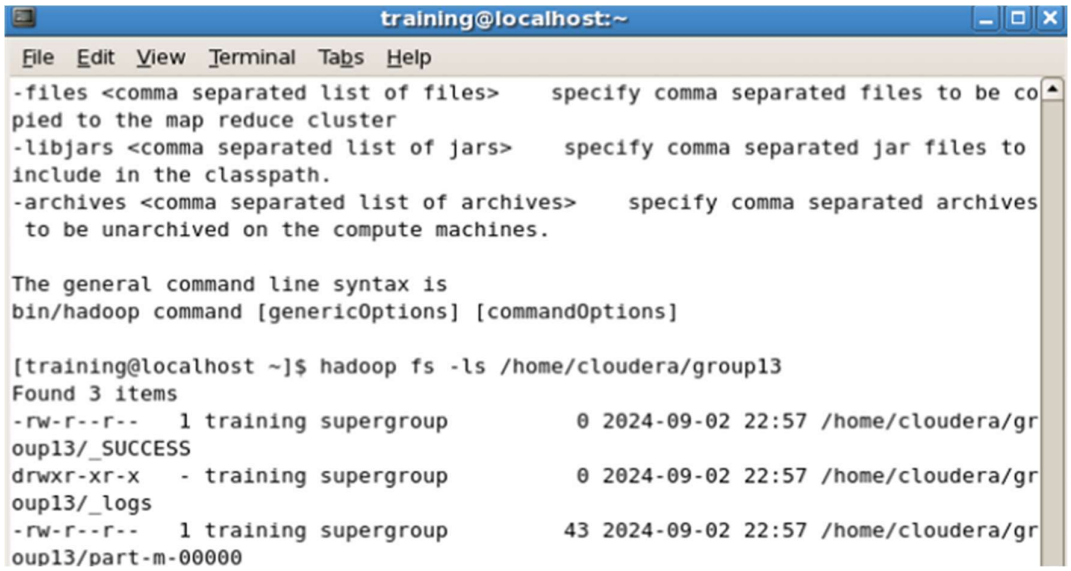

training@localhost:~
File Edit View Terminal Tabs Help
java.io.IOException: Destination '/home/training/./register.java' already exists
at org.apache.commons.io.FileUtils.moveFile(FileUtils.java:1811)
at com.cloudera.sqoop.orm.CompilationManager.compile(CompilationManager.
java:229)
at com.cloudera.sqoop.tool.CodeGenTool.generateORM(CodeGenTool.java:85)
at com.cloudera.sqoop.tool.ImportTool.importTable(ImportTool.java:369)
at com.cloudera.sqoop.tool.ImportTool.run(ImportTool.java:455)
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)
24/09/02 22:57:37 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-trai
ning/compile/e65f7eff3f8b83b64b959924ef5a9172/register.jar
24/09/02 22:57:37 WARN manager.MySQLManager: It looks like you are importing fro
m mysql.
24/09/02 22:57:37 WARN manager.MySQLManager: This transfer can be faster! Use th
e --direct
24/09/02 22:57:37 WARN manager.MySQLManager: option to exercise a MySQL-specific
fast path.
24/09/02 22:57:37 INFO manager.MySQLManager: Setting zero DATETIME behavior to c
onvertToNull (mysql)
24/09/02 22:57:37 INFO mapreduce.ImportJobBase: Beginning import of register


training@localhost:~
File Edit View Terminal Tabs Help
24/09/02 22:57:43 INFO mapred.JobClient: SLOTS_MILLIS_REDUCES=0
24/09/02 22:57:43 INFO mapred.JobClient: FileSystemCounters
24/09/02 22:57:43 INFO mapred.JobClient: HDFS_BYTES_READ=87
24/09/02 22:57:43 INFO mapred.JobClient: FILE_BYTES_WRITTEN=65845
24/09/02 22:57:43 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=43
24/09/02 22:57:43 INFO mapred.JobClient: Map-Reduce Framework
24/09/02 22:57:43 INFO mapred.JobClient: Map input records=4
24/09/02 22:57:43 INFO mapred.JobClient: Spilled Records=0
24/09/02 22:57:43 INFO mapred.JobClient: Map output records=4
24/09/02 22:57:43 INFO mapred.JobClient: SPLIT_RAW_BYTES=87
24/09/02 22:57:43 INFO mapreduce.ImportJobBase: Transferred 43 bytes in 5.6765 s
econds (7.5751 bytes/sec)
24/09/02 22:57:43 INFO mapreduce.ImportJobBase: Retrieved 4 records.


training@localhost:~
File Edit View Terminal Tabs Help
at com.cloudera.sqoop.Sqoop.main(Sqoop.java:239)
24/09/02 22:57:37 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-trai
ning/compile/e65f7eff3f8b83b64b959924ef5a9172/register.jar
24/09/02 22:57:37 WARN manager.MySQLManager: It looks like you are importing fro
m mysql.
24/09/02 22:57:37 WARN manager.MySQLManager: This transfer can be faster! Use th
e --direct
24/09/02 22:57:37 WARN manager.MySQLManager: option to exercise a MySQL-specific
fast path.
24/09/02 22:57:37 INFO manager.MySQLManager: Setting zero DATETIME behavior to c
onvertToNull (mysql)
24/09/02 22:57:37 INFO mapreduce.ImportJobBase: Beginning import of register
24/09/02 22:57:38 INFO mapred.JobClient: Running job: job_202409022156_0002
24/09/02 22:57:39 INFO mapred.JobClient: map 0% reduce 0%
24/09/02 22:57:43 INFO mapred.JobClient: map 100% reduce 0%
24/09/02 22:57:43 INFO mapred.JobClient: Job complete: job_202409022156_0002
24/09/02 22:57:43 INFO mapred.JobClient: Counters: 12
24/09/02 22:57:43 INFO mapred.JobClient: Job Counters
24/09/02 22:57:43 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=3643
24/09/02 22:57:43 INFO mapred.JobClient: Total time spent by all reduces wai
ting after reserving slots (ms)=0
24/09/02 22:57:43 INFO mapred.JobClient: Total time spent by all maps waitin
g after reserving slots (ms)=0
24/09/02 22:57:43 INFO mapred.JobClient: Launched map tasks=1


training@localhost:~
File Edit View Terminal Tabs Help
13 --username root --table register --target-dir=/home/cloudera/group13 -m 1
24/09/02 22:57:36 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
24/09/02 22:57:36 INFO tool.CodeGenTool: Beginning code generation
24/09/02 22:57:36 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `register` AS t LIMIT 1
24/09/02 22:57:36 INFO orm.CompilationManager: HADOOP_HOME is /usr/lib/hadoop
24/09/02 22:57:36 INFO orm.CompilationManager: Found hadoop core jar at: /usr/li
b/hadoop/hadoop-core.jar
24/09/02 22:57:37 ERROR orm.CompilationManager: Could not rename /tmp/sqoop-trai
ning/compile/e65f7eff3f8b83b64b959924ef5a9172/register.java to /home/training/./
register.java
java.io.IOException: Destination '/home/training/./register.java' already exists
at org.apache.commons.io.FileUtils.moveFile(FileUtils.java:1811)
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.ImportTool.importTable(ImportTool.java:369)
at com.cloudera.sqoop.tool.ImportTool.run(ImportTool.java:455)
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)


training@localhost:~
File Edit View Terminal Tabs Help
[-mkdir <path>]
[-setrep [-R] [-w] <rep> <path/file>]
[-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...]
run installed applications
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


training@localhost:~
File Edit View Terminal Tabs Help
-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 -ls /home/cloudera/group13
Found 3 items
-rw-r--r-- 1 training supergroup 0 2024-09-02 22:57 /home/cloudera/gr
oup13/_SUCCESS
drwxr-xr-x - training supergroup 0 2024-09-02 22:57 /home/cloudera/gr
oup13/_logs
-rw-r--r-- 1 training supergroup 43 2024-09-02 22:57 /home/cloudera/gr
oup13/part-m-00000

Experiment 3

Aim: Programming exercise on HBASE

Theory

Apache HBase is a distributed, scalable, NoSQL database designed to provide real-time read and write access to large datasets. It is modeled after Google's Bigtable and is built on top of the Hadoop Distributed File System (HDFS). HBase is capable of handling billions of rows of data spread across thousands of commodity servers and provides capabilities such as real-time random access to data, scalability, and fault tolerance.

Key Concepts of HBase

1. Column-Oriented Store:
    - Unlike traditional relational databases, which are row-oriented, HBase is a column-family-oriented store. Data is stored in tables, but tables are organized into rows and column families, with each family containing multiple columns. This allows HBase to store sparse data efficiently.
2. Tables, Rows, and Columns:
    - Table: The basic unit where data is stored, similar to a table in an RDBMS.
    - Row: Each row in an HBase table is uniquely identified by a row key. Rows in HBase are sorted by their row keys.
    - Column Family: Each row contains one or more column families. A column family is a logical grouping of columns.
    - Columns: Each column belongs to a column family, and columns are defined by a combination of the family name and column qualifier.
3. Versioning:
    - HBase supports versioning of data. Each cell (intersection of row and column) can store multiple versions of data, where each version is timestamped. This allows you to keep historical data and perform time-based queries.
4. Scalability and Distributed Architecture:
    - HBase is designed to scale horizontally by adding more nodes to the cluster. Data is automatically partitioned and distributed across multiple servers called RegionServers. Each RegionServer is responsible for serving a portion of the data (referred to as regions).

    - The master server, HBase Master, manages and assigns regions to RegionServers, handles load balancing, and performs administrative tasks.
5. HDFS Integration:
    - HBase relies on HDFS (Hadoop Distributed File System) for storage. While HBase stores data in a columnar format, the actual data files are stored in HDFS, benefiting from its replication, fault tolerance, and scalability.
6. Automatic Sharding:
    - HBase automatically shards data by splitting large tables into smaller regions. As data grows, regions are split and distributed across different RegionServers. This allows HBase to handle very large datasets efficiently.
7. Strong Consistency:
    - HBase provides strong consistency for reads and writes, meaning that once a write is confirmed, all subsequent reads will reflect that write immediately. This makes HBase suitable for real-time applications that require up-to-date data.
8. Schema-less Design:
    - HBase follows a schema-less design, meaning it does not require a predefined schema. Columns can be added on the fly without altering the entire table. This flexibility makes HBase ideal for storing semi-structured or sparse data.

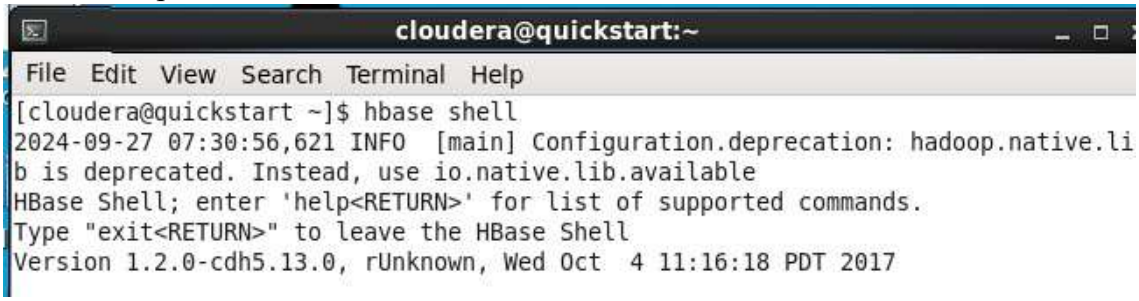
Components of HBase

1. HBase Master:
    - The HBase Master is responsible for managing the HBase cluster. It monitors RegionServers, assigns regions, and handles load balancing. The Master also handles administrative operations such as creating or deleting tables.
2. RegionServer:
    - RegionServers are responsible for handling read and write requests from clients. Each RegionServer manages a set of regions, and each region contains a portion of a table's rows. RegionServers store data in HDFS using a file format called HFile.
3. Zookeeper:
    - Apache Zookeeper is used by HBase to coordinate the cluster. Zookeeper helps with maintaining configuration information, providing

distributed synchronization, and managing leader election. It ensures the availability of the HBase cluster.

4. Regions:
    - A region is a subset of a table's rows, and each RegionServer can manage multiple regions. When a region becomes too large, it is split into two, and these new regions are assigned to different RegionServers.

Code & Output:




```
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>
```

```
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):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> 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
```

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

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

```
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>
```

```
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> 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> 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> 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
```

Experiment 4

Aim: Experiment for Word Counting using Hadoop Map-Reduce

Theory:

The Word Count problem is one of the most basic and widely used examples to explain the working of Hadoop MapReduce. It demonstrates the ability of the MapReduce framework to process and analyze large datasets in a distributed environment.

1. Introduction to Hadoop MapReduce

Hadoop MapReduce is a programming model for processing large datasets in parallel across a Hadoop cluster. It divides the work into two key phases:

- Map Phase: This phase processes input data and generates intermediate key-value pairs.
- Reduce Phase: This phase processes intermediate key-value pairs from the map phase and produces the final output.

The Word Count problem involves counting the occurrences of each word in a given text dataset. The dataset is divided and distributed across multiple nodes of a Hadoop cluster, and each node counts the words in the data assigned to it. Finally, the results from all nodes are combined to get the final word count.

2. Components of the Word Count Problem

The Word Count implementation using Hadoop MapReduce involves two main components:

- Mapper Function
- Reducer Function

3. MapReduce Workflow for Word Count

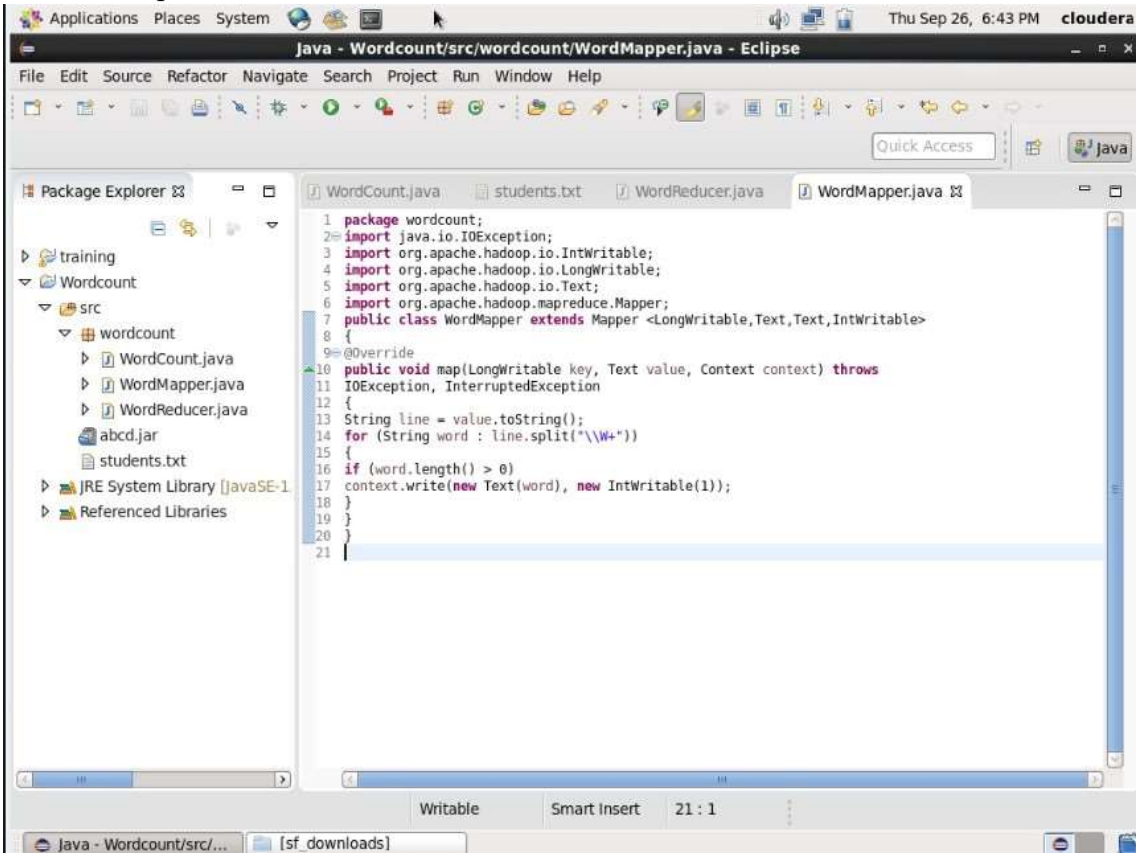
- Input: A set of text files containing sentences or words.
- Output: A set of key-value pairs where each key is a word and the value is the frequency of that word in the input dataset.

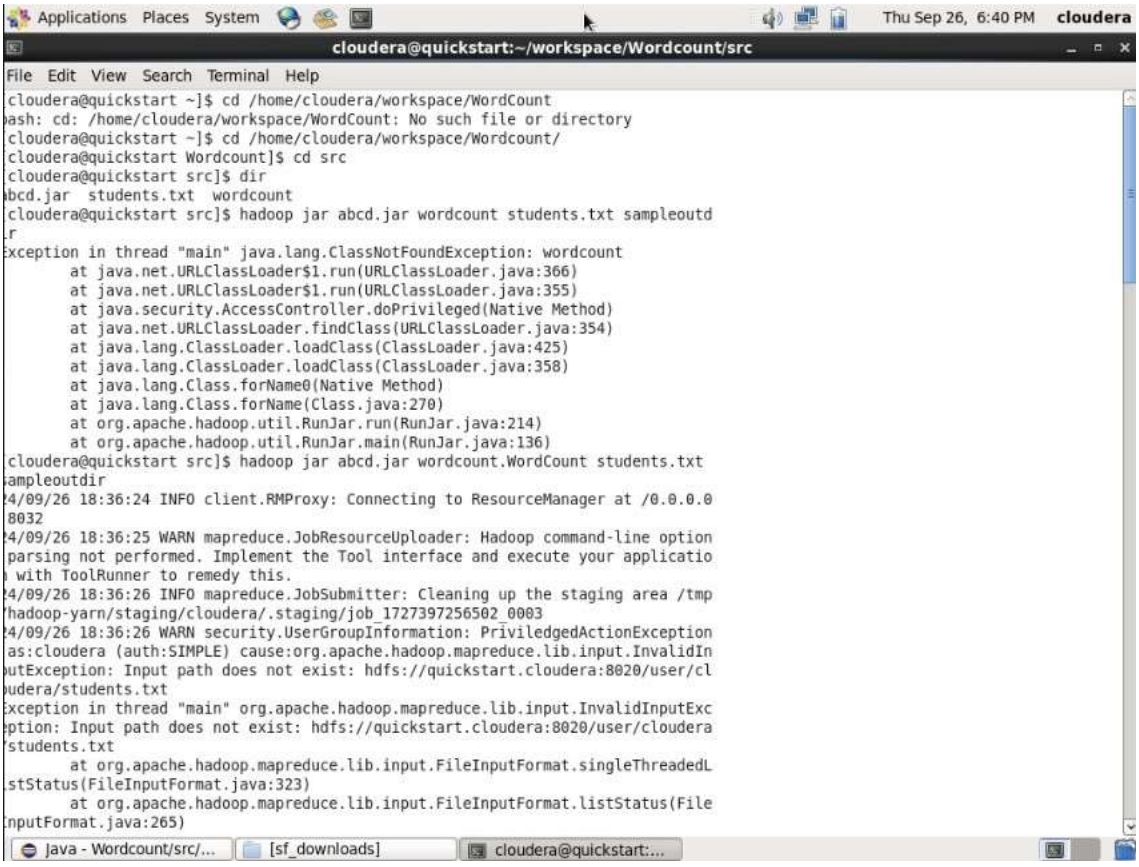
4. Phases of Word Count in Hadoop MapReduce

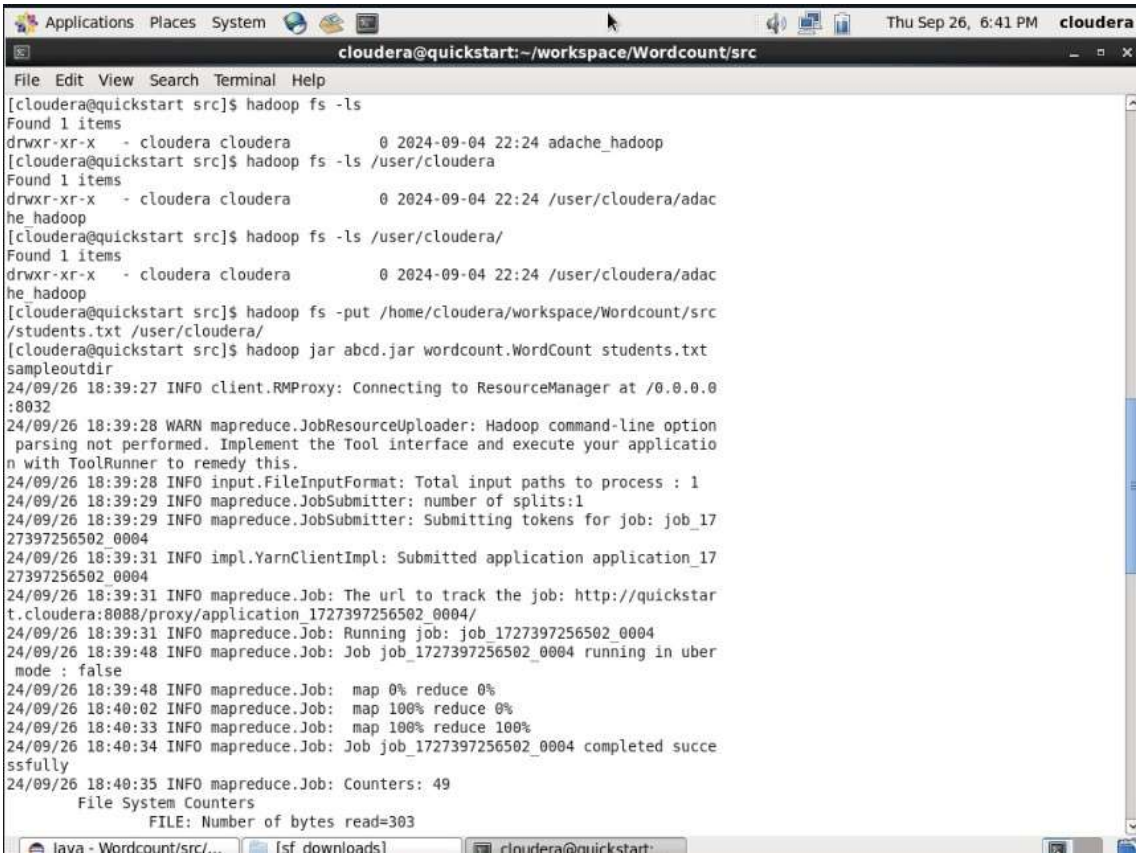
A. Map Phase

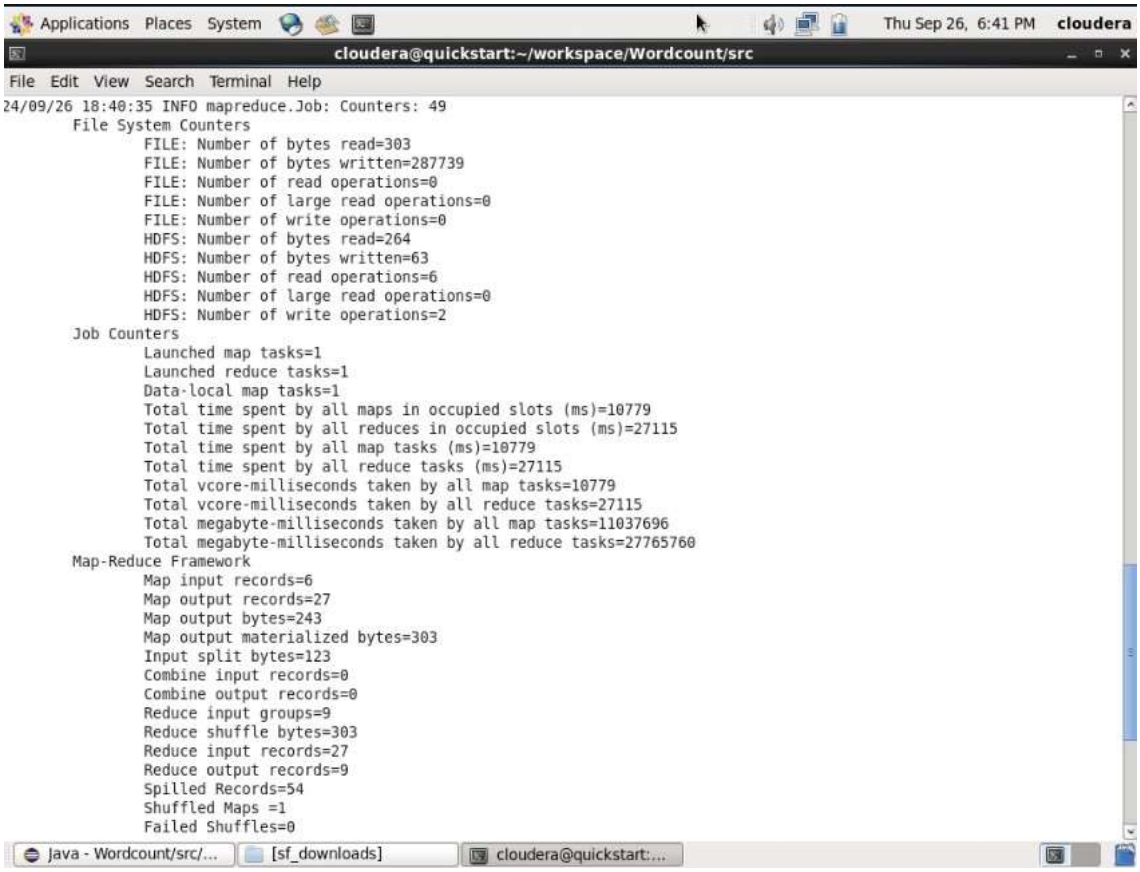
- The Mapper reads the input line by line.
- For each line, it breaks the line into words (tokenization).
- For each word, the Mapper emits a key-value pair where the word is the key, and the value is `1` (indicating one occurrence of the word).

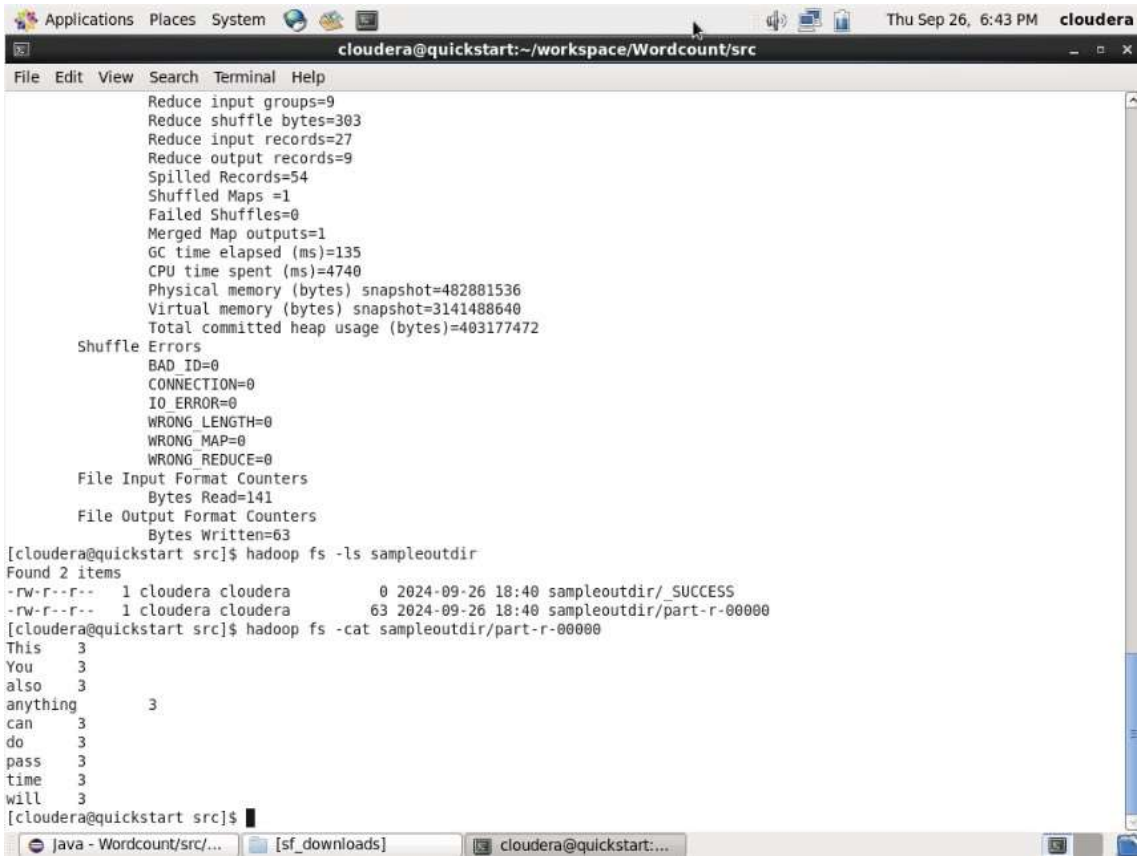
Code & Output:

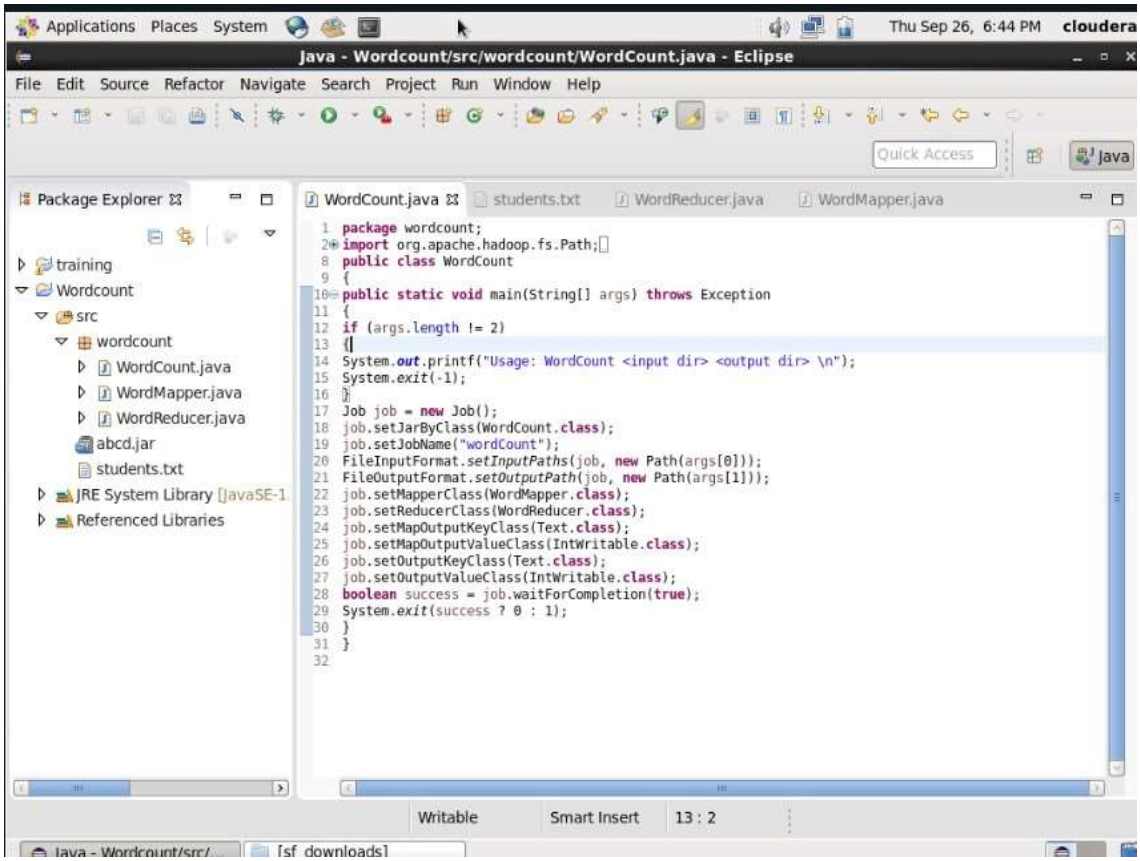


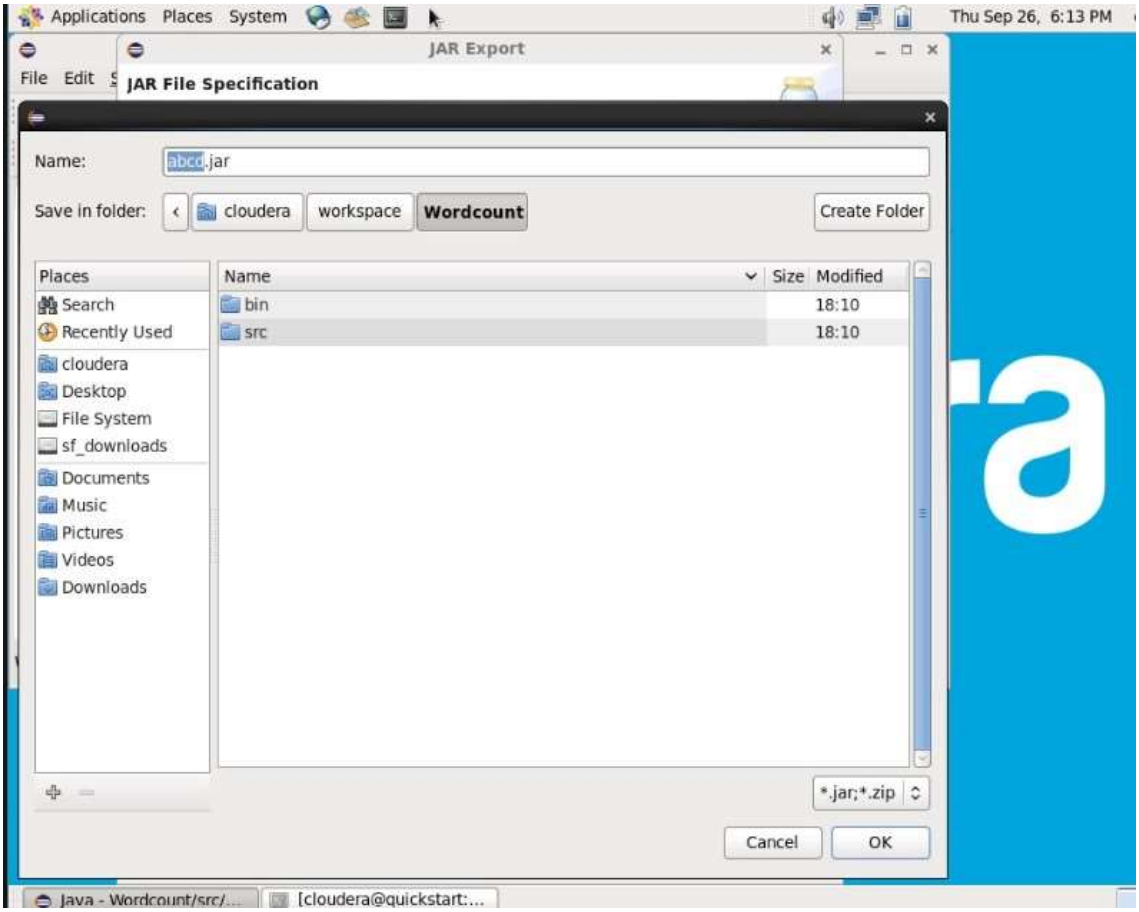

Applications Places System
Thu Sep 26, 6:40 PM cloudera
cloudera@quickstart:~/workspace/Wordcount/src
File Edit View Search Terminal Help
cloudera@quickstart ~]$ cd /home/cloudera/workspace/WordCount
ash: cd: /home/cloudera/workspace/WordCount: No such file or directory
cloudera@quickstart ~]$ cd /home/cloudera/workspace/Wordcount/
cloudera@quickstart Wordcount]$ cd src
cloudera@quickstart src]$ dir
bcd.jar students.txt wordcount
cloudera@quickstart src]$ hadoop jar abcd.jar wordcount students.txt sampleoutd
r
xception in thread "main" java.lang.ClassNotFoundException: wordcount
at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
at java.lang.Class.forName0(Native Method)
at java.lang.Class.forName(Class.java:270)
at org.apache.hadoop.util.RunJar.run(RunJar.java:214)
at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
cloudera@quickstart src]$ hadoop jar abcd.jar wordcount.WordCount students.txt
ampleoutdir
4/09/26 18:36:24 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
8032
4/09/26 18:36:25 WARN mapreduce.JobResourceUploader: Hadoop command-line option
parsing not performed. Implement the Tool interface and execute your applicatio
with ToolRunner to remedy this.
4/09/26 18:36:26 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp
hadoop-yarn/staging/cloudera/.staging/job_1727397256502_0003
4/09/26 18:36:26 WARN security.UserGroupInformation: PriviledgedActionException
as:cloudera (auth:SIMPLE) cause:org.apache.hadoop.mapreduce.lib.input.InvalidIn
utException: Input path does not exist: hdfs://quickstart.cloudera:8020/user/cl
udera/students.txt
xception in thread "main" org.apache.hadoop.mapreduce.lib.input.InvalidInputExc
ption: Input path does not exist: hdfs://quickstart.cloudera:8020/user/cloudera
students.txt
at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.singleThreadedL
stStatus(FileInputFormat.java:323)
at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.listStatus(File
nputFormat.java:265)
java - Wordcount/src/...
[sf_downloads]
cloudera@quickstart:...

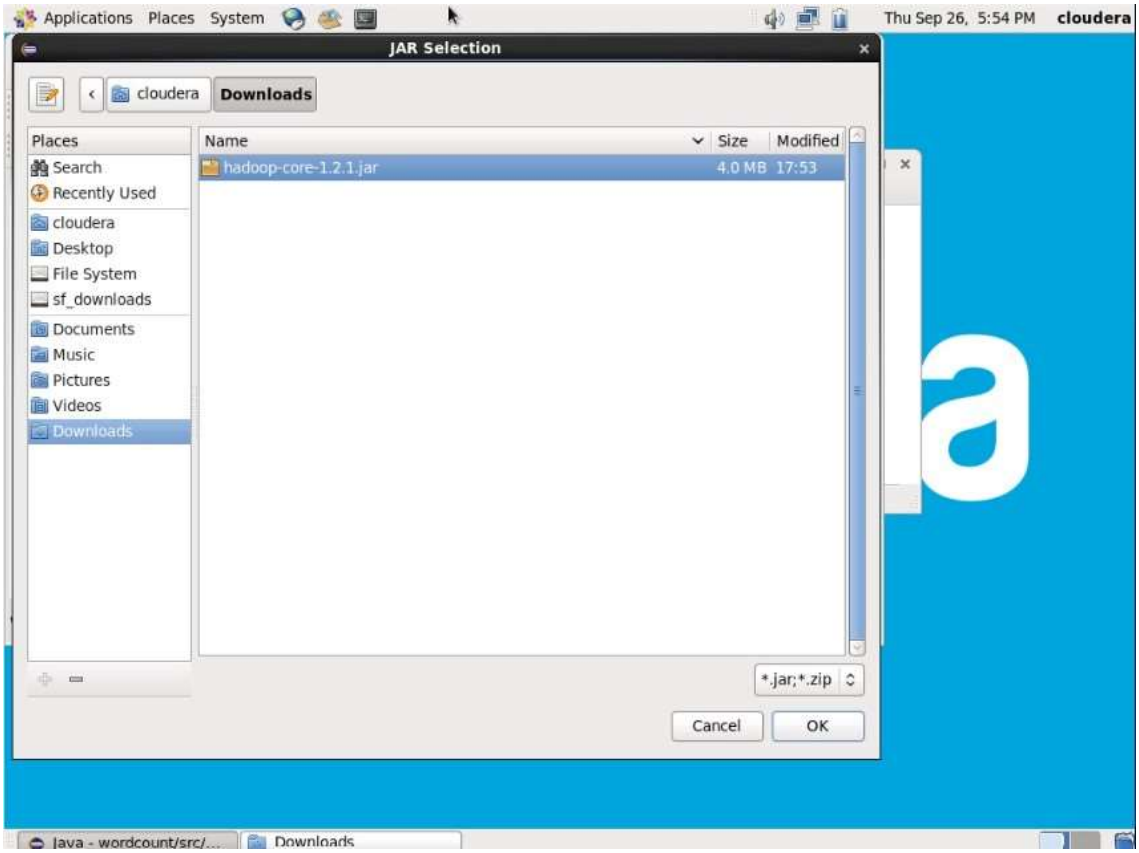

Applications Places System
Thu Sep 26, 6:41 PM cloudera
cloudera@quickstart:~/workspace/Wordcount/src
File Edit View Search Terminal Help
[cloudera@quickstart src]$ hadoop fs -ls
Found 1 items
drwxr-xr-x - cloudera cloudera 0 2024-09-04 22:24 adache_hadoop
[cloudera@quickstart src]$ hadoop fs -ls /user/cloudera
Found 1 items
drwxr-xr-x - cloudera cloudera 0 2024-09-04 22:24 /user/cloudera/adac
he_hadoop
[cloudera@quickstart src]$ hadoop fs -ls /user/cloudera/
Found 1 items
drwxr-xr-x - cloudera cloudera 0 2024-09-04 22:24 /user/cloudera/adac
he_hadoop
[cloudera@quickstart src]$ hadoop fs -put /home/cloudera/workspace/Wordcount/src
/students.txt /user/cloudera/
[cloudera@quickstart src]$ hadoop jar abcd.jar wordcount.WordCount students.txt
sampleoutdir
24/09/26 18:39:27 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
24/09/26 18:39:28 WARN mapreduce.JobResourceUploader: Hadoop command-line option
parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
24/09/26 18:39:28 INFO input.FileInputFormat: Total input paths to process : 1
24/09/26 18:39:29 INFO mapreduce.JobSubmitter: number of splits:1
24/09/26 18:39:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_17
27397256502_0004
24/09/26 18:39:31 INFO impl.YarnClientImpl: Submitted application application_17
27397256502_0004
24/09/26 18:39:31 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1727397256502_0004/
24/09/26 18:39:31 INFO mapreduce.Job: Running job: job_1727397256502_0004
24/09/26 18:39:48 INFO mapreduce.Job: Job job_1727397256502_0004 running in uber
mode : false
24/09/26 18:39:48 INFO mapreduce.Job: map 0% reduce 0%
24/09/26 18:40:02 INFO mapreduce.Job: map 100% reduce 0%
24/09/26 18:40:33 INFO mapreduce.Job: map 100% reduce 100%
24/09/26 18:40:34 INFO mapreduce.Job: Job job_1727397256502_0004 completed succe
ssfully
24/09/26 18:40:35 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=303
java - Wordcount/src/...
[sf_downloads]
cloudera@quickstart:

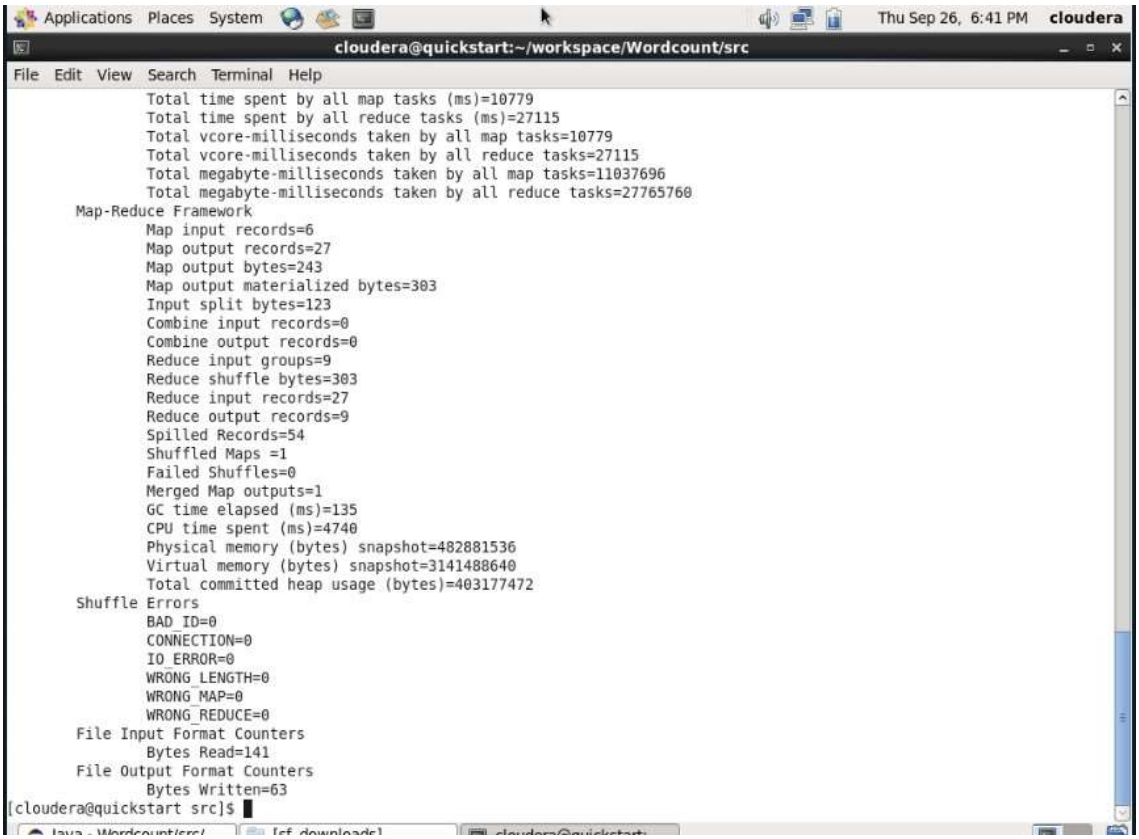

Applications Places System
Thu Sep 26, 6:41 PM cloudera
cloudera@quickstart:~/workspace/Wordcount/src
File Edit View Search Terminal Help
24/09/26 18:40:35 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=303
FILE: Number of bytes written=287739
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=264
HDFS: Number of bytes written=63
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=10779
Total time spent by all reduces in occupied slots (ms)=27115
Total time spent by all map tasks (ms)=10779
Total time spent by all reduce tasks (ms)=27115
Total vcore-milliseconds taken by all map tasks=10779
Total vcore-milliseconds taken by all reduce tasks=27115
Total megabyte-milliseconds taken by all map tasks=11037696
Total megabyte-milliseconds taken by all reduce tasks=27765760
Map-Reduce Framework
Map input records=6
Map output records=27
Map output bytes=243
Map output materialized bytes=303
Input split bytes=123
Combine input records=0
Combine output records=0
Reduce input groups=9
Reduce shuffle bytes=303
Reduce input records=27
Reduce output records=9
Spilled Records=54
Shuffled Maps =1
Failed Shuffles=0
Java - Wordcount/src/...
[sf_downloads]
cloudera@quickstart:...

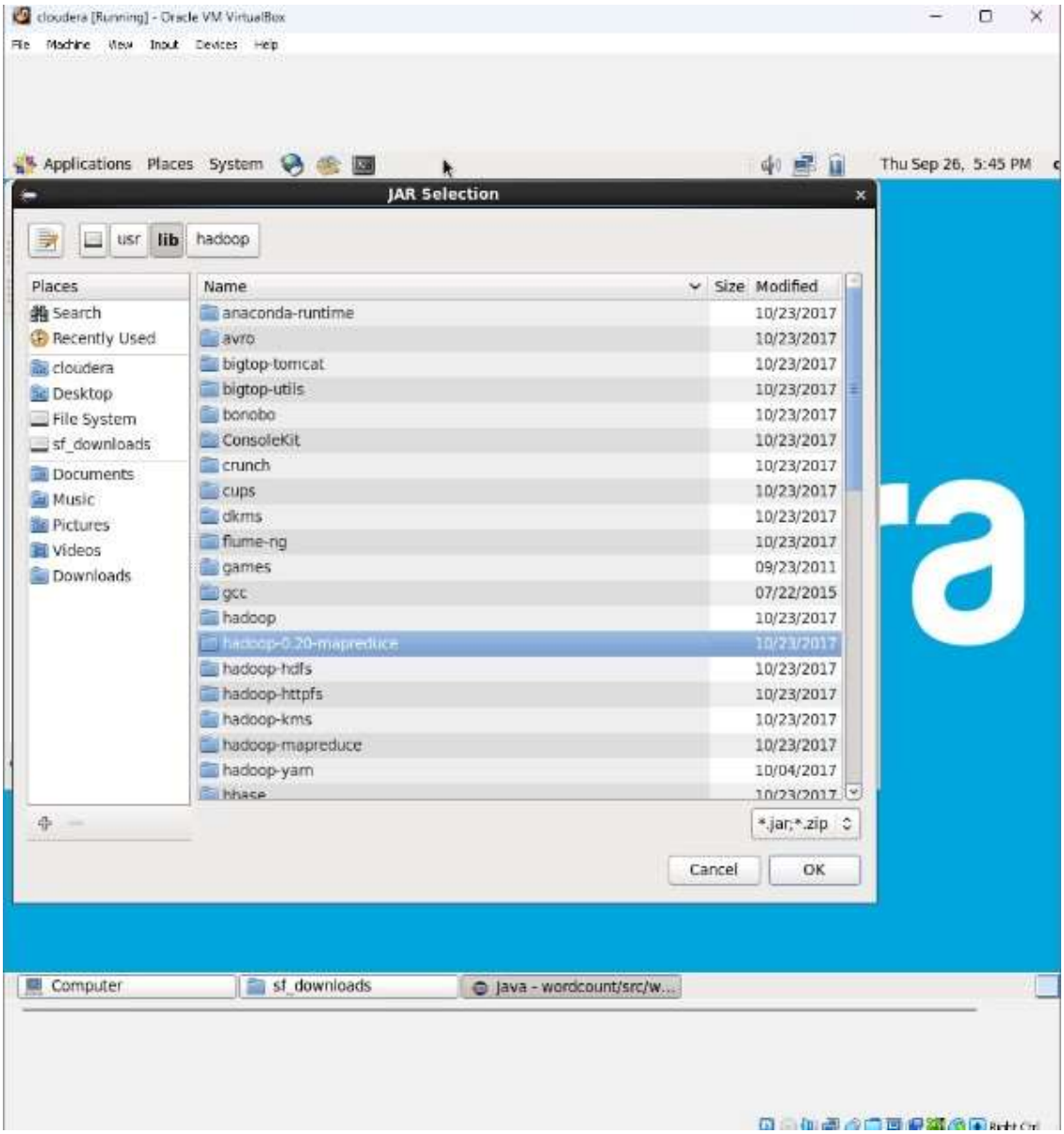

Applications Places System
Thu Sep 26, 6:43 PM cloudera
cloudera@quickstart:~/workspace/Wordcount/src
File Edit View Search Terminal Help
Reduce input groups=9
Reduce shuffle bytes=303
Reduce input records=27
Reduce output records=9
Spilled Records=54
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=135
CPU time spent (ms)=4740
Physical memory (bytes) snapshot=482881536
Virtual memory (bytes) snapshot=3141488640
Total committed heap usage (bytes)=403177472
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=141
File Output Format Counters
Bytes Written=63
[cloudera@quickstart src]$ hadoop fs -ls sampleoutdir
Found 2 items
-rw-r--r-- 1 cloudera cloudera 0 2024-09-26 18:40 sampleoutdir/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 63 2024-09-26 18:40 sampleoutdir/part-r-00000
[cloudera@quickstart src]$ hadoop fs -cat sampleoutdir/part-r-00000
This 3
You 3
also 3
anything 3
can 3
do 3
pass 3
time 3
will 3
[cloudera@quickstart src]$
Java - Wordcount/src/...
[sf_downloads]
cloudera@quickstart:...

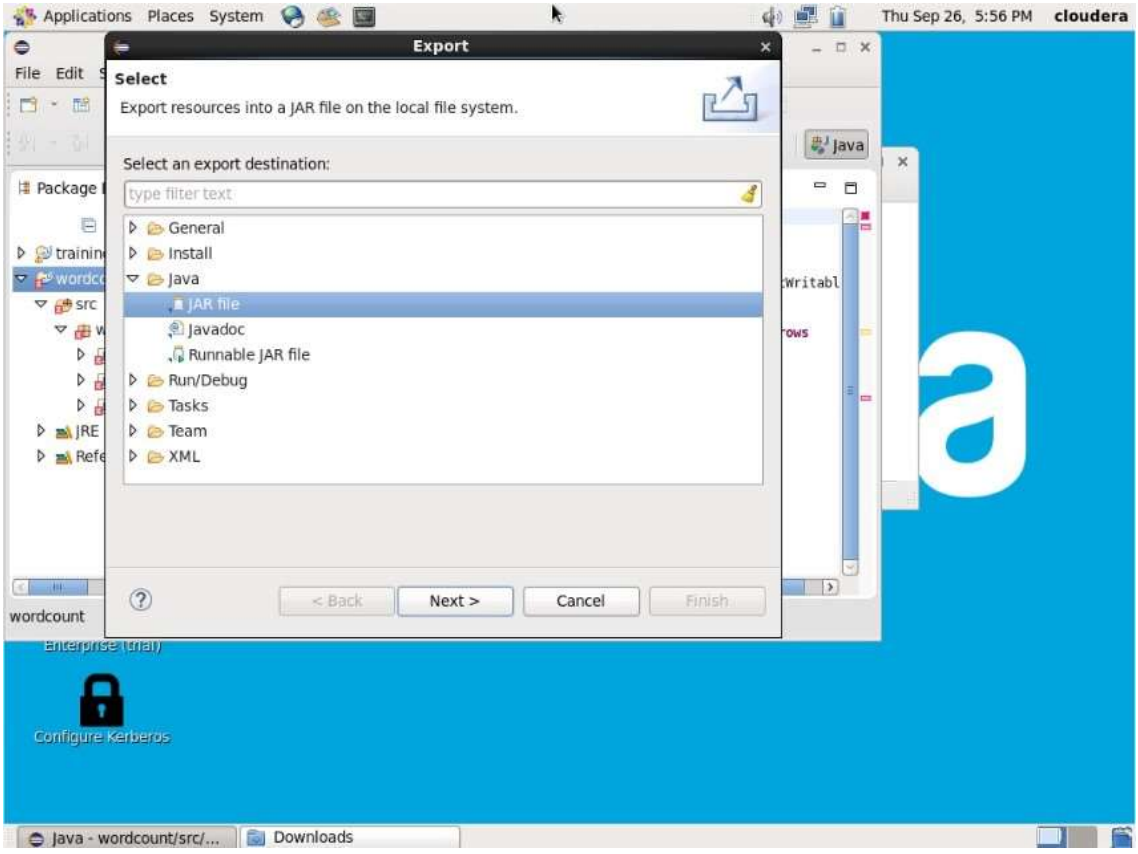

Applications Places System
Thu Sep 26, 6:44 PM cloudera
Java - Wordcount/src/wordcount/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
training
Wordcount
src
wordcount
WordCount.java
WordMapper.java
WordReducer.java
abcd.jar
students.txt
JRE System Library [JavaSE-1
Referenced Libraries
WordCount.java
students.txt
WordReducer.java
WordMapper.java
1 package wordcount;
2 import org.apache.hadoop.fs.Path;
8 public class WordCount
9 {
10 public static void main(String[] args) throws Exception
11 {
12 if (args.length != 2)
13 {
14 System.out.printf("Usage: WordCount <input dir> <output dir> \n");
15 System.exit(-1);
16 }
17 Job job = new Job();
18 job.setJarByClass(WordCount.class);
19 job.setJobName("wordCount");
20 FileInputFormat.setInputPaths(job, new Path(args[0]));
21 FileOutputFormat.setOutputPath(job, new Path(args[1]));
22 job.setMapperClass(WordMapper.class);
23 job.setReducerClass(WordReducer.class);
24 job.setMapOutputKeyClass(Text.class);
25 job.setMapOutputValueClass(IntWritable.class);
26 job.setOutputKeyClass(Text.class);
27 job.setOutputValueClass(IntWritable.class);
28 boolean success = job.waitForCompletion(true);
29 System.exit(success ? 0 : 1);
30 }
31 }
32
Writable
Smart Insert
13 : 2
Java - Wordcount/src/...
[sf_downloads]

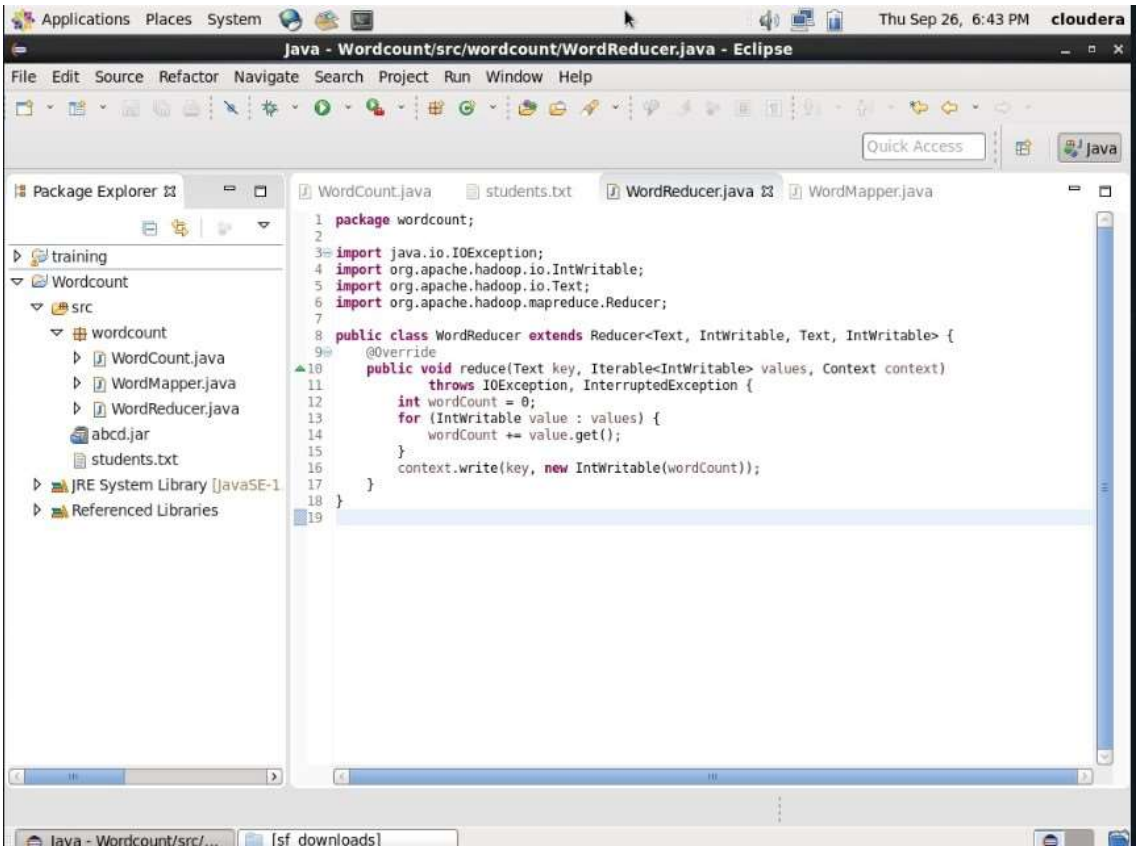

Applications
Places
System
Thu Sep 26, 6:13 PM
JAR Export
File
Edit
JAR File Specification
Name:
abcd.jar
Save in folder:
cloudera
workspace
Wordcount
Create Folder
Places
Search
Recently Used
cloudera
Desktop
File System
sf_downloads
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
bin
18:10
src
18:10
.jar;.zip
Cancel
OK
Java - Wordcount/src/...
[cloudera@quickstart:...

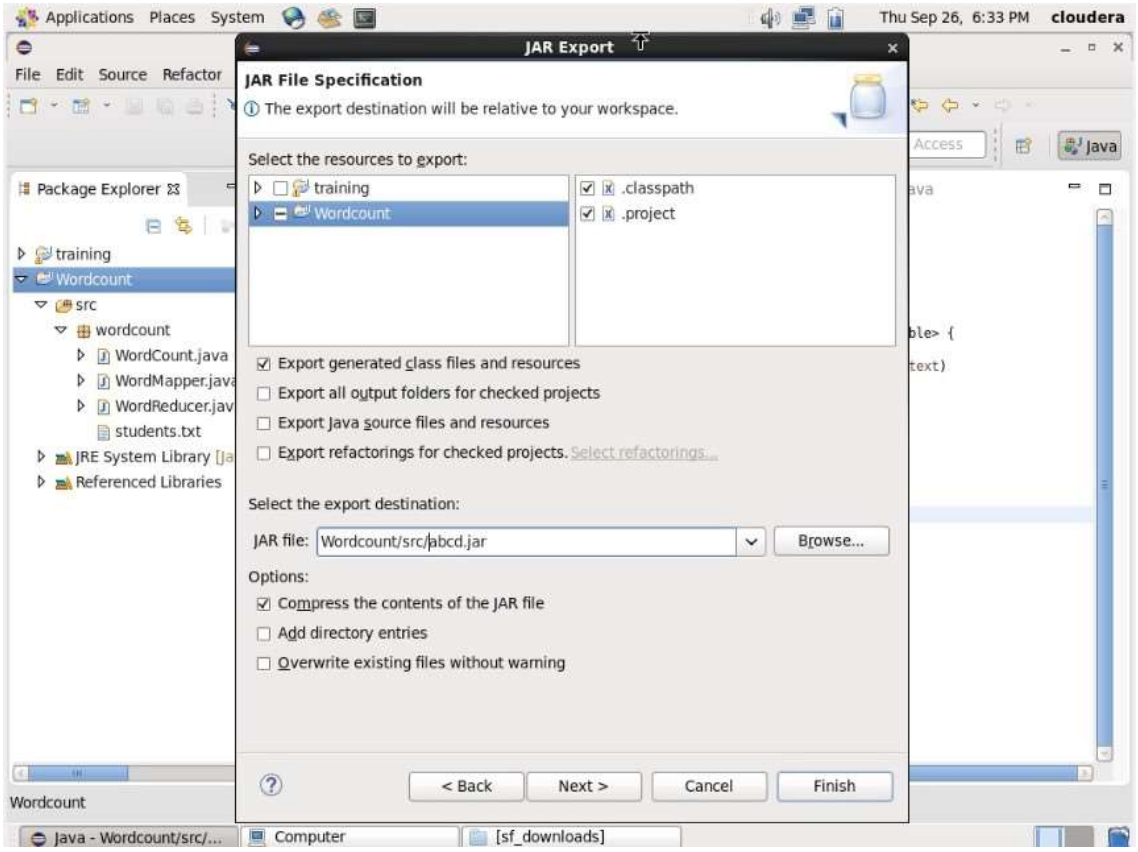

Applications Places System
Thu Sep 26, 5:54 PM cloudera
JAR Selection
cloudera
Downloads
Places
Search
Recently Used
cloudera
Desktop
File System
sf_downloads
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
hadoop-core-1.2.1.jar
4.0 MB 17:53
.jar;.zip
Cancel
OK
Downloads

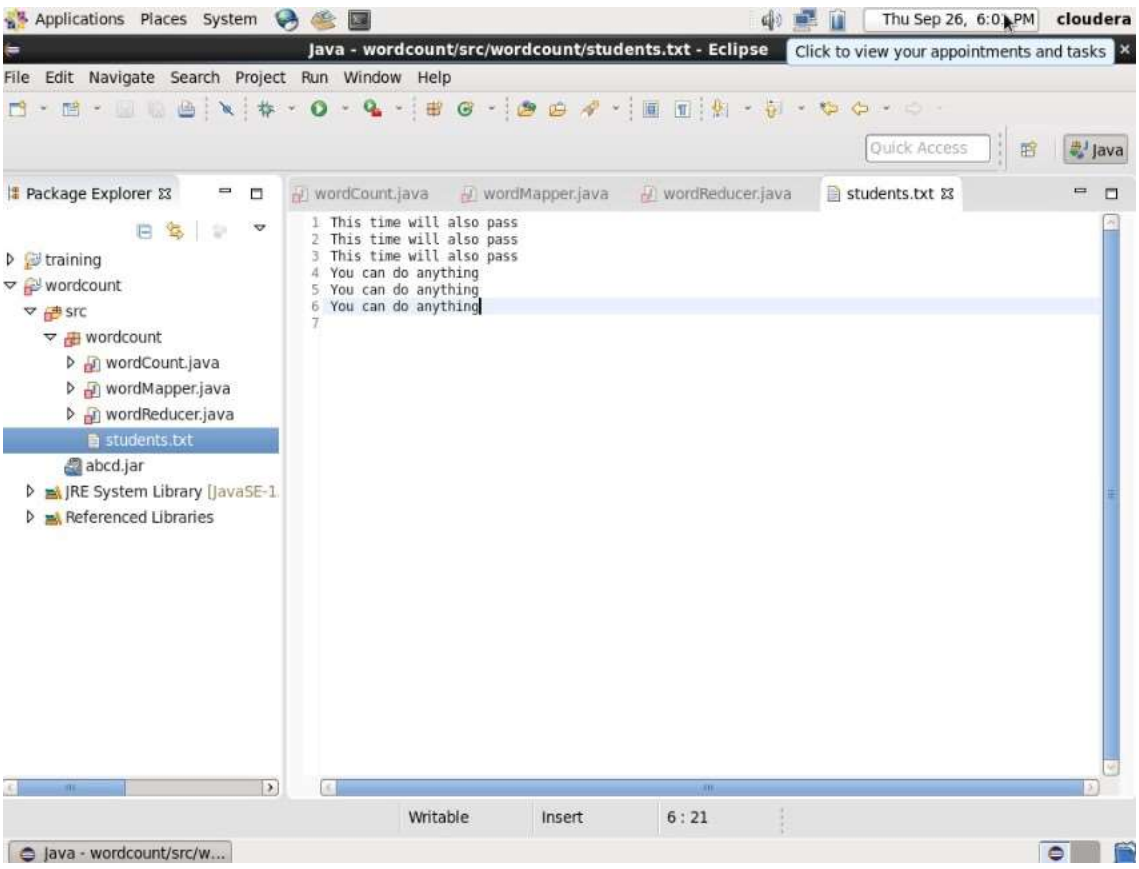

Applications Places System
Thu Sep 26, 6:41 PM cloudera
cloudera@quickstart:~/workspace/Wordcount/src
File Edit View Search Terminal Help
Total time spent by all map tasks (ms)=10779
Total time spent by all reduce tasks (ms)=27115
Total vcore-milliseconds taken by all map tasks=10779
Total vcore-milliseconds taken by all reduce tasks=27115
Total megabyte-milliseconds taken by all map tasks=11037696
Total megabyte-milliseconds taken by all reduce tasks=27765760
Map-Reduce Framework
Map input records=6
Map output records=27
Map output bytes=243
Map output materialized bytes=303
Input split bytes=123
Combine input records=0
Combine output records=0
Reduce input groups=9
Reduce shuffle bytes=303
Reduce input records=27
Reduce output records=9
Spilled Records=54
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=135
CPU time spent (ms)=4740
Physical memory (bytes) snapshot=482881536
Virtual memory (bytes) snapshot=3141488640
Total committed heap usage (bytes)=403177472
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=141
File Output Format Counters
Bytes Written=63
[cloudera@quickstart src]$

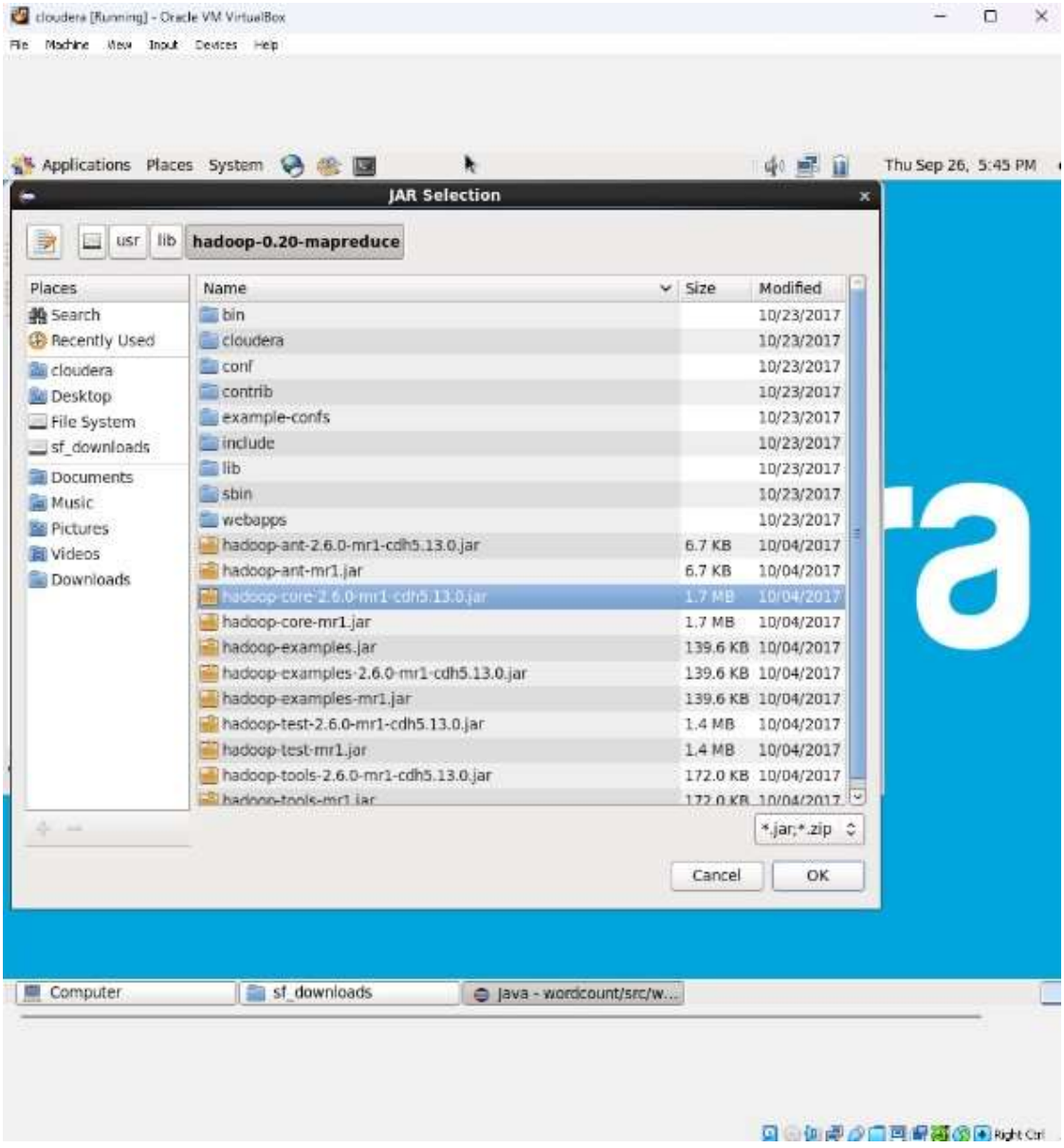

cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Thu Sep 26, 5:45 PM
JAR Selection
usr lib hadoop
Places
Search
Recently Used
cloudera
Desktop
File System
sf_downloads
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
anaconda-runtime 10/23/2017
avro 10/23/2017
bigtop-tomcat 10/23/2017
bigtop-utils 10/23/2017
bonobo 10/23/2017
ConsoleKit 10/23/2017
crunch 10/23/2017
cups 10/23/2017
dkms 10/23/2017
flume-ng 10/23/2017
games 09/23/2011
gcc 07/22/2015
hadoop 10/23/2017
hadoop-0.20-mapreduce 10/23/2017
hadoop-hdfs 10/23/2017
hadoop-httpfs 10/23/2017
hadoop-kms 10/23/2017
hadoop-mapreduce 10/23/2017
hadoop-yarn 10/04/2017
hbase 10/23/2017
.jar;.zip
Cancel
OK
Computer
sf_downloads
Java - wordcount/src/w...

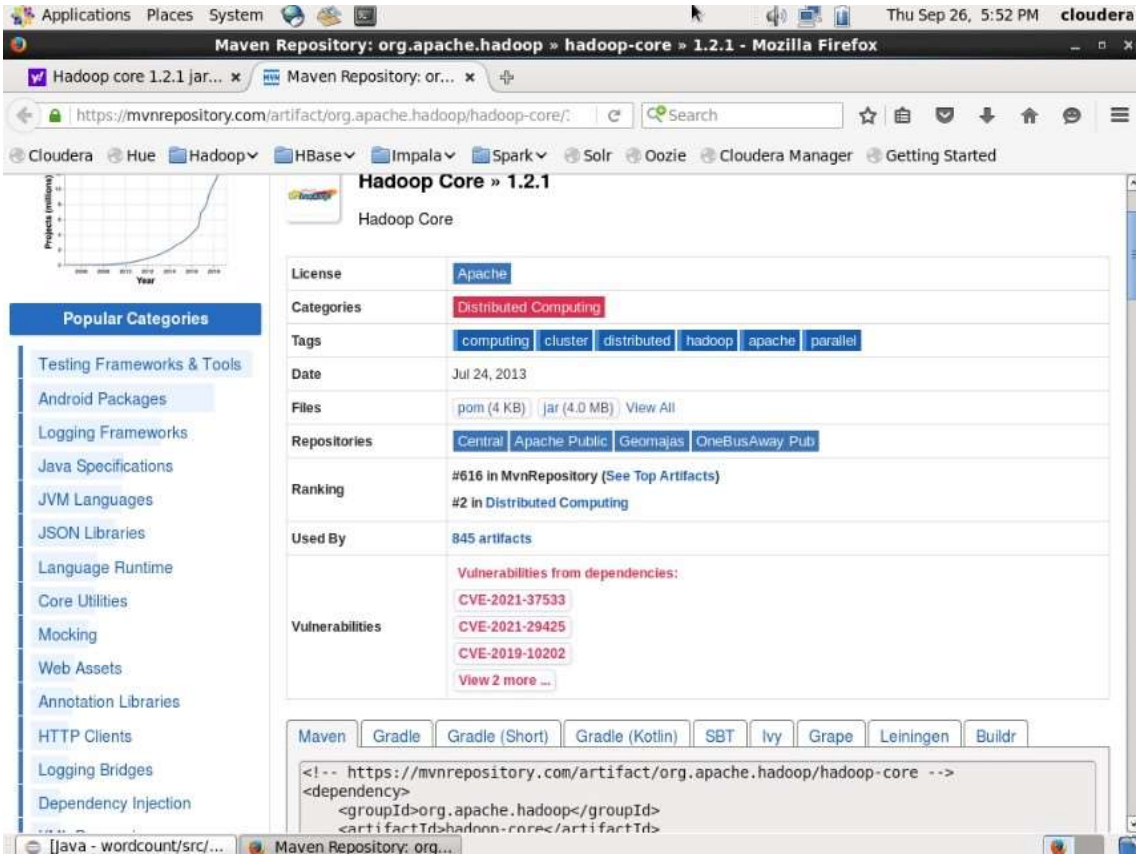

Applications Places System
Thu Sep 26, 5:56 PM cloudera
Export
Select
Export resources into a JAR file on the local file system.
Select an export destination:
type filter text
General
Install
Java
JAR file
Javadoc
Runnable JAR file
Run/Debug
Tasks
Team
XML
< Back
Next >
Cancel
Finish
wordcount
Configure Kerberos
Java - wordcount/src/...
Downloads

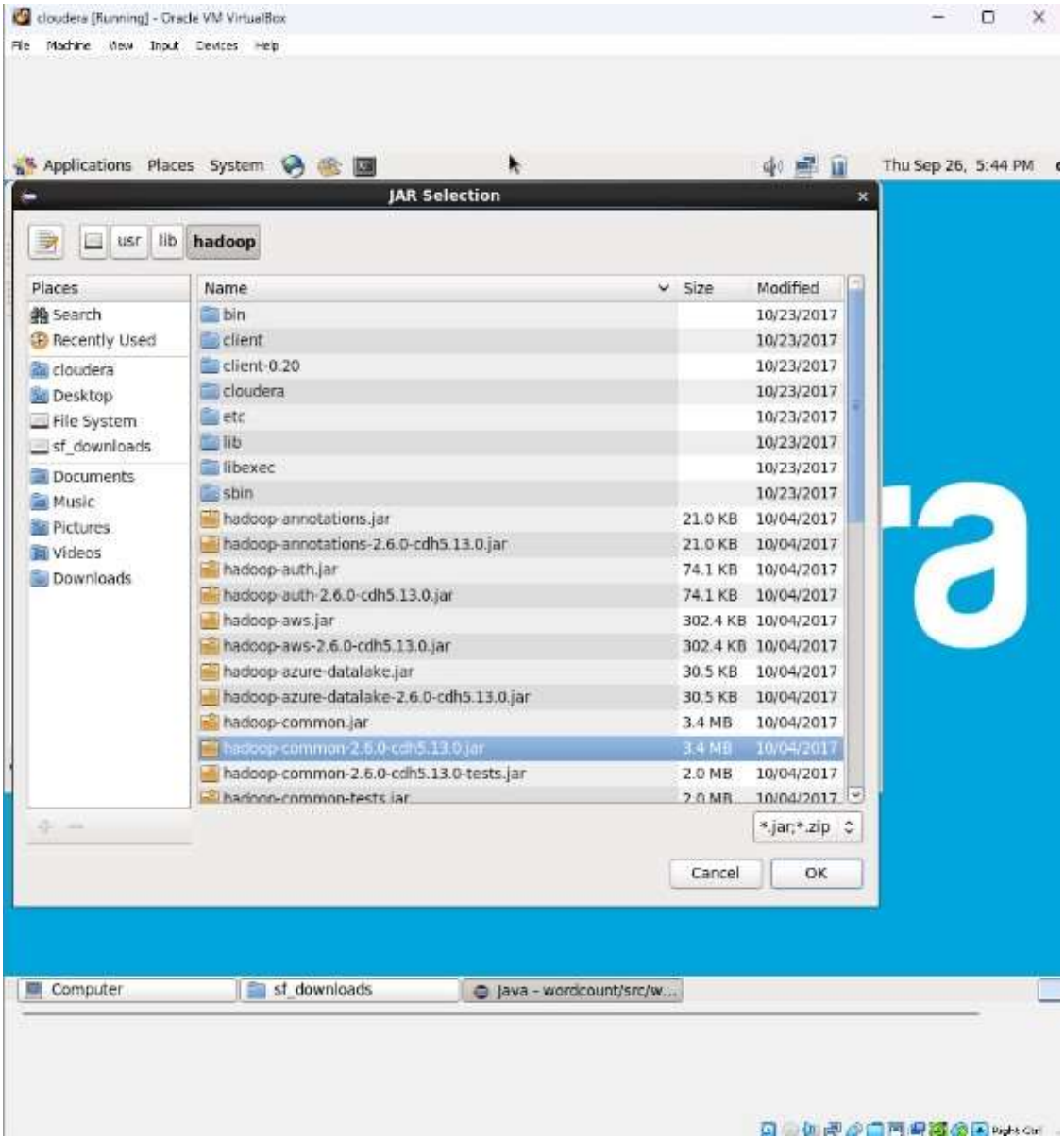

Applications Places System
Thu Sep 26, 6:43 PM cloudera
Java - Wordcount/src/wordcount/WordReducer.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
training
Wordcount
src
wordcount
WordCount.java
WordMapper.java
WordReducer.java
abcd.jar
students.txt
JRE System Library [JavaSE-1
Referenced Libraries
WordCount.java
students.txt
WordReducer.java
WordMapper.java
1 package wordcount;
2
3 import java.io.IOException;
4 import org.apache.hadoop.io.IntWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Reducer;
7
8 public class WordReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
9 @Override
10 public void reduce(Text key, Iterable<IntWritable> values, Context context)
11 throws IOException, InterruptedException {
12 int wordCount = 0;
13 for (IntWritable value : values) {
14 wordCount += value.get();
15 }
16 context.write(key, new IntWritable(wordCount));
17 }
18 }
19
Java - Wordcount/src/...
[sf_downloads]


Applications Places System
Thu Sep 26, 6:33 PM cloudera
JAR Export
JAR File Specification
The export destination will be relative to your workspace.
Select the resources to export:
training
Wordcount
.classpath
.project
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file: Wordcount/src/abcd.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Cancel
Finish
Package Explorer
training
Wordcount
src
wordcount
WordCount.java
WordMapper.java
WordReducer.java
students.txt
JRE System Library
Referenced Libraries
Java - Wordcount/src/...
Computer
[sf_downloads]


Applications Places System
Thu Sep 26, 6:01 PM cloudera
Java - wordcount/src/wordcount/students.txt - Eclipse
Click to view your appointments and tasks
File Edit Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
wordCount.java
wordMapper.java
wordReducer.java
students.txt
1 This time will also pass
2 This time will also pass
3 This time will also pass
4 You can do anything
5 You can do anything
6 You can do anything
training
wordcount
src
wordcount
wordCount.java
wordMapper.java
wordReducer.java
students.txt
abcd.jar
JRE System Library [JavaSE-1.
Referenced Libraries
Writable
Insert
6 : 21
Java - wordcount/src/w...


cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Thu Sep 26, 5:45 PM
JAR Selection
usr
lib
hadoop-0.20-mapreduce
Places
Search
Recently Used
cloudera
Desktop
File System
sf_downloads
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
bin 10/23/2017
cloudera 10/23/2017
conf 10/23/2017
contrib 10/23/2017
example-confs 10/23/2017
include 10/23/2017
lib 10/23/2017
sbin 10/23/2017
webapps 10/23/2017
hadoop-ant-2.6.0-mr1-cdh5.13.0.jar 6.7 KB 10/04/2017
hadoop-ant-mr1.jar 6.7 KB 10/04/2017
hadoop-core-2.6.0-mr1-cdh5.13.0.jar 1.7 MB 10/04/2017
hadoop-core-mr1.jar 1.7 MB 10/04/2017
hadoop-examples.jar 139.6 KB 10/04/2017
hadoop-examples-2.6.0-mr1-cdh5.13.0.jar 139.6 KB 10/04/2017
hadoop-examples-mr1.jar 139.6 KB 10/04/2017
hadoop-test-2.6.0-mr1-cdh5.13.0.jar 1.4 MB 10/04/2017
hadoop-test-mr1.jar 1.4 MB 10/04/2017
hadoop-tools-2.6.0-mr1-cdh5.13.0.jar 172.0 KB 10/04/2017
.jar;.zip
Cancel
OK
Computer
sf_downloads
Java - wordcount/src/w...
Right Ctrl


Applications Places System
Thu Sep 26, 5:52 PM cloudera
Maven Repository: org.apache.hadoop » hadoop-core » 1.2.1 - Mozilla Firefox
Hadoop core 1.2.1 jar...
Maven Repository: or...
https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Projects (millions)
Year
Popular Categories
Testing Frameworks & Tools
Android Packages
Logging Frameworks
Java Specifications
JVM Languages
JSON Libraries
Language Runtime
Core Utilities
Mocking
Web Assets
Annotation Libraries
HTTP Clients
Logging Bridges
Dependency Injection
Hadoop Core » 1.2.1
Hadoop Core
License
Apache
Categories
Distributed Computing
Tags
computing cluster distributed hadoop apache parallel
Date
Jul 24, 2013
Files
pom (4 KB) jar (4.0 MB) View All
Repositories
Central Apache Public Geomajas OneBusAway Pub
Ranking
#616 in MvnRepository (See Top Artifacts)
#2 in Distributed Computing
Used By
845 artifacts
Vulnerabilities
Vulnerabilities from dependencies:
CVE-2021-37533
CVE-2021-29425
CVE-2019-10202
View 2 more ...
Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr
<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core -->
<dependency>
<groupId>org.apache.hadoop</groupId>
[Java - wordcount/src/...
Maven Repository: org...


cloudera [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Thu Sep 26, 5:44 PM
JAR Selection
usr lib hadoop
Places
Search
Recently Used
cloudera
Desktop
File System
sf_downloads
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
bin 10/23/2017
client 10/23/2017
client-0.20 10/23/2017
cloudera 10/23/2017
etc 10/23/2017
lib 10/23/2017
libexec 10/23/2017
sbin 10/23/2017
hadoop-annotations.jar 21.0 KB 10/04/2017
hadoop-annotations-2.6.0-cdh5.13.0.jar 21.0 KB 10/04/2017
hadoop-auth.jar 74.1 KB 10/04/2017
hadoop-auth-2.6.0-cdh5.13.0.jar 74.1 KB 10/04/2017
hadoop-aws.jar 302.4 KB 10/04/2017
hadoop-aws-2.6.0-cdh5.13.0.jar 302.4 KB 10/04/2017
hadoop-azure-datalake.jar 30.5 KB 10/04/2017
hadoop-azure-datalake-2.6.0-cdh5.13.0.jar 30.5 KB 10/04/2017
hadoop-common.jar 3.4 MB 10/04/2017
hadoop-common-2.6.0-cdh5.13.0.jar 3.4 MB 10/04/2017
hadoop-common-2.6.0-cdh5.13.0-tests.jar 2.0 MB 10/04/2017
.jar;.zip
Cancel
OK
Computer
sf_downloads
Java - wordcount/src/w...

Experiment 5

Aim: Experiment on pig

Theory:

Apache Pig is a high-level platform for processing large datasets in the Hadoop ecosystem. It provides an abstraction over MapReduce and simplifies the process of writing complex data transformations. Pig enables users to write scripts using a language called Pig Latin, which is designed to handle both structured and semi-structured data. It is mainly used for data analysis, ETL (Extract, Transform, Load) tasks, and ad-hoc processing of large-scale datasets.

Key Concepts of Apache Pig

1. Pig Latin:
    - Pig Latin is a high-level data flow language used to define data analysis programs in Pig. It is more intuitive than writing complex MapReduce jobs directly, as it abstracts much of the underlying computation. Pig Latin supports operations like filtering, joining, sorting, grouping, and aggregation.
2. Pig Execution Modes:
    - Local Mode: Pig runs on a single machine, processing data stored in the local filesystem. This mode is useful for small datasets or development purposes.
    - MapReduce Mode: Pig runs on a Hadoop cluster, distributing jobs across nodes to process large datasets stored in HDFS (Hadoop Distributed File System).
3. Data Model:
    - Pig has a flexible, nested data model that supports complex, semi-structured data. The main data types are:
        - Atom: A single value (e.g., string, integer).
        - Tuple: A record of multiple fields.
        - Bag: A collection of tuples (like a table in a database).
        - Map: A collection of key-value pairs, useful for handling semi-structured data like JSON.
4. Pig Architecture:
    - Pig Scripts: Users write Pig scripts in Pig Latin to specify data transformations.

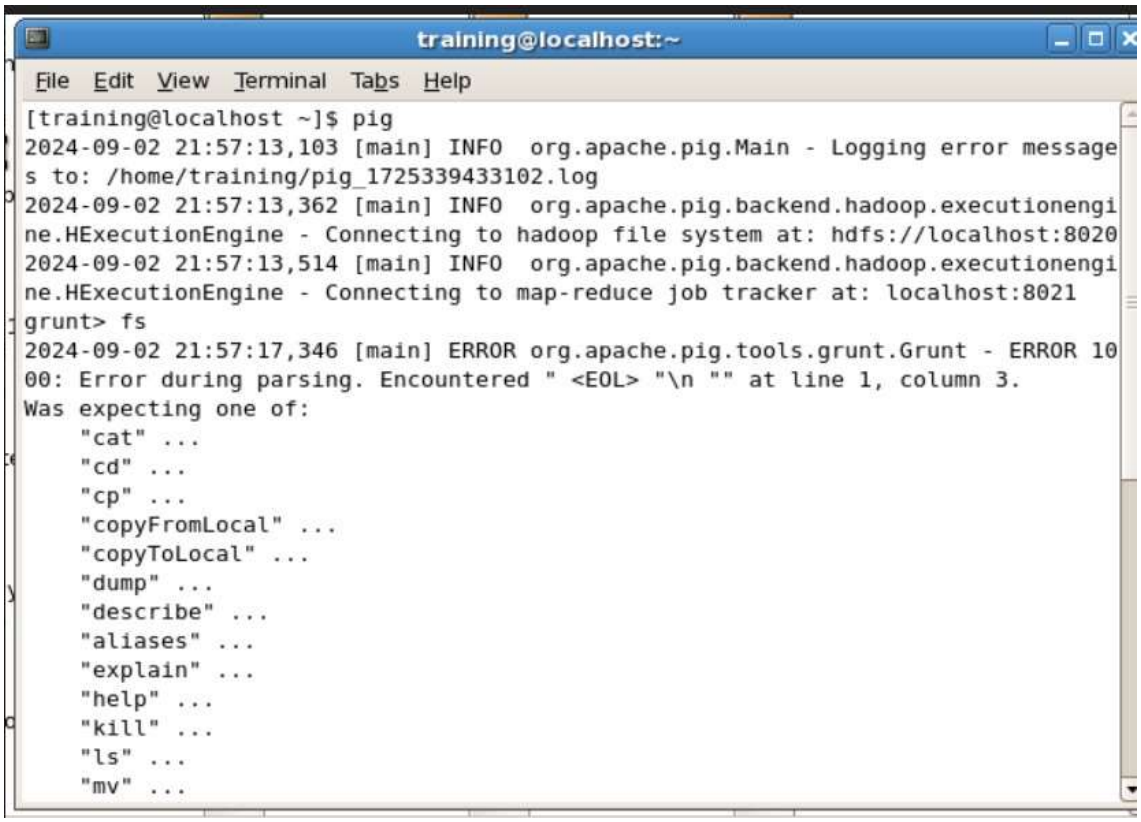
- Parser: The Pig script is parsed and checked for syntax and logical errors.
- Optimizer: The parsed script is optimized by reordering operations to improve efficiency.
- Compiler: The optimized Pig Latin script is compiled into a series of MapReduce jobs (or other execution engines like Tez or Spark).
- Execution Engine: The compiled MapReduce jobs are executed on a Hadoop cluster, or in local mode, depending on the environment.

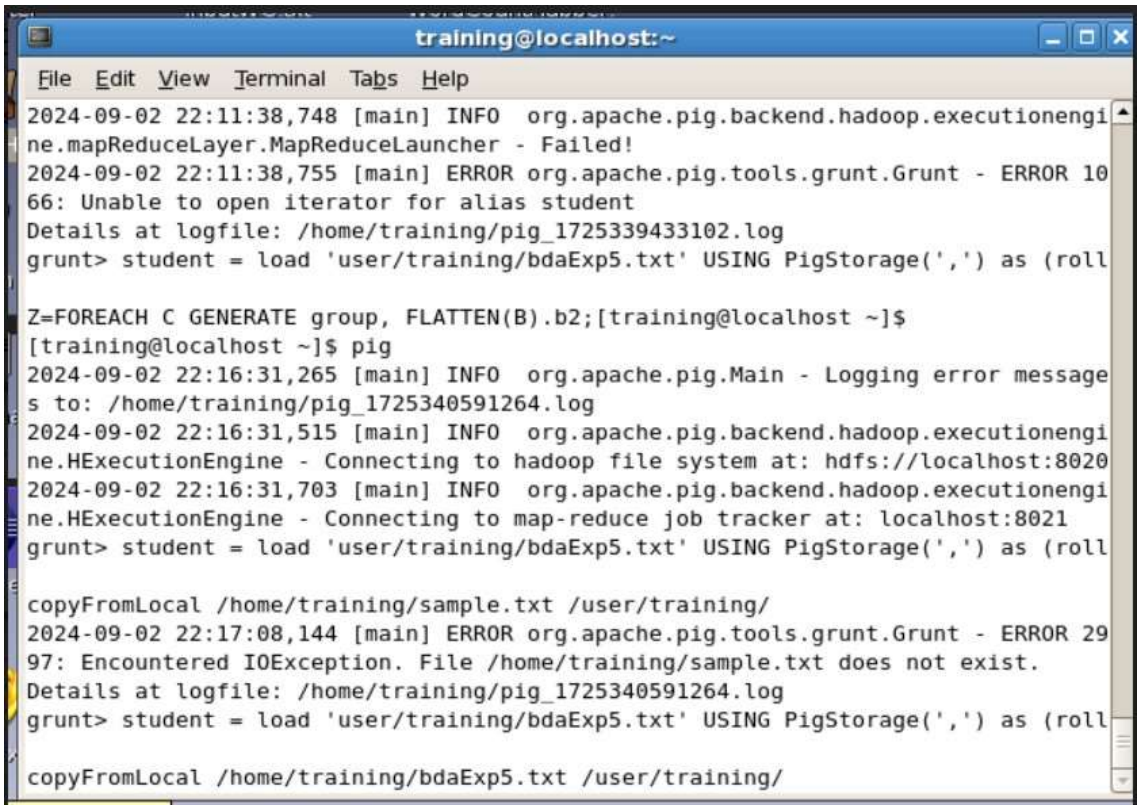
5. UDF (User Defined Functions):
    - Pig supports UDFs, allowing users to write custom functions in Java, Python, or other languages. These functions extend Pig's functionality by enabling users to implement specific data transformations, which are not available in native Pig Latin commands.

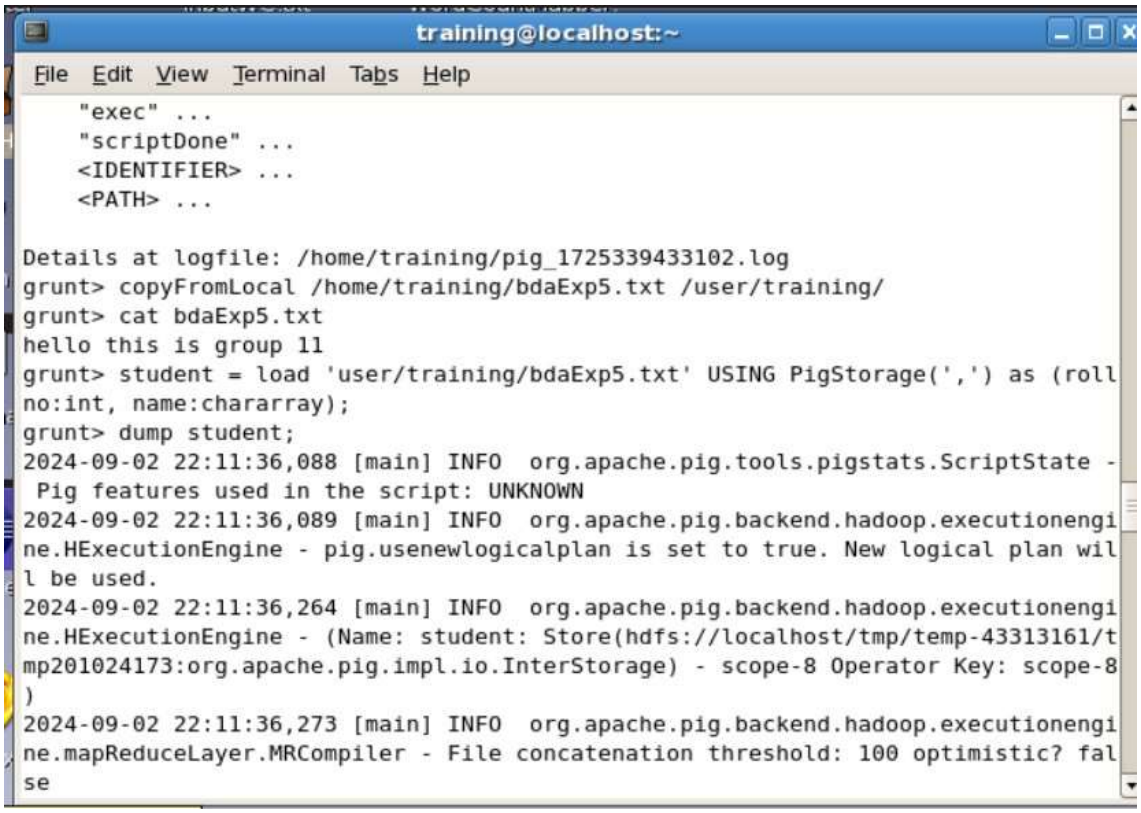
Features of Apache Pig

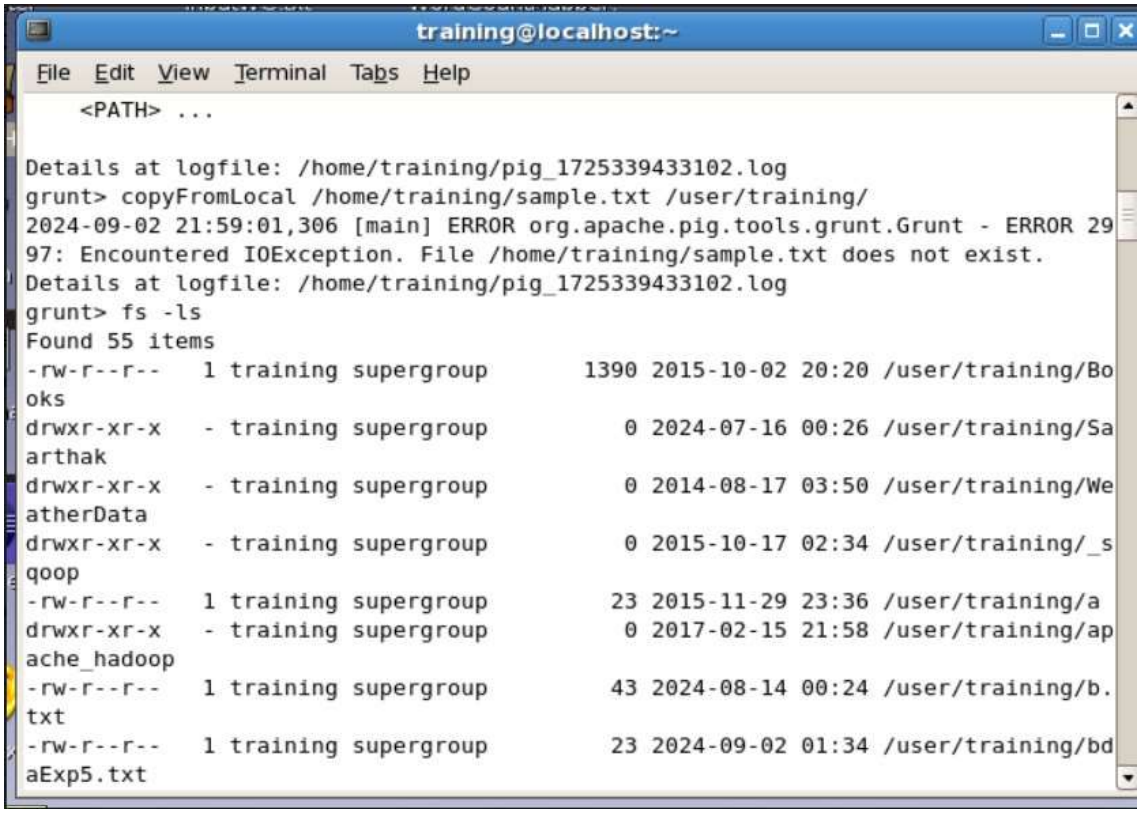
1. Ease of Use:
    - Pig Latin simplifies writing data transformation tasks compared to writing native MapReduce code. It allows users to work at a higher level of abstraction, focusing on the logic rather than the underlying MapReduce implementation.
2. Schema Flexibility:
    - Pig can process both structured and semi-structured data. It does not enforce schemas at the time of data loading, making it suitable for working with data from diverse sources.
3. Fault Tolerance:
    - Since Pig runs on top of Hadoop, it inherits Hadoop's fault tolerance. If a node fails during execution, Hadoop automatically restarts the failed task on another node, ensuring that the job completes.
4. Extensibility:
    - Pig can be easily extended through UDFs, enabling users to incorporate custom processing logic for advanced tasks that cannot be achieved using Pig's built-in operators.
5. Interoperability with Hadoop:
    - Pig is tightly integrated with the Hadoop ecosystem, allowing it to read from and write to HDFS, and work with Hadoop-based data stores like HBase and Hive.

Code & Output:


training@localhost:~
File Edit View Terminal Tabs Help
[training@localhost ~]$ pig
2024-09-02 21:57:13,103 [main] INFO org.apache.pig.Main - Logging error message
s to: /home/training/pig_1725339433102.log
2024-09-02 21:57:13,362 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2024-09-02 21:57:13,514 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
grunt> fs
2024-09-02 21:57:17,346 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 10
00: 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" ...


training@localhost:~
File Edit View Terminal Tabs Help
2024-09-02 22:11:38,748 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Failed!
2024-09-02 22:11:38,755 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 10
66: Unable to open iterator for alias student
Details at logfile: /home/training/pig_1725339433102.log
grunt> student = load 'user/training/bdaExp5.txt' USING PigStorage(',') as (roll
Z=FOREACH C GENERATE group, FLATTEN(B).b2;[training@localhost ~]$
[training@localhost ~]$ pig
2024-09-02 22:16:31,265 [main] INFO org.apache.pig.Main - Logging error message
s to: /home/training/pig_1725340591264.log
2024-09-02 22:16:31,515 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2024-09-02 22:16:31,703 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
grunt> student = load 'user/training/bdaExp5.txt' USING PigStorage(',') as (roll
copyFromLocal /home/training/sample.txt /user/training/
2024-09-02 22:17:08,144 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 29
97: Encountered IOException. File /home/training/sample.txt does not exist.
Details at logfile: /home/training/pig_1725340591264.log
grunt> student = load 'user/training/bdaExp5.txt' USING PigStorage(',') as (roll
copyFromLocal /home/training/bdaExp5.txt /user/training/


training@localhost:~
File Edit View Terminal Tabs Help
"exec" ...
"scriptDone" ...
<IDENTIFIER> ...
<PATH> ...
Details at logfile: /home/training/pig_1725339433102.log
grunt> copyFromLocal /home/training/bdaExp5.txt /user/training/
grunt> cat bdaExp5.txt
hello this is group 11
grunt> student = load 'user/training/bdaExp5.txt' USING PigStorage(',') as (roll
no:int, name:chararray);
grunt> dump student;
2024-09-02 22:11:36,088 [main] INFO org.apache.pig.tools.pigstats.ScriptState -
Pig features used in the script: UNKNOWN
2024-09-02 22:11:36,089 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - pig.usenewlogicalplan is set to true. New logical plan wil
l be used.
2024-09-02 22:11:36,264 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - (Name: student: Store(hdfs://localhost/tmp/temp-43313161/t
mp201024173:org.apache.pig.impl.io.InterStorage) - scope-8 Operator Key: scope-8
)
2024-09-02 22:11:36,273 [main] INFO org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? fal
se


training@localhost:~
File Edit View Terminal Tabs Help
<PATH> ...
Details at logfile: /home/training/pig_1725339433102.log
grunt> copyFromLocal /home/training/sample.txt /user/training/
2024-09-02 21:59:01,306 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 29
97: Encountered IOException. File /home/training/sample.txt does not exist.
Details at logfile: /home/training/pig_1725339433102.log
grunt> fs -ls
Found 55 items
-rw-r--r-- 1 training supergroup 1390 2015-10-02 20:20 /user/training/Bo
oks
drwxr-xr-x - training supergroup 0 2024-07-16 00:26 /user/training/Sa
arthak
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 43 2024-08-14 00:24 /user/training/b.
txt
-rw-r--r-- 1 training supergroup 23 2024-09-02 01:34 /user/training/bd
aExp5.txt

Experiment 6

Aim: Create HIVE Database and Descriptive analytics (basic statistics)

Theory:

Apache Hive is a data warehouse infrastructure built on top of Hadoop, designed for querying and managing large datasets stored in Hadoop's HDFS (Hadoop Distributed File System). It provides an abstraction layer over Hadoop that allows users to execute SQL-like queries (HiveQL) to manipulate and analyze big data. Below are some key concepts and components associated with Hive:

Key Concepts

1. Data Warehouse Infrastructure:
    - Hive facilitates the organization and querying of large datasets, providing features similar to traditional data warehouses but optimized for distributed data processing.
2. HiveQL:
    - Hive uses its own SQL-like query language called HiveQL (or HQL), which allows users to perform data analysis without needing to understand complex MapReduce code. HiveQL supports operations like SELECT, JOIN, and GROUP BY.
3. Schema on Read:
    - In Hive, the schema is applied at the time of data reading (query time), rather than at data writing (load time). This means you can store unstructured data and define its schema later when you run queries.
4. Metastore:
    - Hive uses a metastore to store metadata about the tables and data stored in HDFS. The metastore holds information like table schemas, column types, and partition information. Users can access this metadata through Hive's command-line interface or JDBC/ODBC connections.
5. Data Storage:
    - Data in Hive can be stored in various formats, including plain text, ORC (Optimized Row Columnar), Parquet, and Avro. These formats help optimize storage efficiency and query performance.
6. Partitioning and Bucketing:
    - Partitioning: Hive tables can be partitioned based on column values (e.g., date, region). This helps improve query performance by enabling Hive to read only the relevant partitions of the data.

    - Bucketing: This technique allows for further subdividing data within a partition into smaller, more manageable files (buckets), improving query performance and simplifying operations like joins.
7. Execution Engine:
    - Hive translates HiveQL queries into a series of MapReduce jobs that run on Hadoop. However, with the introduction of Tez and Spark, users can choose different execution engines for better performance.

Components of Hive

1. Hive CLI:
    - The Hive Command Line Interface (CLI) allows users to interact with the Hive system by executing HiveQL commands directly.
2. HiveServer2:
    - HiveServer2 provides a more robust and efficient service for client applications to execute queries against Hive. It supports multiple sessions, authentication, and JDBC/ODBC connections.
3. Thrift:
    - Apache Thrift is used to allow different programming languages to communicate with Hive, enabling developers to build applications in languages like Python, Java, and C++.
4. Hive Web Interface:
    - Some distributions of Hive include a web-based interface for managing queries and browsing data.

Use Cases

- Data Analysis: Hive is widely used for batch processing and data analysis tasks, making it suitable for business intelligence applications.
- ETL Operations: Hive can be used in Extract, Transform, Load (ETL) processes to prepare data for further analysis.
- Log Analysis: Organizations often use Hive to analyze large volumes of log data generated by applications, servers, or IoT devices.
- Data Aggregation: Hive is effective in performing aggregations over large datasets, enabling users to summarize and report on data.

Advantages of Hive

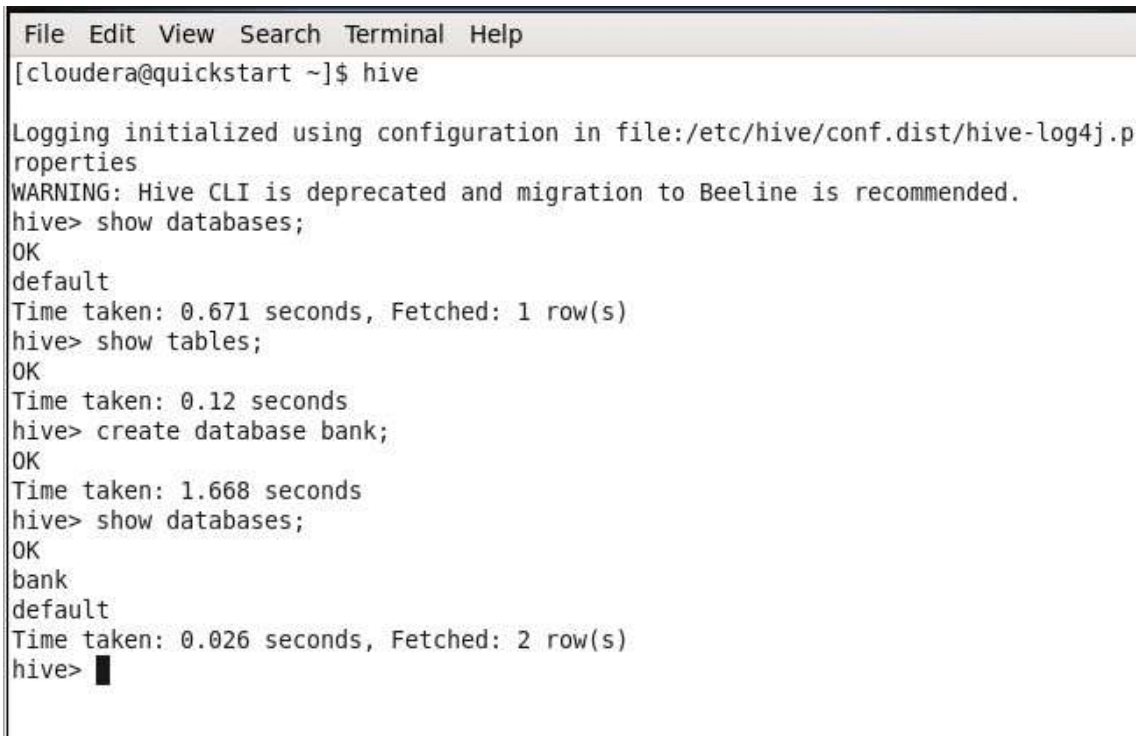
- Ease of Use: HiveQL is similar to SQL, making it accessible for analysts and data scientists familiar with traditional databases.

- Scalability: Hive is designed to scale with the underlying Hadoop architecture, allowing it to handle petabytes of data.
- Integration with Hadoop Ecosystem: Hive integrates seamlessly with other Hadoop components like HDFS, HBase, and Pig, providing a comprehensive big data processing framework.

Limitations of Hive

- Latency: Hive is optimized for batch processing and can have higher query latency compared to real-time databases. It's not suitable for low-latency or real-time querying.
- Complex Queries: Some complex queries can be challenging to optimize, and users may need to write intricate HiveQL to achieve specific results.
- Limited Support for Transactions: Hive has limited support for transactions compared to traditional relational databases, making it less suitable for certain types of OLTP applications.

Code & Output



```
File  Edit  View  Search  Terminal  Help
Time taken: 1.668 seconds
hive> show databases;
OK
bank
default
Time taken: 0.026 seconds, Fetched: 2 row(s)
hive> use bank;
OK
Time taken: 0.194 seconds
hive> create table emp(id INT,name STRING,sal DOUBLE)row format delimited fields
 terminated by ',' stored as textfile;
OK
Time taken: 0.527 seconds
hive> show tables;
OK
emp
Time taken: 0.043 seconds, Fetched: 1 row(s)
hive> describe emp;
OK
id                      int
name                    string
sal                     double
Time taken: 0.272 seconds, Fetched: 3 row(s)
hive>
```

```
[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
```

```
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>
```

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

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

```
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
```

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

```
hive> exit;
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
```

Experiment 7

Aim: Implement Bloom Filter using Python/R Programming

Theory:

A Bloom filter is a space-efficient probabilistic data structure used to test whether an element is part of a set. It is known for its compactness and speed, but it introduces a trade-off: false positives (where it incorrectly reports an element is in the set) are possible, but false negatives (where it incorrectly reports an element is not in the set) are not.

Key Characteristics:

- False positives possible: It may mistakenly indicate that an element is in the set when it isn't.
- No false negatives: If it says an element is not in the set, you can be certain it is not.
- Space efficiency: It uses less memory than other data structures (like hash sets) to store elements.

How Bloom Filters Work:

1. Bit Array: A Bloom filter consists of a bit array of size m, all initialized to 0.
2. Hash Functions: It uses k different independent hash functions. Each hash function maps an input element to one of the positions in the bit array, uniformly over the range [0, m-1].

Operations in Bloom Filters:

1. Insertion:
    - To insert an element into the filter, the element is passed through k hash functions, which each produce an index in the bit array. ○ Set the bits at the resulting indices to 1.
2. For example, if the element x is hashed to indices 2, 5, and 8, the bits at those positions in the bit array are set to 1.
3. Query (Membership Test):
    - To check if an element is in the set, it is hashed using the same k hash functions.
    - If all the corresponding bits for the hash values are set to 1, the element is possibly in the set (though false positives are possible).

- If any of the corresponding bits are 0, the element is definitely not in the set.

Code:

```
def main():
    # Read the length of the stream data m = int(input("Enter
    the length of the stream data (m): ")) data_stream = [0] *
    m

    # Read the number of inputs to train the array n =
    int(input("Enter the number of inputs to train the array (n): "))

    # Training the array
    for _ in range(n):
        input_value = int(input("Enter a number to train the array:
        ")) hash1 = (input_value % 5) % m hash2 = ((2 *
        input_value + 3) % 5) % m

        data_stream[hash1] = 1
        data_stream[hash2] = 1

    # Print the training array print("Training Array:", '
    '.join(map(str, data_stream)))

    # Read the number of times to test x =
    int(input("Enter the number of times to test (x): "))

    # Testing the array
    for _ in range(x):
        test_value = int(input("Enter a number to test:
        ")) hash1 = (test_value % 5) % m hash2 = ((2 *
        test_value + 3) % 5) % m

        if data_stream[hash1] == 0 and data_stream[hash2] == 0:
            print("Number is not in the stream.")
        elif data_stream[hash1] == 1 and data_stream[hash2] == 1:
            print("Number is in the stream.") else:
print("Value not present in the stream.") if
__name__ == "__main__":
```

main()

Output:

```
Enter the length of the stream data (m): 10
Enter the number of inputs to train the array (n): 3
Enter a number to train the array: 7
Enter a number to train the array: 12
Enter a number to train the array: 15
Training Array: 1 0 1 1 0 0 0 0 0 0
Enter the number of times to test (x): 2
Enter a number to test: 7
Number is in the stream.
Enter a number to test: 9
Number is not in the stream.

Process finished with exit code 0
```

Experiment 8

Aim: Implement FM algorithm using Python/R programming

Theory:

The FM (Flajolet-Martin) algorithm is a probabilistic algorithm used for estimating the number of distinct elements (also known as the cardinality) in a large data stream or set, where it is computationally impractical to store all elements or perform exact counting.

Key Concepts Behind the FM Algorithm:

1. Hashing: The core idea of the FM algorithm is to apply a hash function to each element of the stream. This converts each element into a smaller and uniformly distributed number within a fixed range.
2. Binary Representation and Trailing Zeros: After hashing, each hashed value is converted into its binary representation. The algorithm then counts the number of trailing zeros in the binary form of the hashed values. The number of trailing zeros in a hash value has a relationship with the probability of how often elements will appear based on the size of the data stream.
3. Maximum Trailing Zeros: The FM algorithm tracks the maximum number of trailing zeros observed among all the hashed values. This number is then used to estimate the number of distinct elements. The rationale is that longer runs of trailing zeros are increasingly rare, and the presence of such a run indicates a higher probability of a larger cardinality.
4. Estimation: The number of distinct elements (or the cardinality) is estimated as 2max trailing zeros2^{\text{max trailing zeros}}2max trailing zeros. If the maximum number of trailing zeros found is, for example, 3, the algorithm estimates that the number of distinct elements is approximately 23=82^3 = 823=8.

Steps of the FM Algorithm:

1. Hashing the Stream: Each element of the data stream is hashed using a hash function to obtain uniformly distributed values. For example, the hash function might be `hash(x) = (6 * x + 1) % 5`.
2. Binary Conversion: The hashed value is then converted to its binary form, ensuring it has a fixed number of bits (like 3-bit binary).

3. Counting Trailing Zeros: The algorithm counts how many trailing zeros are present in the binary representation of each hashed value. For example:
    - Hashed value: 0 → Binary: `000` → Trailing zeros: 3
    - Hashed value: 3 → Binary: `011` → Trailing zeros: 0
    - Hashed value: 4 → Binary: `100` → Trailing zeros: 2
4. Finding Maximum Trailing Zeros: The FM algorithm keeps track of the maximum number of trailing zeros observed. This value is crucial for estimating the number of distinct elements.
5. Cardinality Estimation: The final estimate of distinct elements is calculated as 2max trailing zeros2^{\text{max trailing zeros}}2max trailing zeros. If the largest number of trailing zeros observed is 3, the estimate is 23=82^3 = 823=8 distinct elements.

Code:

```
def hash(x):
    # Hash function: (6 * x + 1) % 5
    return (6 * x + 1) % 5

def to_three_bit_binary(num):
    # Convert to binary and ensure it has 3 bits (padding with leading
    zeros) binary = bin(num)[2:] # Remove '0b' prefix return binary.zfill(3)
    # Pad with leading zeros to make it 3-bit

def count_trailing_zeros(arr):
    result = [] for binary in arr:
    count = 0 encountered_one
    = False
        # Iterate from the last bit (right to left) and count trailing zeros
        for j in range(len(binary) - 1, -1, -1):
            if binary[j] == '0' and not encountered_one:
                count += 1 elif
            binary[j] == '1':
                encountered_one = True
        result.append(count)
    return result

def main():
```

```
    # Get the size of the input array size =
    int(input("Enter the size of the array: "))
    input_array = []
    # Take input elements for the array
    print("Enter elements of the array:")
    for i in range(size):
        element = int(input(f"Element {i + 1}: "))
        input_array.append(element)

    # Apply the hash function to all elements
    hashed_array = [hash(x) for x in input_array]
    print(f"Hashed array: {hashed_array}")

    # Convert each hashed value to 3-bit binary binary_array =
    [to_three_bit_binary(x) for x in hashed_array] print(f"Binary
    array: {binary_array}")

    # Count trailing zeros in each binary representation
    trailing_zeros_array = count_trailing_zeros(binary_array)
    print(f"Trailing zeros array: {trailing_zeros_array}")

    # Find the maximum number of trailing zeros
    max_trailing_zeros = max(trailing_zeros_array)

    # Calculate the number of distinct elements as 2 ^ (max trailing
    zeros) power_of_two = 2 ** max_trailing_zeros print(f"Maximum
    trailing zeros: {max_trailing_zeros}") print(f"The number of distinct
    elements: {power_of_two}")

if __name__ ==

"__main__": main()
```

Output:

```
Enter the size of the array: 5
Enter elements of the array:
Element 1: 4
Element 2: 7
Element 3: 12
Element 4: 3
Element 5: 9
```

Hashed array: [0, 3, 3, 4, 0]
Binary array: ['000', '011', '011', '100', '000']
Trailing zeros array: [3, 0, 0, 2, 3]
Maximum trailing zeros: 3
The number of distinct elements: 8

Process finished with exit code 0

Experiment 9

Aim: Data Visualization using R

Theory:

Data visualization is a crucial aspect of understanding and communicating insights from a dataset. In R, there are various ways to represent data graphically, including histograms, scatter plots, boxplots, heatmaps, and bar plots. These graphical methods help us interpret complex data structures and trends at a glance. The provided code examples use these techniques to explore data from the "Placement_Dataset.csv."

1. Histogram of MBA Percentage

Histogram: A histogram is used to visualize the distribution of a single quantitative variable. It represents the frequency of data points falling within defined intervals (bins). In this example, the MBA percentage (`mba_p`) of the candidates is plotted on the x-axis, while the frequency (count) of candidates falling within each percentage range is on the y-axis.

- Purpose: To understand the distribution of MBA scores among the candidates.
- Main Insight: Histograms can help identify the central tendency, spread, and skewness of the data. In this case, we can determine if most candidates score high or low in MBA exams.

2. Scatter Plot of Test Scores vs Salary

Scatter Plot: A scatter plot is used to observe the relationship between two continuous variables. Here, the test scores (`etest_p`) are plotted on the x-axis, while salaries (`salary`) are on the y-axis. This helps identify correlations or trends.

- Purpose: To examine if there is any relationship between the candidate's test scores and their offered salary.
- Main Insight: Scatter plots reveal whether an increase in test scores leads to an increase in salary, and the strength of this relationship.

3. Box Plot of HSC Percentage

Boxplot: A box plot provides a summary of the distribution of a dataset. It displays the median, quartiles, and potential outliers of the data. In this example, the HSC percentage (`hsc_p`) is represented.

- Purpose: To show the spread and variability of HSC scores.
- Main Insight: Boxplots are particularly useful for identifying the interquartile range (middle 50% of data) and any outliers in the data. The `notch = TRUE` option highlights the confidence interval around the median, helping compare medians between groups.

4. Heatmap of Selected Variables

Heatmap: A heatmap visualizes data through color gradients, allowing for easy identification of patterns and relationships between variables. In this example, a subset of the data (columns 3, 5, 8, and 11) is selected and visualized.

- Purpose: To observe patterns and correlations between variables.
- Main Insight: Heatmaps help in understanding how different variables are related across multiple candidates by using color to represent the magnitude of the values.

5. Grouped Bar Plot of Gender vs Employment Status

Bar Plot: A bar plot is used to display the frequency or proportion of categorical data. In this case, the plot shows the relationship between `gender` and `status` (employment status). The bars are grouped by gender.

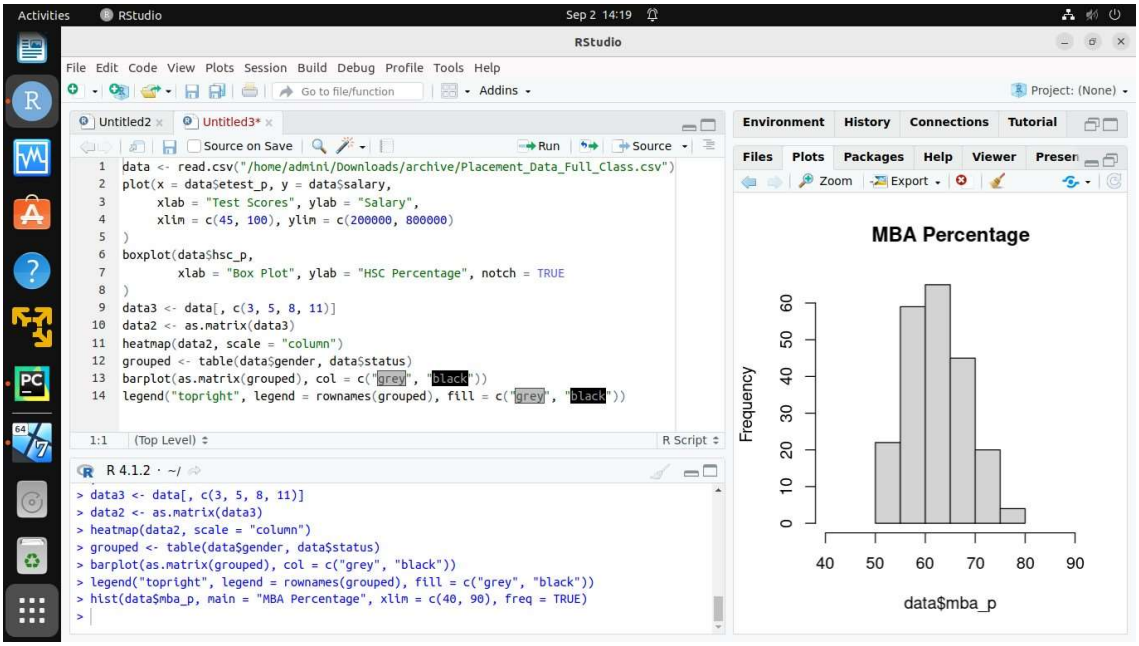
- Purpose: To compare the employment status between different genders.
- Main Insight: Grouped bar plots help in visualizing categorical comparisons, such as whether males or females have a higher employment rate.

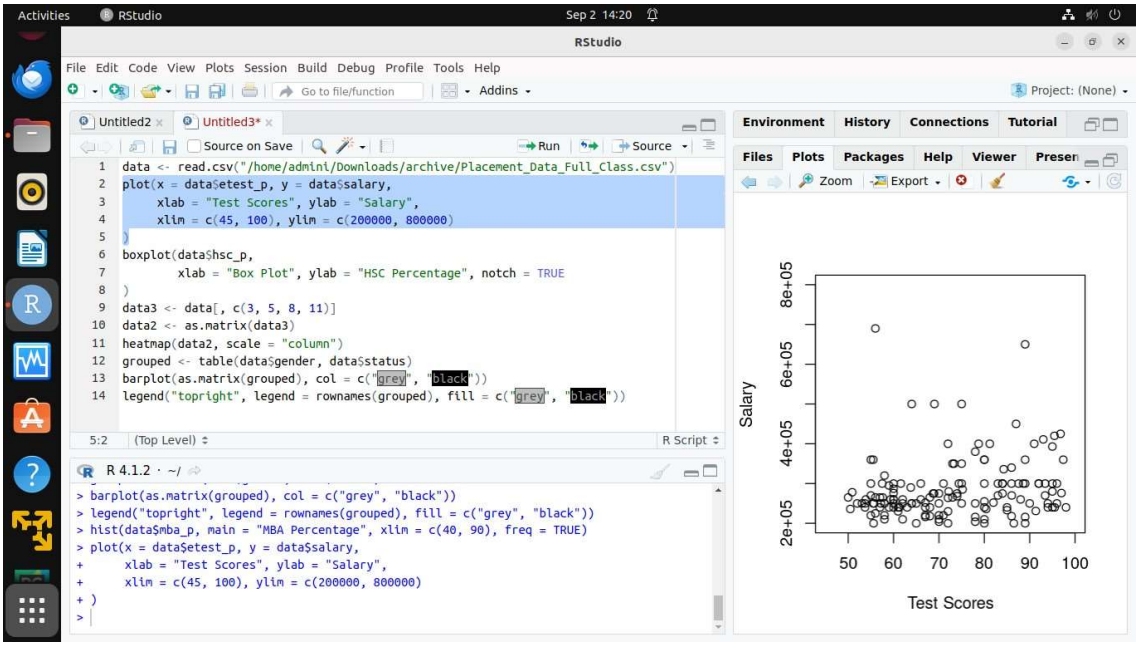
Code:

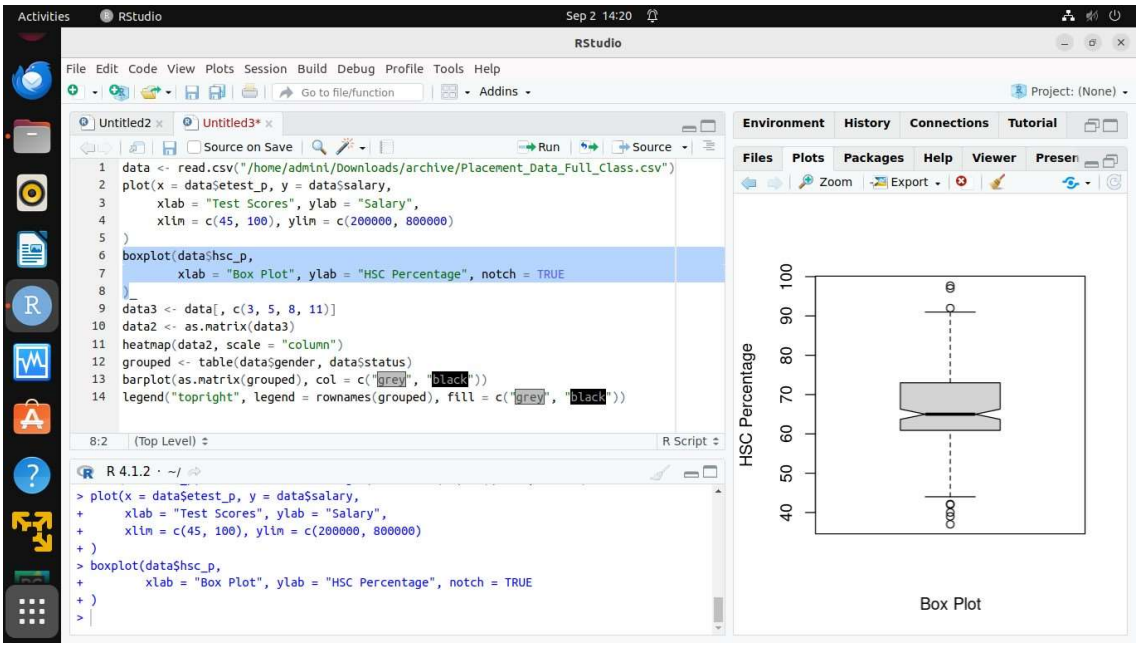
```
data <- read.csv("Placement_Dataset.csv") hist(data$mba_p, main = "MBA
Percentage", xlim = c(40, 90), freq = TRUE) data <-
read.csv("Placement_Dataset.csv") plot(x = data$etest_p, y = data$salary,
xlab = "Test Scores", ylab = "Salary", xlim = c(45, 100), ylim = c(200000,
800000)
)
```

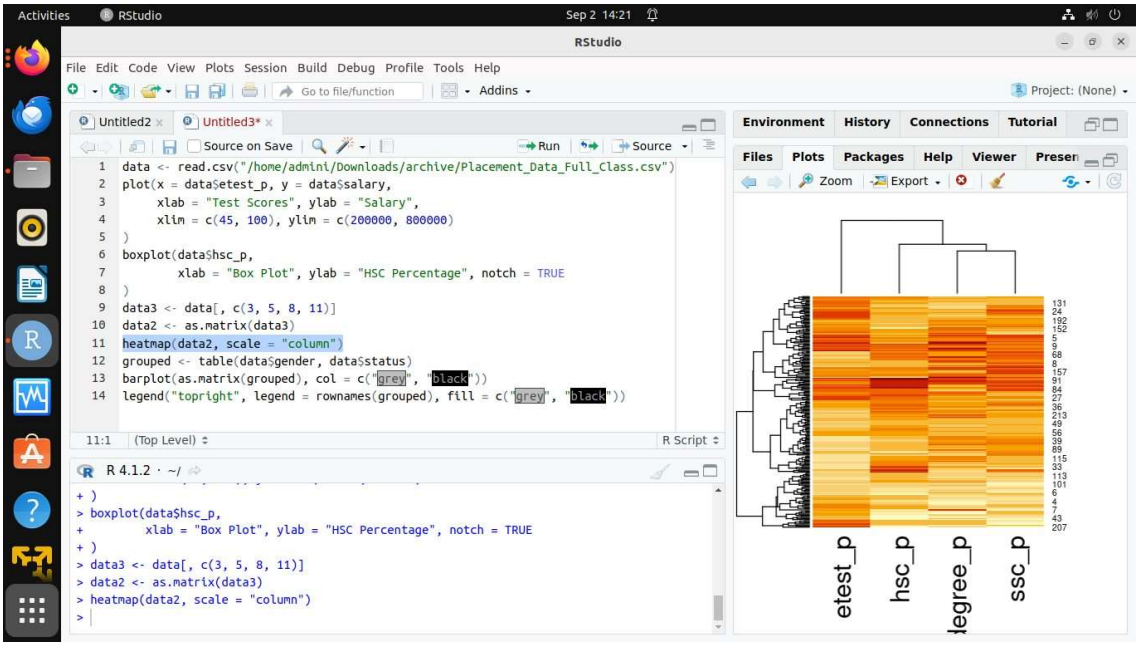
```
data <- read.csv("Placement_Dataset.csv")
boxplot(data$hsc_p, xlab = "Box Plot", ylab = "HSC
Percentage", notch = TRUE
)
data <- read.csv("Placement_Dataset.csv")
data3 <- data[, c(3, 5, 8, 11)]a
data2 <- as.matrix(data3) heatmap(data2, scale =
"column") data <-
read.csv("Placement_Dataset.csv") grouped <-
table(data$gender, data$status)
barplot(as.matrix(grouped), col = c("grey",
"black"))
legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
```

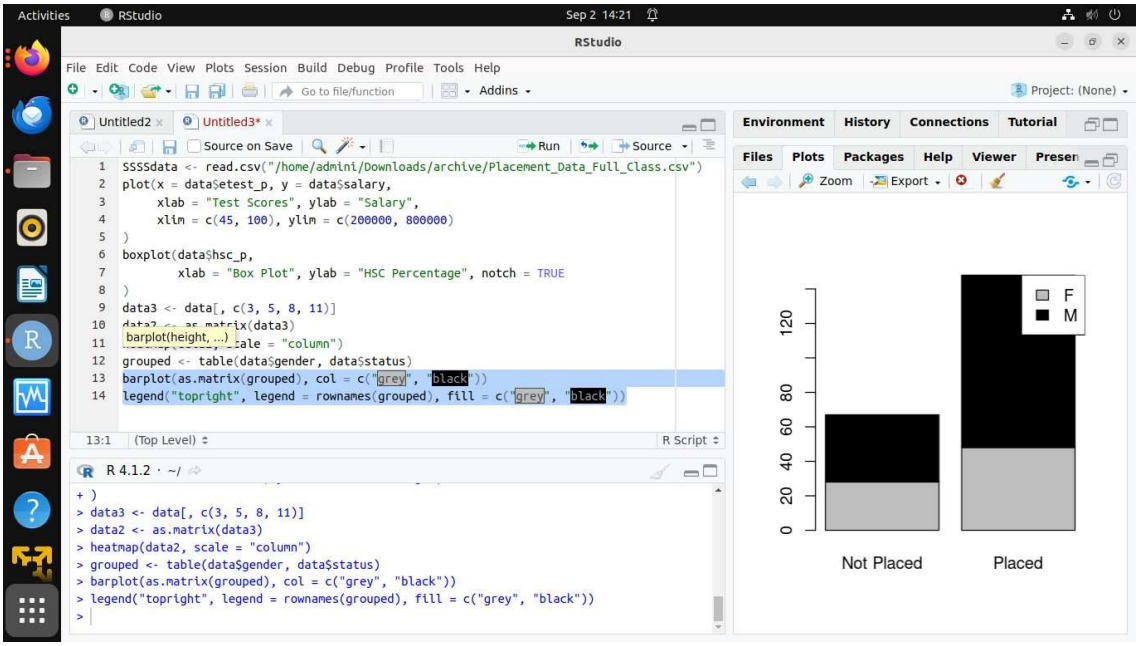
Output:


Activities
RStudio
Sep 2 14:19
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled3*
Source on Save
Run
Source
1 data <- read.csv("/home/admini/Downloads/archive/Placement_Data_Full_Class.csv")
2 plot(x = data$etest_p, y = data$salary,
3 xlab = "Test Scores", ylab = "Salary",
4 xlim = c(45, 100), ylim = c(200000, 800000)
5)
6 boxplot(data$hsc_p,
7 xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
8)
9 data3 <- data[, c(3, 5, 8, 11)]
10 data2 <- as.matrix(data3)
11 heatmap(data2, scale = "column")
12 grouped <- table(data$gender, data$status)
13 barplot(as.matrix(grouped), col = c("grey", "black"))
14 legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
1:1 (Top Level) R Script
R 4.1.2 · ~/
> data3 <- data[, c(3, 5, 8, 11)]
> data2 <- as.matrix(data3)
> heatmap(data2, scale = "column")
> grouped <- table(data$gender, data$status)
> barplot(as.matrix(grouped), col = c("grey", "black"))
> legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
> hist(data$mba_p, main = "MBA Percentage", xlim = c(40, 90), freq = TRUE)
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presen
Zoom Export
MBA Percentage
Frequency
0 10 20 30 40 50 60
40 50 60 70 80 90
data$mba_p


Activities
RStudio
Sep 2 14:20
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled3*
Source on Save
Run
Source
1 data <- read.csv("/home/admini/Downloads/archive/Placement_Data_Full_Class.csv")
2 plot(x = data$etest_p, y = data$salary,
3 xlab = "Test Scores", ylab = "Salary",
4 xlim = c(45, 100), ylim = c(200000, 800000)
5)
6 boxplot(data$hsc_p,
7 xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
8)
9 data3 <- data[, c(3, 5, 8, 11)]
10 data2 <- as.matrix(data3)
11 heatmap(data2, scale = "column")
12 grouped <- table(data$gender, data$status)
13 barplot(as.matrix(grouped), col = c("grey", "black"))
14 legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
5:2 (Top Level) R Script
R 4.1.2 · ~/
> barplot(as.matrix(grouped), col = c("grey", "black"))
> legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
> hist(data$mba_p, main = "MBA Percentage", xlim = c(40, 90), freq = TRUE)
> plot(x = data$etest_p, y = data$salary,
+ xlab = "Test Scores", ylab = "Salary",
+ xlim = c(45, 100), ylim = c(200000, 800000)
+)
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presen
Zoom Export
Salary
2e+05 4e+05 6e+05 8e+05
50 60 70 80 90 100
Test Scores


Activities
RStudio
Sep 2 14:20
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled3*
Source on Save
Run
Source
1 data <- read.csv("/home/admini/Downloads/archive/Placement_Data_Full_Class.csv")
2 plot(x = data$etest_p, y = data$salary,
3 xlab = "Test Scores", ylab = "Salary",
4 xlim = c(45, 100), ylim = c(200000, 800000)
5)
6 boxplot(data$hsc_p,
7 xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
8)
9 data3 <- data[, c(3, 5, 8, 11)]
10 data2 <- as.matrix(data3)
11 heatmap(data2, scale = "column")
12 grouped <- table(data$gender, data$status)
13 barplot(as.matrix(grouped), col = c("grey", "black"))
14 legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
8:2 (Top Level)
R Script
R 4.1.2 · ~/
> plot(x = data$etest_p, y = data$salary,
+ xlab = "Test Scores", ylab = "Salary",
+ xlim = c(45, 100), ylim = c(200000, 800000)
+)
> boxplot(data$hsc_p,
+ xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
+)
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Zoom
Export
HSC Percentage
40
50
60
70
80
90
100
Box Plot


Activities
RStudio
Sep 2 14:21
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled3*
Source on Save
Run
Source
1 data <- read.csv("/home/admini/Downloads/archive/Placement_Data_Full_Class.csv")
2 plot(x = data$etest_p, y = data$salary,
3 xlab = "Test Scores", ylab = "Salary",
4 xlim = c(45, 100), ylim = c(200000, 800000)
5)
6 boxplot(data$hsc_p,
7 xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
8)
9 data3 <- data[, c(3, 5, 8, 11)]
10 data2 <- as.matrix(data3)
11 heatmap(data2, scale = "column")
12 grouped <- table(data$gender, data$status)
13 barplot(as.matrix(grouped), col = c("grey", "black"))
14 legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
11:1 (Top Level)
R Script
R 4.1.2 · ~/
+)
> boxplot(data$hsc_p,
+ xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
+)
> data3 <- data[, c(3, 5, 8, 11)]
> data2 <- as.matrix(data3)
> heatmap(data2, scale = "column")
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Zoom
Export
etest_p
hsc_p
ssc_p


Activities
RStudio
Sep 2 14:21
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled2
Untitled3*
Source on Save
Run
Source
1 SSSSdata <- read.csv("/home/admin1/Downloads/archive/Placement_Data_Full_Class.csv")
2 plot(x = data$etest_p, y = data$salary,
3 xlab = "Test Scores", ylab = "Salary",
4 xlim = c(45, 100), ylim = c(200000, 800000)
5)
6 boxplot(data$hsc_p,
7 xlab = "Box Plot", ylab = "HSC Percentage", notch = TRUE
8)
9 data3 <- data[, c(3, 5, 8, 11)]
10 data2 <- as.matrix(data3)
11 barplot(height, ...)
12 grouped <- table(data$gender, data$status)
13 barplot(as.matrix(grouped), col = c("grey", "black"))
14 legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
13:1 (Top Level)
R Script
R 4.1.2 · ~/
+)
> data3 <- data[, c(3, 5, 8, 11)]
> data2 <- as.matrix(data3)
> heatmap(data2, scale = "column")
> grouped <- table(data$gender, data$status)
> barplot(as.matrix(grouped), col = c("grey", "black"))
> legend("topright", legend = rownames(grouped), fill = c("grey", "black"))
>
Environment History Connections Tutorial
Files Plots Packages Help Viewer Preser
Zoom
Export
0 20 40 60 80 120
F
M
Not Placed
Placed

Mini Project :-Youtube Data Analysis

CHAPTER 1: INTRODUCTION

In this project, I focused on big data analytics using a comprehensive approach, integrating multiple technologies to handle and analyze large datasets. The objective was to analyze FBI crime data, covering multiple steps from data ingestion to final analysis.

First, the crime data was loaded into a MySQL database, which served as the initial storage. From there, the data was transferred to HBase using Sqoop to leverage HBase's NoSQL capabilities for large-scale data handling. Once stored in HBase, Hive was utilized for data querying, enabling a familiar SQL-like interface to interact with the data while processing it in a distributed manner using MapReduce.

To conduct the final analysis, a MapReduce program was created to process the data and produce insights, such as the number of crimes committed based on their type and whether arrests were made. The results were then saved to HDFS.

For visualization, the MapReduce output was processed and transformed into a format that could be analyzed and visualized using Python. Using libraries like Matplotlib, I was able to generate plots that illustrated the crime distribution and arrest frequencies, providing valuable insights into the dataset.

This project showcases a full pipeline of data handling, from ingestion and storage to processing and visualization, using big data tools to handle complex, large-scale datasets.

CHAPTER 2: DATA DESCRIPTION AND ANALYSIS

The dataset utilized in this project encompasses three primary CSV files: albums, tracks, and artists. Each dataset contributes unique attributes crucial for understanding the Spotify ecosystem comprehensively.

1. Albums Data:

This dataset contains detailed information about albums available on Spotify.

Key attributes include:

- rank: Ranking of the YouTuber based on certain criteria (e.g., subscribers)
- youtuber: Name of the YouTube channel
- Video views: Total number of video views on the channel (stored as a string)
- Video count: Total number of videos uploaded by the channel
- Subscribers: Number of subscribers (stored as a string, requires cleaning)

- Category: Category or genre the YouTube channel falls under (e.g., Music, Gaming)
- Started: Year the YouTube channel was started

Analysis:

Analysis can include:

- **Average Subscribers per Category**
  This query calculates the average number of subscribers for YouTube channels in each category after cleaning the subscribers column (removing commas and converting to integer format).
- **Maximum Video Views per Category**
  This query retrieves the highest number of video views across all YouTube channels in each category, giving insights into the top-performing videos in different genres.
- **Total Video Count per Category**
  This query calculates the total number of videos uploaded in each category. It sums up the video counts for each category to gauge the overall content volume in various genres.
- **Top 5 YouTubers by Subscribers**
  This query lists the top 5 YouTube channels ranked by their number of subscribers.

These analyses help in understanding the performance metrics across various YouTube categories, identifying trends in viewership, content creation, and subscriber growth.

CHAPTER 3: DESIGN OF DATA PIPELINE

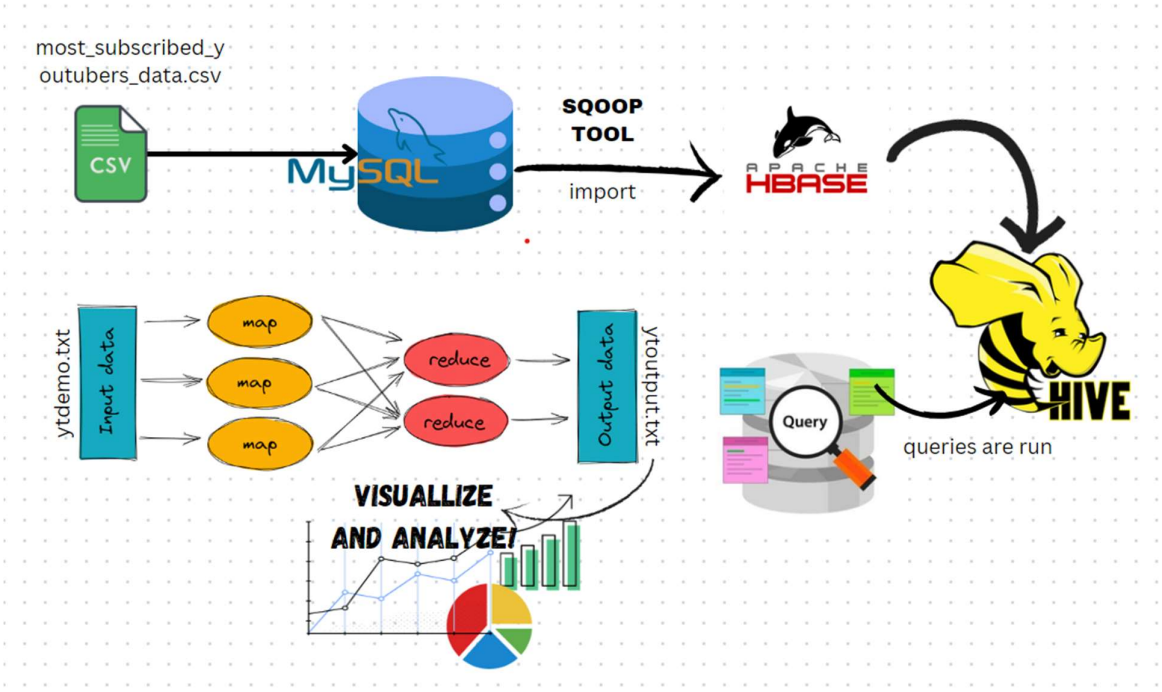
High-Level Architecture of Data Pipeline

1. **Data Ingestion**
   - **YouTube Data API** or **CSV Import**
   - Stored in **MySQL** (Relational Database)
2. **Data Storage**
   - **Sqoop** → Import to **HBase** / **HDFS** for distributed storage and processing
3. **Data Processing**
   - **Hive** external tables for querying the data in **HBase**
   - **MapReduce** jobs for large-scale aggregations and analysis
4. **Data Output**

- Results stored in **HDFS**
- Exported as CSV for further analysis

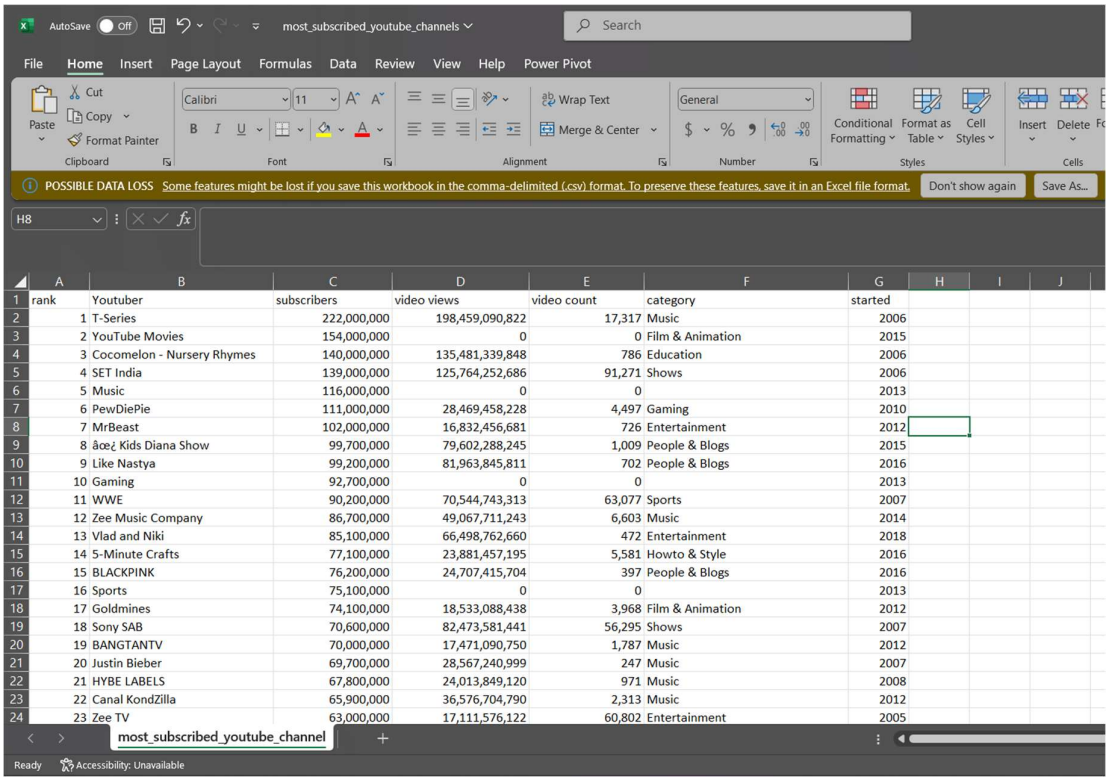
5. **Data Visualization**
    - **Python (Pandas, Matplotlib)** to create visual reports based on the analysis

This data pipeline ensures smooth handling of the YouTube dataset, from ingestion to final analysis and visualization.



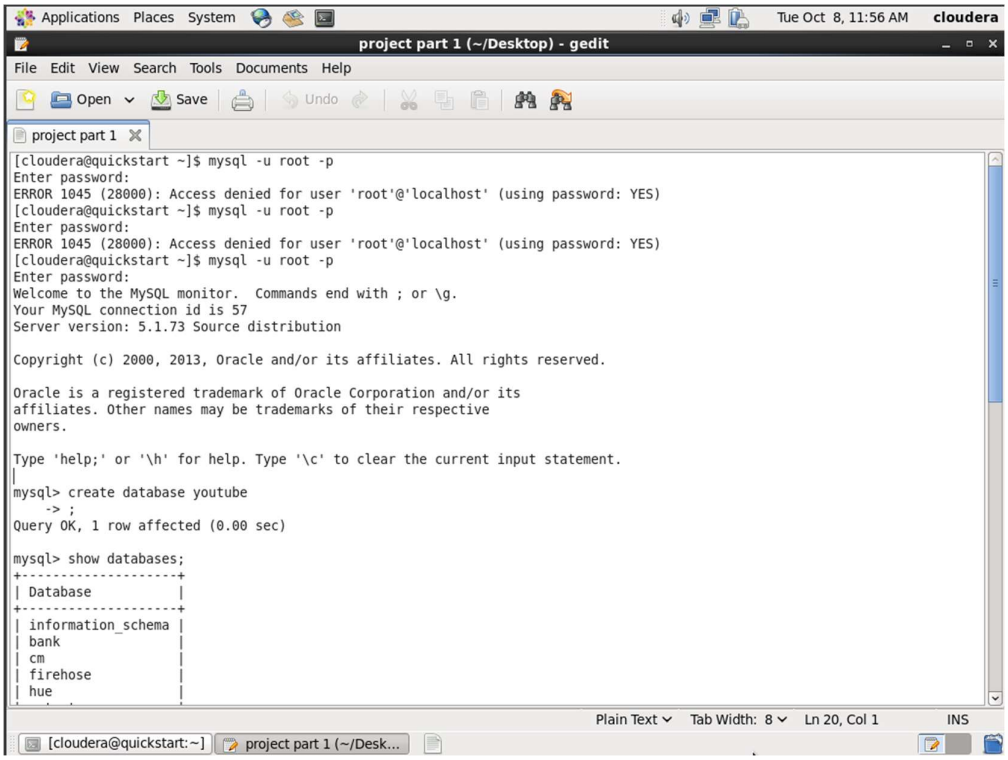
CHAPTER 4: RESULT ANALYSIS

Csv file–

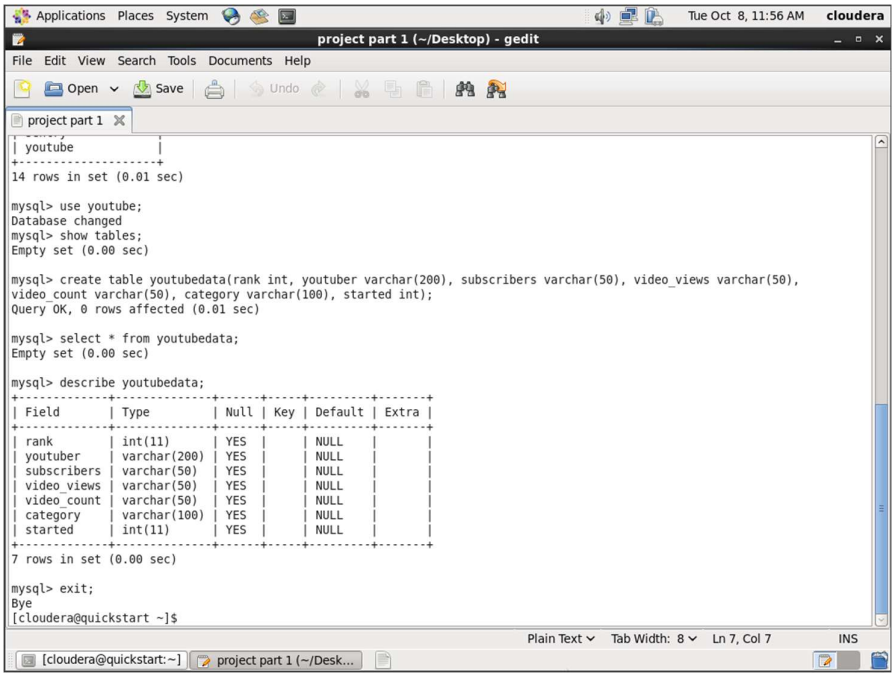


rank	Youtuber	subscribers	video views	video count	category	started
1	T-Series	222,000,000	198,459,090,822	17,317	Music	2006
2	YouTube Movies	154,000,000	0	0	Film & Animation	2015
3	Cocomelon - Nursery Rhymes	140,000,000	135,481,339,848	786	Education	2006
4	SET India	139,000,000	125,764,252,686	91,271	Shows	2006
5	Music	116,000,000	0	0		2013
6	PewDiePie	111,000,000	28,469,458,228	4,497	Gaming	2010
7	MrBeast	102,000,000	16,832,456,681	726	Entertainment	2012
8	âœ¿ Kids Diana Show	99,700,000	79,602,288,245	1,009	People & Blogs	2015
9	Like Nastya	99,200,000	81,963,845,811	702	People & Blogs	2016
10	Gaming	92,700,000	0	0		2013
11	WWE	90,200,000	70,544,743,313	63,077	Sports	2007
12	Zee Music Company	86,700,000	49,067,711,243	6,603	Music	2014
13	Vlad and Niki	85,100,000	66,498,762,660	472	Entertainment	2018
14	5-Minute Crafts	77,100,000	23,881,457,195	5,581	Howto & Style	2016
15	BLACKPINK	76,200,000	24,707,415,704	397	People & Blogs	2016
16	Sports	75,100,000	0	0		2013
17	Goldmines	74,100,000	18,533,088,438	3,968	Film & Animation	2012
18	Sony SAB	70,600,000	82,473,581,441	56,295	Shows	2007
19	BANGTANTV	70,000,000	17,471,090,750	1,787	Music	2012
20	Justin Bieber	69,700,000	28,567,240,999	247	Music	2007
21	HYBE LABELS	67,800,000	24,013,849,120	971	Music	2008
22	Canal KondZilla	65,900,000	36,576,704,790	2,313	Music	2012
23	Zee TV	63,000,000	17,111,576,122	60,802	Entertainment	2005

Preparing the database and



creating a table of youtube data



Loading the data into the table:

LOAD DATA INFILE "/home/cloudera/Desktop/windows-shared/most_subscribed_youtube_channels.csv"

INTO TABLE youtube.youtubedata

COLUMNS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

ESCAPED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

Loading the table HBASE:

Plan : We can use Sqoop to load the data from RDBMS to HBASE!

Table Design in HBASE:

Login to HBASE

hbase shell

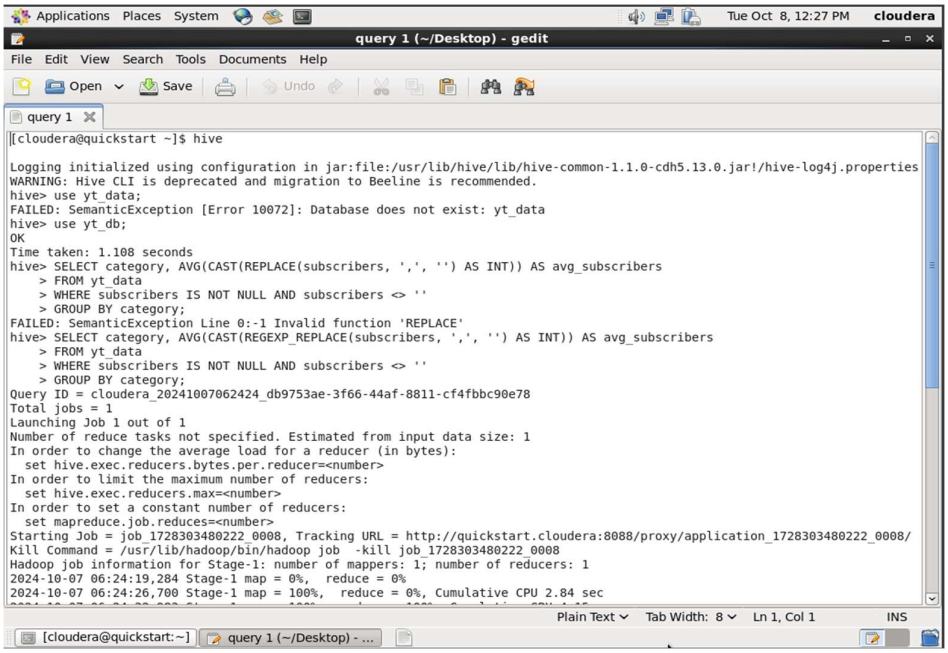
create 'yt','cols'

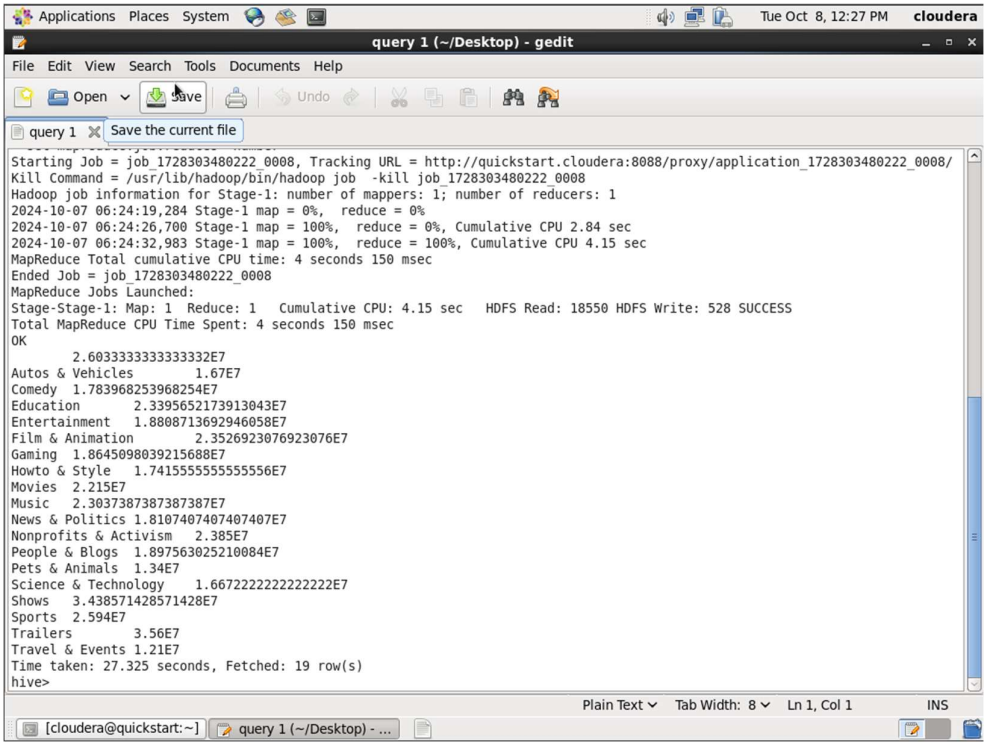
sqoop import --connect jdbc:mysql://localhost/youtube --table youtubedata --hbase-table yt --column-family cols --hbase-row-key rank --m 1 --username root -P

Then we will use hive to access the hbase table and create hive equeries which will create map-reduce

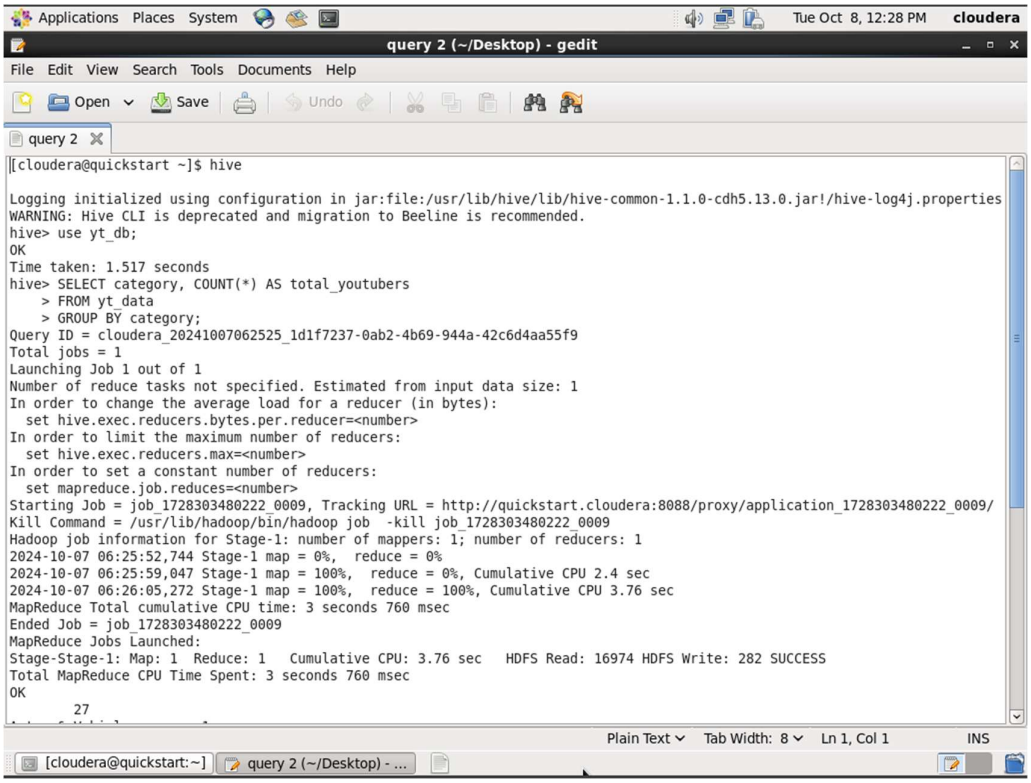
on top of the database.

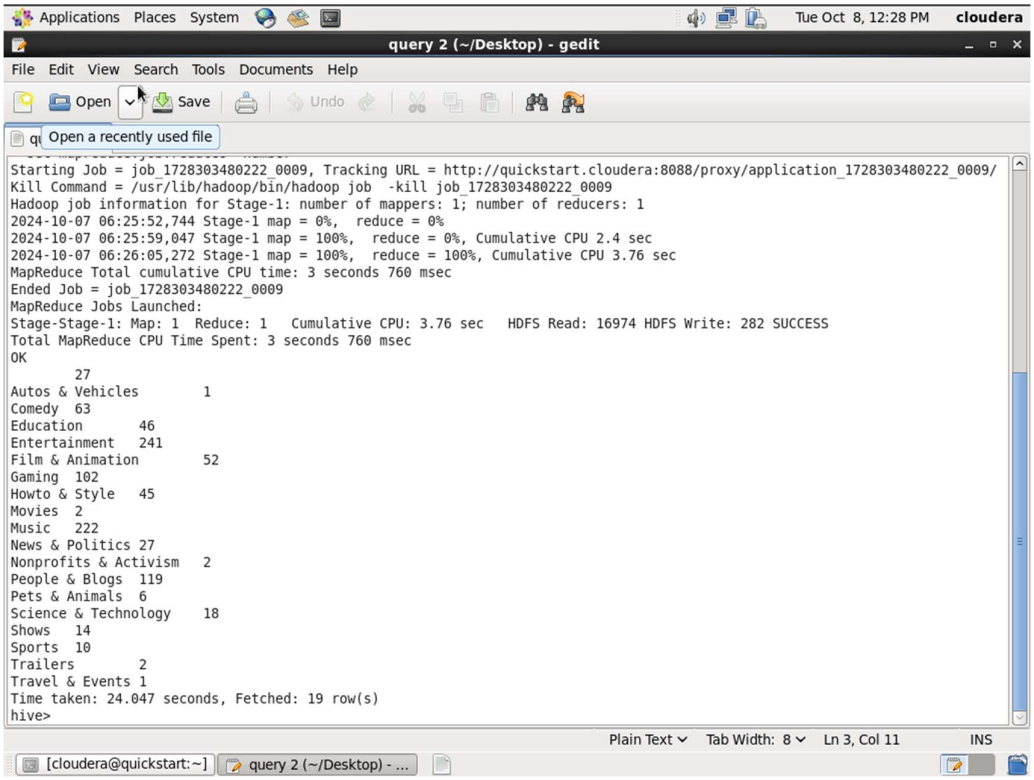
Query 1:



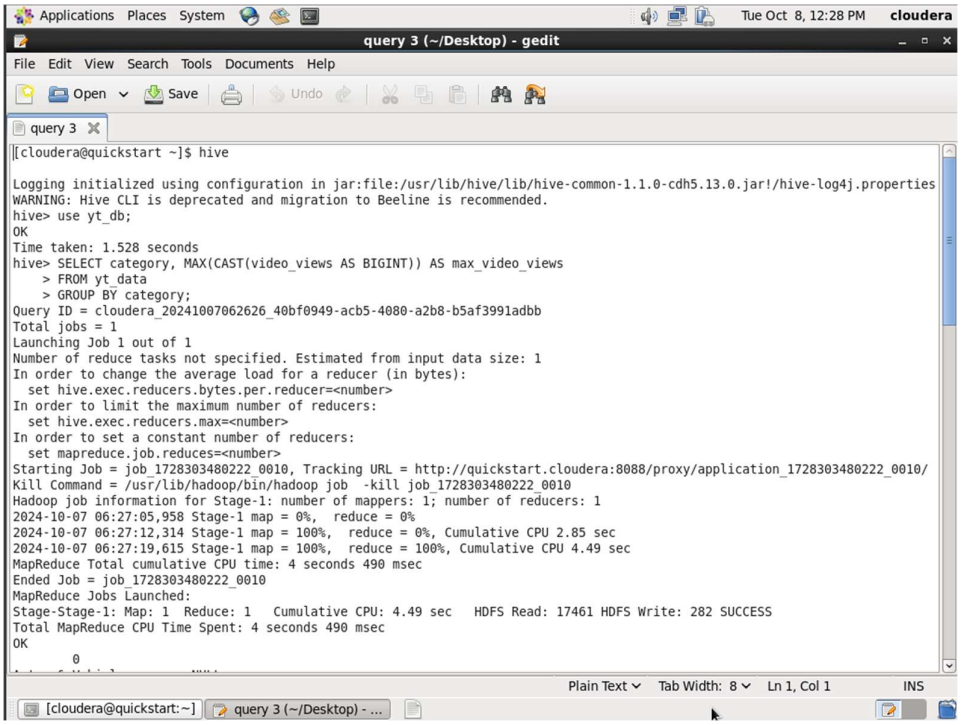


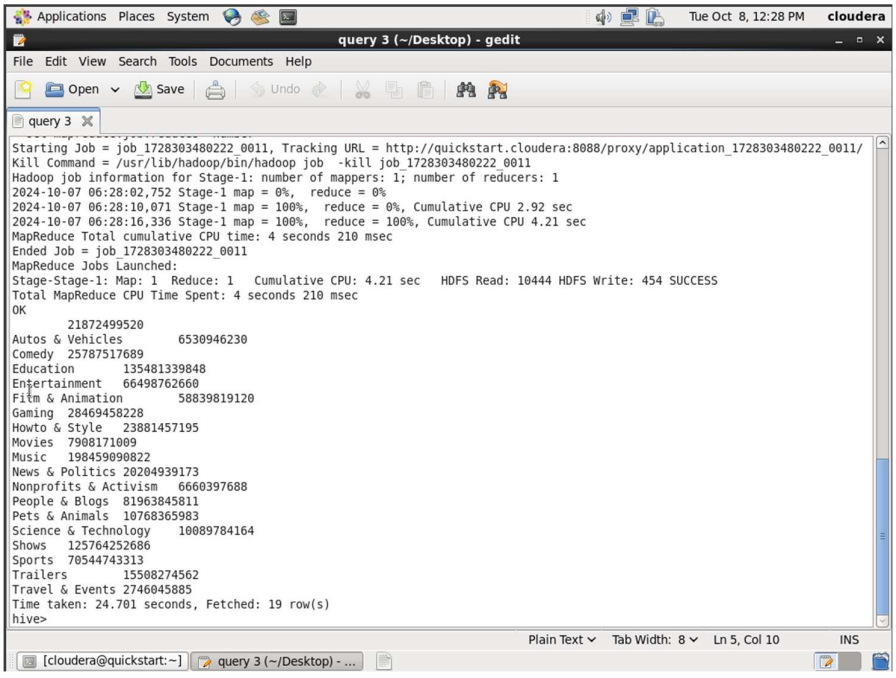
Query 2:


Applications Places System
Tue Oct 8, 12:28 PM cloudera
query 2 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
query 2
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> use yt_db;
OK
Time taken: 1.517 seconds
hive> SELECT category, COUNT(*) AS total_youtubers
> FROM yt_data
> GROUP BY category;
Query ID = cloudera_20241007062525_1d1f7237-0ab2-4b69-944a-42c6d4aa55f9
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
set mapreduce.job.reduces=<number>
Starting Job = job_1728303480222_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1728303480222_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1728303480222_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-07 06:25:52,744 Stage-1 map = 0%, reduce = 0%
2024-10-07 06:25:59,047 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.4 sec
2024-10-07 06:26:05,272 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.76 sec
MapReduce Total cumulative CPU time: 3 seconds 760 msec
Ended Job = job_1728303480222_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.76 sec HDFS Read: 16974 HDFS Write: 282 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 760 msec
OK
27
Plain Text Tab Width: 8 Ln 1, Col 1 INS
[cloudera@quickstart:~]
query 2 (~/Desktop) - ...

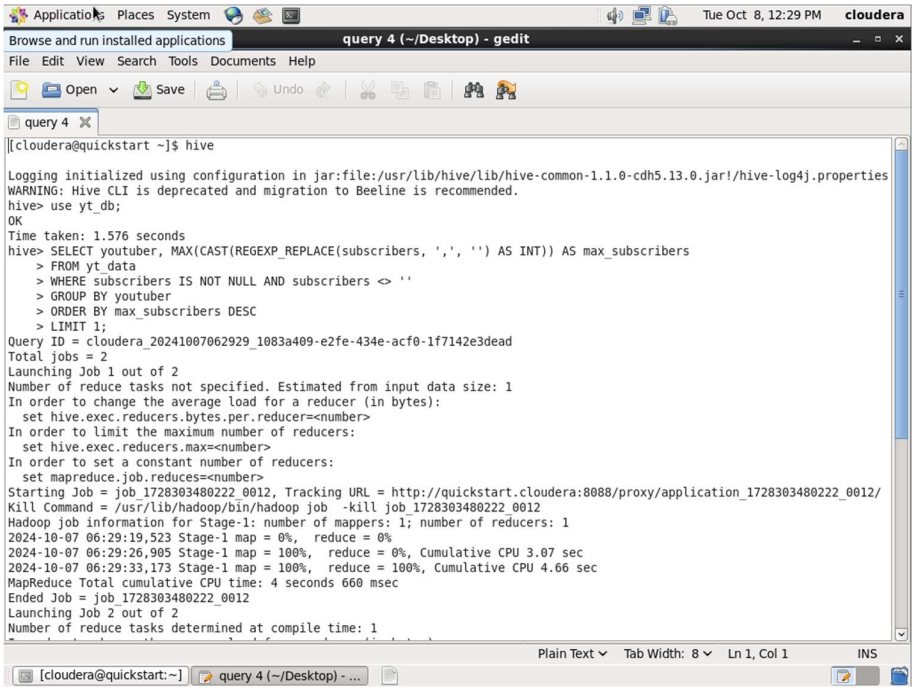

Applications Places System
Tue Oct 8, 12:28 PM cloudera
query 2 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
Open a recently used file
Starting Job = job_1728303480222_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1728303480222_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1728303480222_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-07 06:25:52,744 Stage-1 map = 0%, reduce = 0%
2024-10-07 06:25:59,047 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.4 sec
2024-10-07 06:26:05,272 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.76 sec
MapReduce Total cumulative CPU time: 3 seconds 760 msec
Ended Job = job_1728303480222_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.76 sec HDFS Read: 16974 HDFS Write: 282 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 760 msec
OK
27
Autos & Vehicles 1
Comedy 63
Education 46
Entertainment 241
Film & Animation 52
Gaming 102
Howto & Style 45
Movies 2
Music 222
News & Politics 27
Nonprofits & Activism 2
People & Blogs 119
Pets & Animals 6
Science & Technology 18
Shows 14
Sports 10
Trailers 2
Travel & Events 1
Time taken: 24.047 seconds, Fetched: 19 row(s)
hive>
Plain Text Tab Width: 8 Ln 3, Col 11 INS
[cloudera@quickstart:~]
query 2 (~/Desktop) - ...

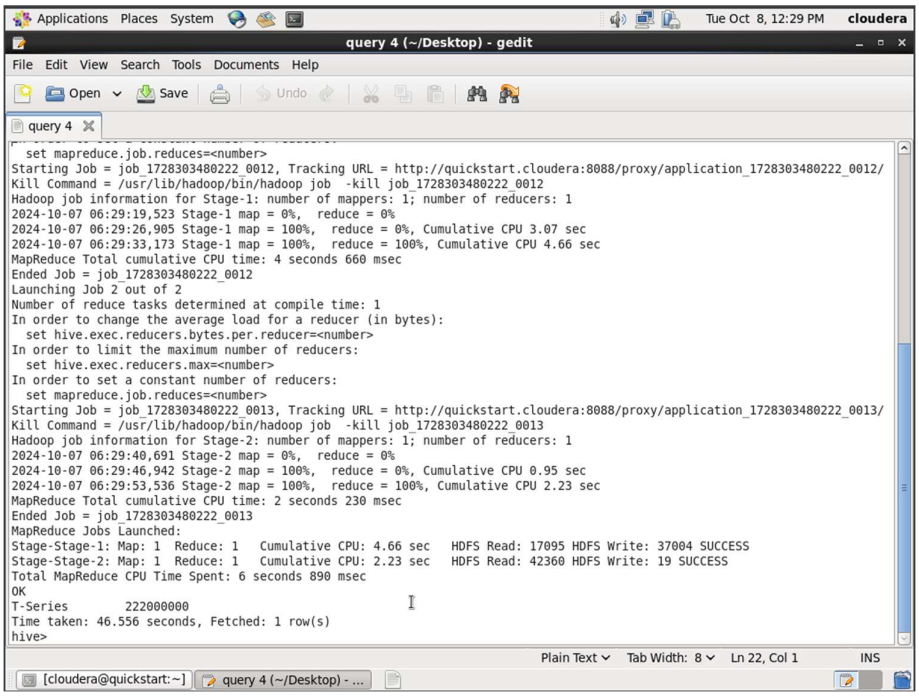
Query 3:


Applications Places System
Tue Oct 8, 12:28 PM cloudera
query 3 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
query 3
[cloudera@quickstart ~]$ hive
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> use yt_db;
OK
Time taken: 1.528 seconds
hive> SELECT category, MAX(CAST(video_views AS BIGINT)) AS max_video_views
> FROM yt_data
> GROUP BY category;
Query ID = cloudera_20241007062626_40bf0949-acb5-4080-a2b8-b5af3991adbb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
set mapreduce.job.reduces=<number>
Starting Job = job_1728303480222_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1728303480222_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1728303480222_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-07 06:27:05,958 Stage-1 map = 0%, reduce = 0%
2024-10-07 06:27:12,314 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.85 sec
2024-10-07 06:27:19,615 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.49 sec
MapReduce Total cumulative CPU time: 4 seconds 490 msec
Ended Job = job_1728303480222_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.49 sec HDFS Read: 17461 HDFS Write: 282 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 490 msec
OK
0
Plain Text Tab Width: 8 Ln 1, Col 1 INS
[cloudera@quickstart:~] query 3 (~/Desktop) - ...

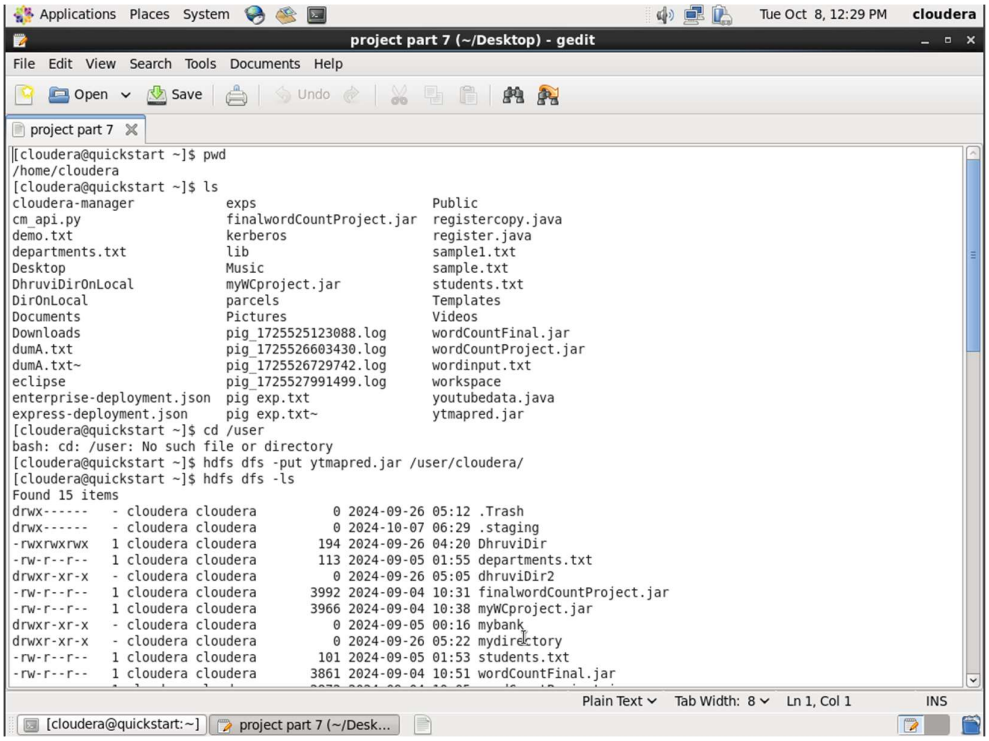

Applications Places System
Tue Oct 8, 12:28 PM cloudera
query 3 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
query 3
Starting Job = job_1728303480222_0011, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1728303480222_0011/
Kill Command = /usr/lib/hadoop/bin/hadoop job -kill job_1728303480222_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-10-07 06:28:02,752 Stage-1 map = 0%, reduce = 0%
2024-10-07 06:28:10,071 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.92 sec
2024-10-07 06:28:16,336 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.21 sec
MapReduce Total cumulative CPU time: 4 seconds 210 msec
Ended Job = job_1728303480222_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.21 sec HDFS Read: 10444 HDFS Write: 454 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 210 msec
OK
21872499520
Autos & Vehicles 6530946230
Comedy 25787517689
Education 135481339848
Entertainment 66498762660
Film & Animation 58839819120
Gaming 28469458228
Howto & Style 23881457195
Movies 7908171009
Music 198459090822
News & Politics 20204939173
Nonprofits & Activism 6660397688
People & Blogs 81963845811
Pets & Animals 10768365983
Science & Technology 10089784164
Shows 125764252686
Sports 70544743313
Trailers 15508274562
Travel & Events 2746045885
Time taken: 24.701 seconds, Fetched: 19 row(s)
hive>
Plain Text Tab Width: 8 Ln 5, Col 10 INS
[cloudera@quickstart:~] query 3 (~/Desktop) - ...

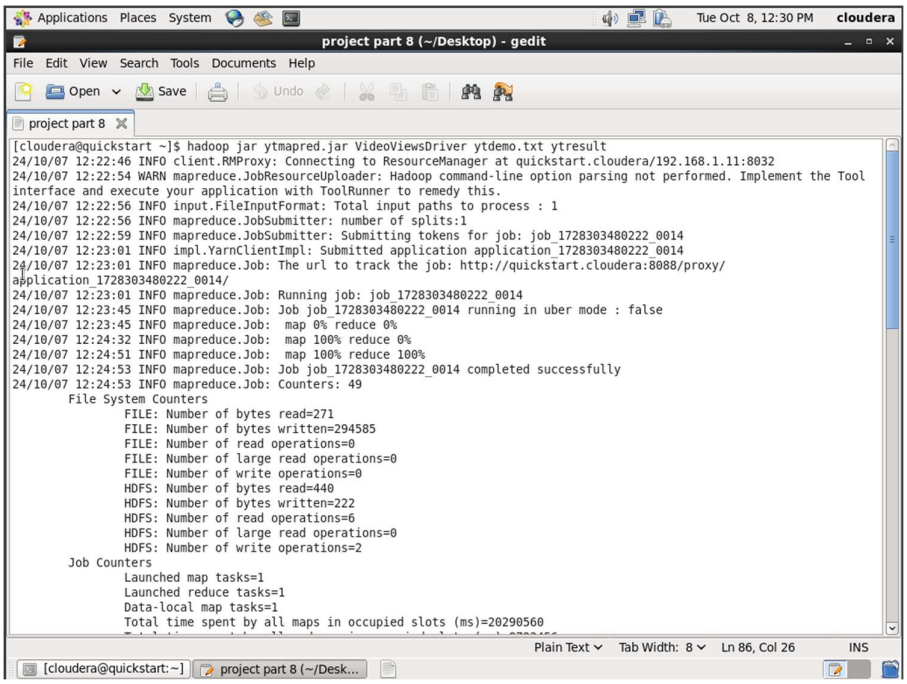
Query 4:

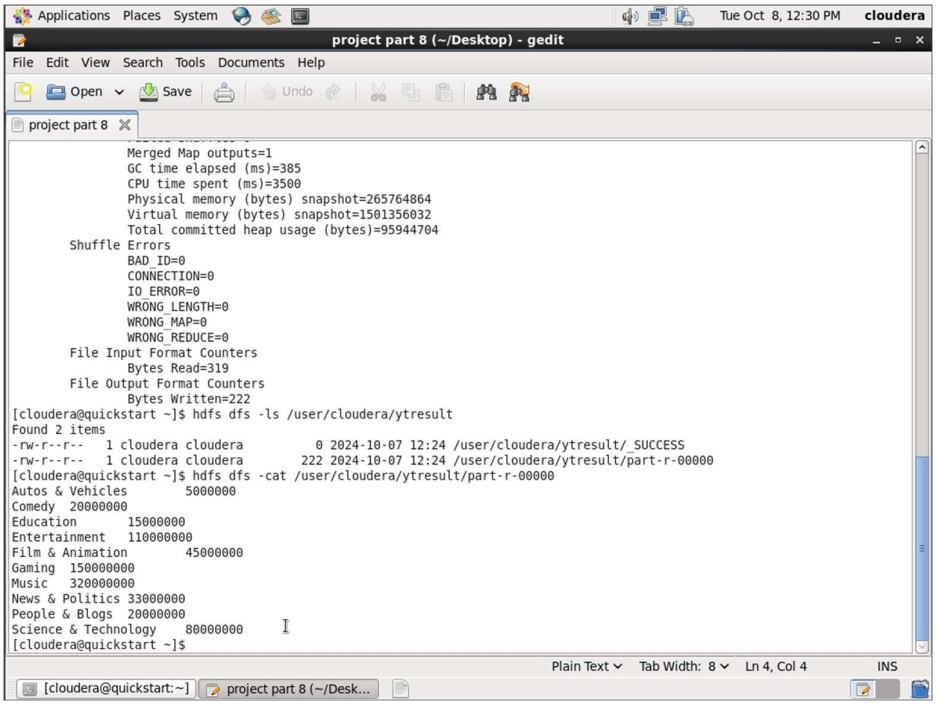


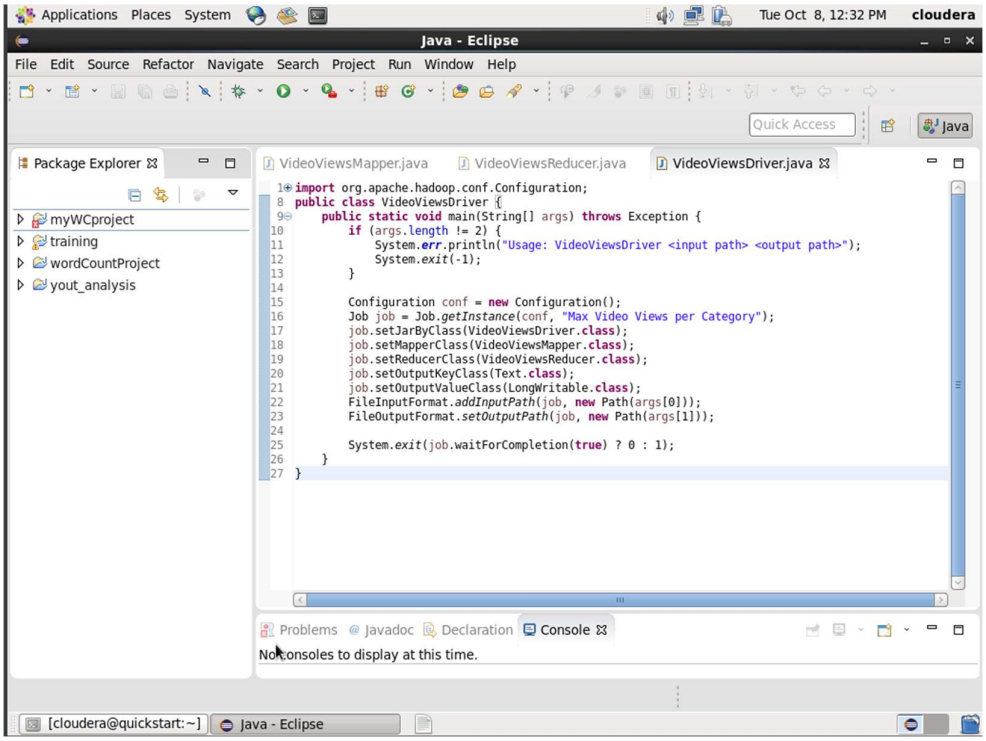


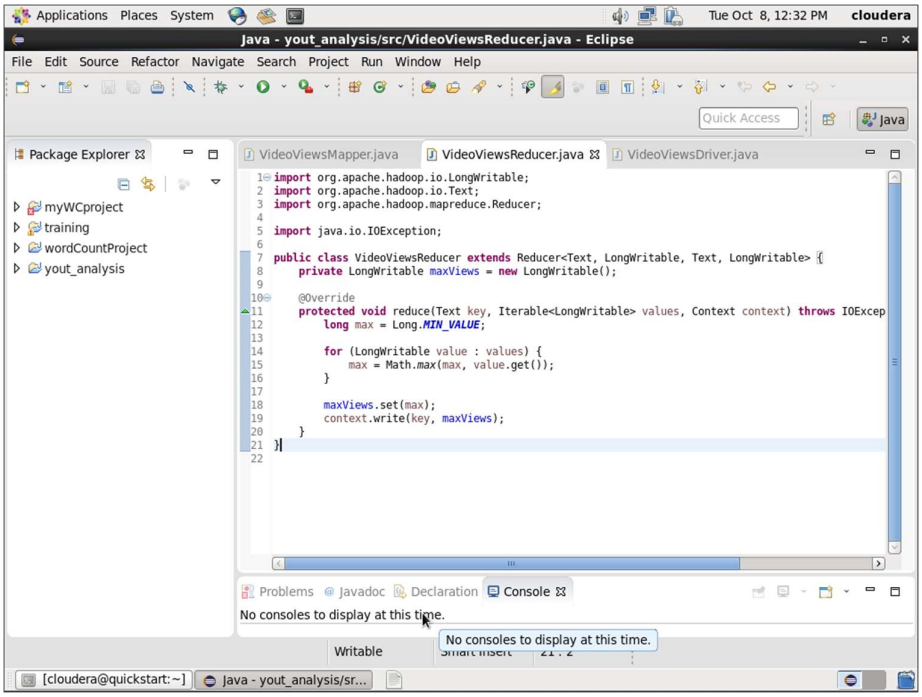
Map reduce program to read the input from HDFS and write the results back to hdfs:

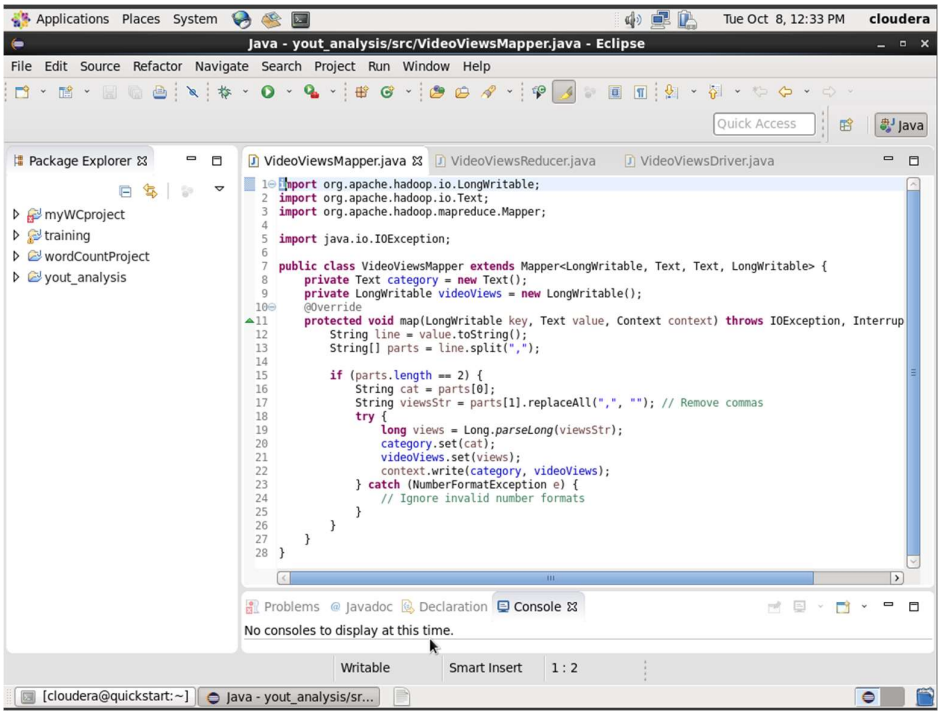

Applications Places System
Tue Oct 8, 12:29 PM
cloudera
project part 7 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
project part 7
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ ls
cloudera-manager exps Public
cm_api.py finalwordCountProject.jar registercopy.java
demo.txt kerberos register.java
departments.txt lib sample1.txt
Desktop Music sample.txt
DhruviDirOnLocal myWCproject.jar students.txt
DirOnLocal parcels Templates
Documents Pictures Videos
Downloads pig_1725525123088.log wordCountFinal.jar
dumA.txt pig_1725526603430.log wordCountProject.jar
dumA.txt~ pig_1725526729742.log wordinput.txt
eclipse pig_1725527991499.log workspace
enterprise-deployment.json pig exp.txt youtubedata.java
express-deployment.json pig exp.txt~ ytmapred.jar
[cloudera@quickstart ~]$ cd /user
bash: cd: /user: No such file or directory
[cloudera@quickstart ~]$ hdfs dfs -put ytmapred.jar /user/cloudera/
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 15 items
drwx------ - cloudera cloudera 0 2024-09-26 05:12 .Trash
drwx------ - cloudera cloudera 0 2024-10-07 06:29 .staging
-rwxrwxrwx 1 cloudera cloudera 194 2024-09-26 04:20 DhruviDir
-rw-r--r-- 1 cloudera cloudera 113 2024-09-05 01:55 departments.txt
drwxr-xr-x - cloudera cloudera 0 2024-09-26 05:05 dhruviDir2
-rw-r--r-- 1 cloudera cloudera 3992 2024-09-04 10:31 finalwordCountProject.jar
-rw-r--r-- 1 cloudera cloudera 3966 2024-09-04 10:38 myWCproject.jar
drwxr-xr-x - cloudera cloudera 0 2024-09-05 00:16 mybank
drwxr-xr-x - cloudera cloudera 0 2024-09-26 05:22 mydirectory
-rw-r--r-- 1 cloudera cloudera 101 2024-09-05 01:53 students.txt
-rw-r--r-- 1 cloudera cloudera 3861 2024-09-04 10:51 wordCountFinal.jar
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
[cloudera@quickstart:~]
project part 7 (~/Desk...


Applications Places System
Tue Oct 8, 12:30 PM
cloudera
project part 8 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
project part 8
[cloudera@quickstart ~]$ hadoop jar ytmapred.jar VideoViewsDriver ytdemo.txt ytresult
24/10/07 12:22:46 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/192.168.1.11:8032
24/10/07 12:22:54 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
24/10/07 12:22:56 INFO input.FileInputFormat: Total input paths to process : 1
24/10/07 12:22:56 INFO mapreduce.JobSubmitter: number of splits:1
24/10/07 12:22:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1728303480222_0014
24/10/07 12:23:01 INFO impl.YarnClientImpl: Submitted application application_1728303480222_0014
24/10/07 12:23:01 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1728303480222_0014/
24/10/07 12:23:01 INFO mapreduce.Job: Running job: job_1728303480222_0014
24/10/07 12:23:45 INFO mapreduce.Job: Job job_1728303480222_0014 running in uber mode : false
24/10/07 12:23:45 INFO mapreduce.Job: map 0% reduce 0%
24/10/07 12:24:32 INFO mapreduce.Job: map 100% reduce 0%
24/10/07 12:24:51 INFO mapreduce.Job: map 100% reduce 100%
24/10/07 12:24:53 INFO mapreduce.Job: Job job_1728303480222_0014 completed successfully
24/10/07 12:24:53 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=271
FILE: Number of bytes written=294585
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=440
HDFS: Number of bytes written=222
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=20290560
Plain Text
Tab Width: 8
Ln 86, Col 26
INS
[cloudera@quickstart:~]
project part 8 (~/Desk...

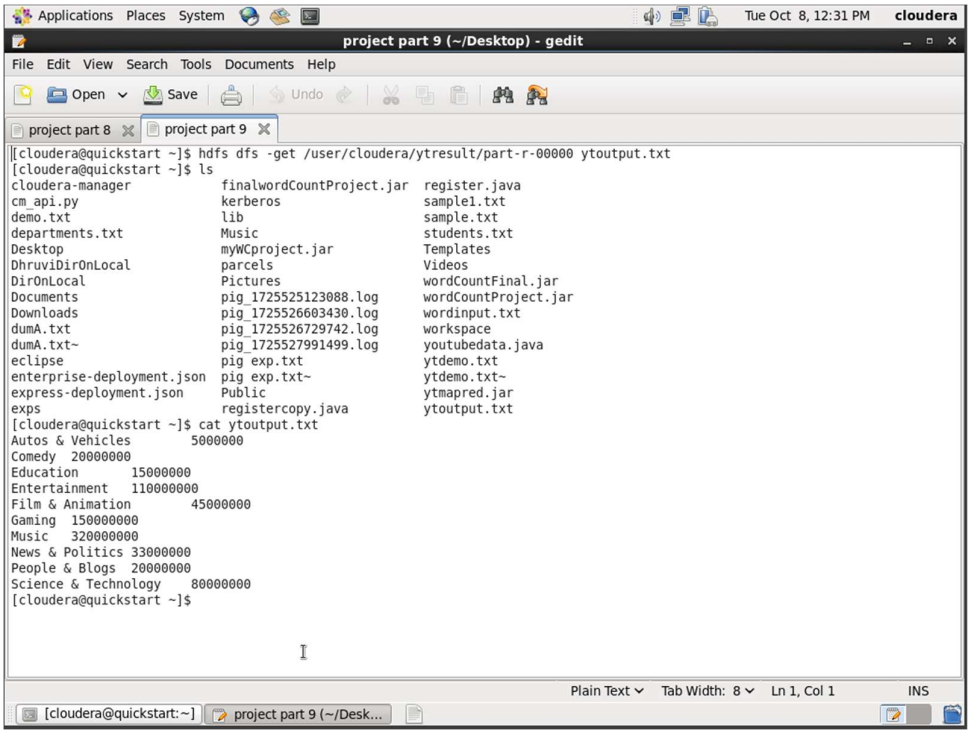

Applications Places System
Tue Oct 8, 12:30 PM cloudera
project part 8 (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
project part 8
Merged Map outputs=1
GC time elapsed (ms)=385
CPU time spent (ms)=3500
Physical memory (bytes) snapshot=265764864
Virtual memory (bytes) snapshot=1501356032
Total committed heap usage (bytes)=95944704
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=319
File Output Format Counters
Bytes Written=222
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/ytresult
Found 2 items
-rw-r--r-- 1 cloudera cloudera 0 2024-10-07 12:24 /user/cloudera/ytresult/_SUCCESS
-rw-r--r-- 1 cloudera cloudera 222 2024-10-07 12:24 /user/cloudera/ytresult/part-r-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/ytresult/part-r-00000
Autos & Vehicles 5000000
Comedy 20000000
Education 15000000
Entertainment 110000000
Film & Animation 45000000
Gaming 150000000
Music 320000000
News & Politics 33000000
People & Blogs 20000000
Science & Technology 80000000
[cloudera@quickstart ~]$
Plain Text Tab Width: 8 Ln 4, Col 4 INS
[cloudera@quickstart:~] project part 8 (~/Desk...


Applications Places System
Tue Oct 8, 12:32 PM cloudera
Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
myWCproject
training
wordCountProject
yout_analysis
VideoViewsMapper.java
VideoViewsReducer.java
VideoViewsDriver.java
import org.apache.hadoop.conf.Configuration;
public class VideoViewsDriver {
public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.err.println("Usage: VideoViewsDriver <input path> <output path>");
System.exit(-1);
}
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Max Video Views per Category");
job.setJarByClass(VideoViewsDriver.class);
job.setMapperClass(VideoViewsMapper.class);
job.setReducerClass(VideoViewsReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(LongWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
Problems Javadoc Declaration Console
No consoles to display at this time.
[cloudera@quickstart:~] Java - Eclipse

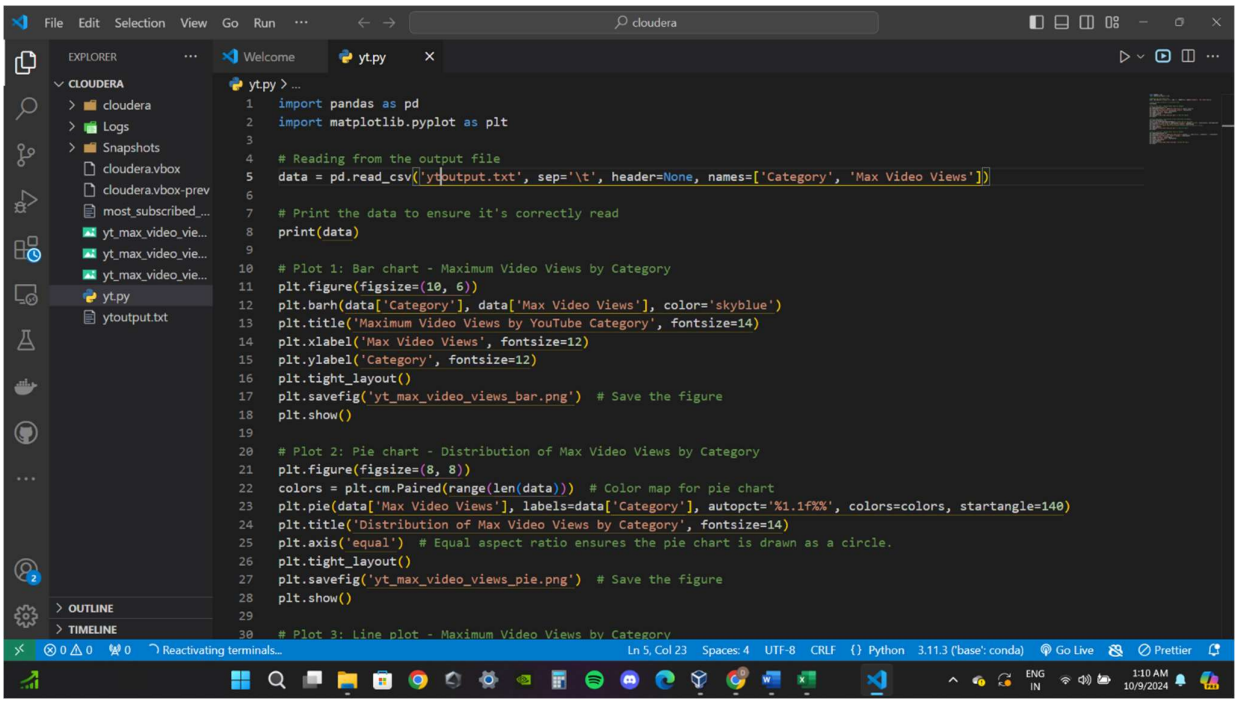

Applications Places System
Tue Oct 8, 12:32 PM cloudera
Java - yout_analysis/src/VideoViewsReducer.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
myWCproject
training
wordCountProject
yout_analysis
VideoViewsMapper.java
VideoViewsReducer.java
VideoViewsDriver.java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
public class VideoViewsReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
private LongWritable maxViews = new LongWritable();
@Override
protected void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOExcep
long max = Long.MIN_VALUE;
for (LongWritable value : values) {
max = Math.max(max, value.get());
}
maxViews.set(max);
context.write(key, maxViews);
}
}
Problems Javadoc Declaration Console
No consoles to display at this time.
No consoles to display at this time.
Writable
[cloudera@quickstart:~]
Java - yout_analysis/sr...


Applications Places System
Tue Oct 8, 12:33 PM cloudera
Java - yout_analysis/src/VideoViewsMapper.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
myWCproject
training
wordCountProject
yout_analysis
VideoViewsMapper.java
VideoViewsReducer.java
VideoViewsDriver.java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
public class VideoViewsMapper extends Mapper<LongWritable, Text, Text, LongWritable> {
private Text category = new Text();
private LongWritable videoViews = new LongWritable();
@Override
protected void map(LongWritable key, Text value, Context context) throws IOException, Interrup
String line = value.toString();
String[] parts = line.split(",");
if (parts.length == 2) {
String cat = parts[0];
String viewsStr = parts[1].replaceAll(",", ""); // Remove commas
try {
long views = Long.parseLong(viewsStr);
category.set(cat);
videoViews.set(views);
context.write(category, videoViews);
} catch (NumberFormatException e) {
// Ignore invalid number formats
}
}
}
}
Problems Javadoc Declaration Console
No consoles to display at this time.
Writable
Smart Insert
1 : 2
[cloudera@quickstart:~]
Java - yout_analysis/sr...

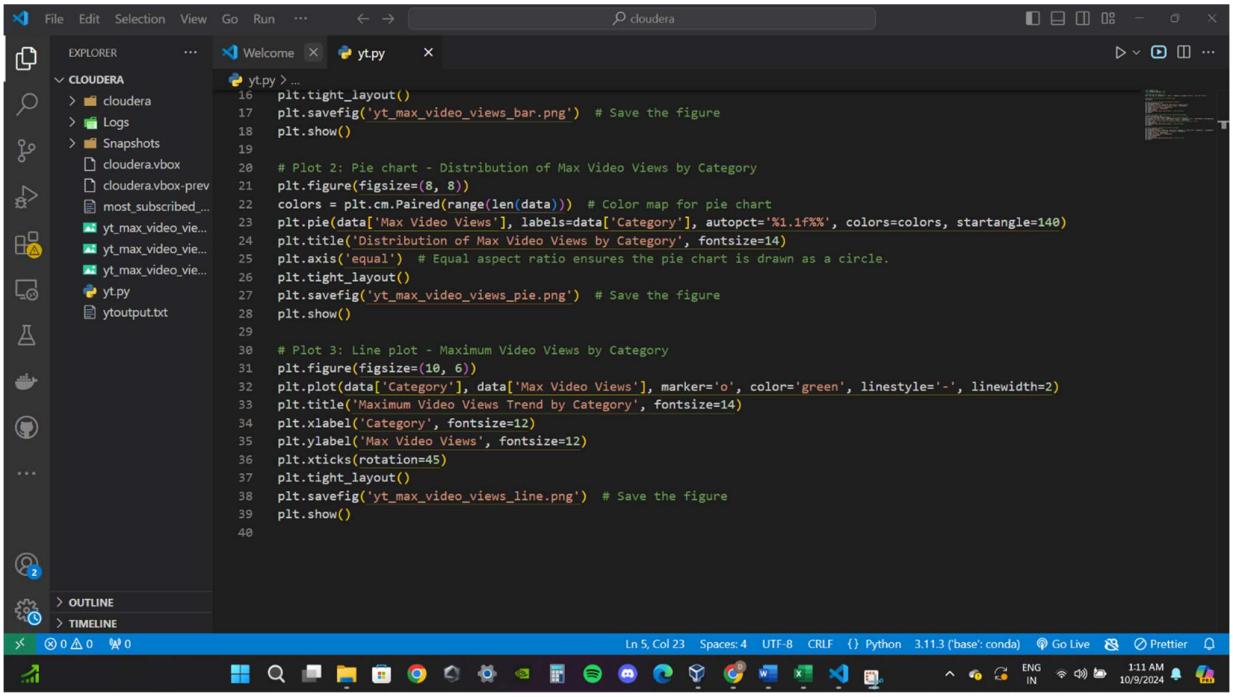
Preparing the output for python plotting:



Create a python program to integrate with Hive and get the required analysis from the Map-reduce result:



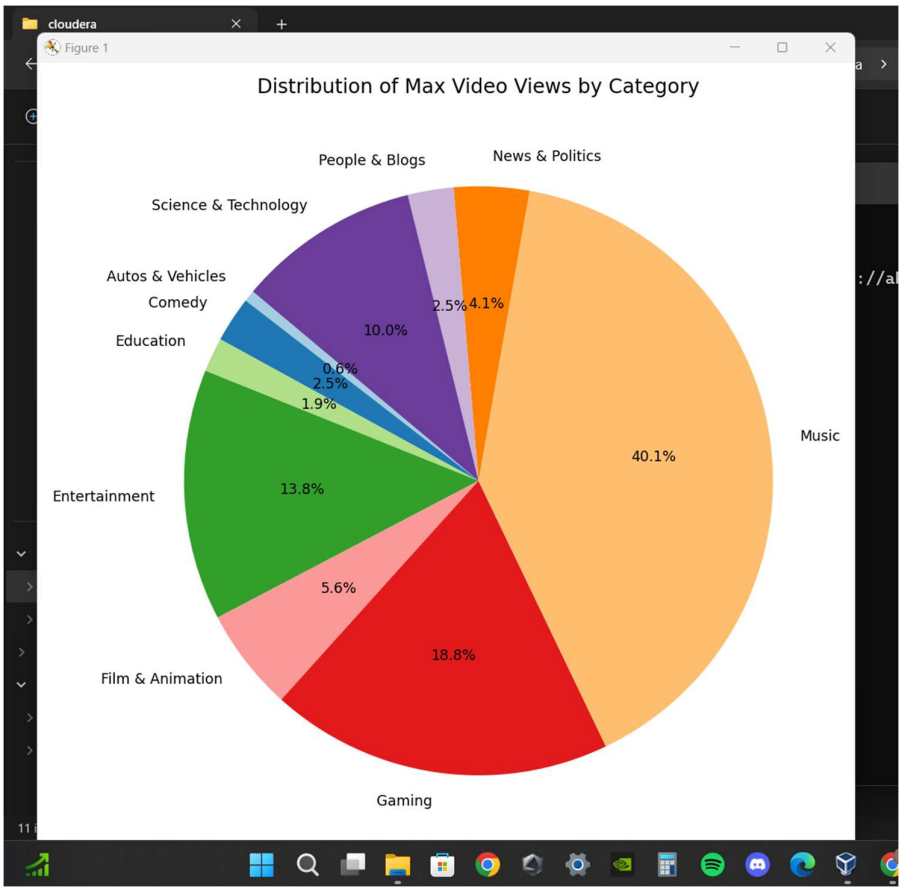
```python
import pandas as pd
import matplotlib.pyplot as plt

# Reading from the output file
data = pd.read_csv('ytoutput.txt', sep='\t', header=None, names=['Category', 'Max Video Views'])

# Print the data to ensure it's correctly read
print(data)

# Plot 1: Bar chart - Maximum Video Views by Category
plt.figure(figsize=(10, 6))
plt.barh(data['Category'], data['Max Video Views'], color='skyblue')
plt.title('Maximum Video Views by YouTube Category', fontsize=14)
plt.xlabel('Max Video Views', fontsize=12)
plt.ylabel('Category', fontsize=12)
plt.tight_layout()
plt.savefig('yt_max_video_views_bar.png')  # Save the figure
plt.show()

# Plot 2: Pie chart - Distribution of Max Video Views by Category
plt.figure(figsize=(8, 8))
colors = plt.cm.Paired(range(len(data)))  # Color map for pie chart
plt.pie(data['Max Video Views'], labels=data['Category'], autopct='%1.1f%%', colors=colors, startangle=140)
plt.title('Distribution of Max Video Views by Category', fontsize=14)
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is drawn as a circle.
plt.tight_layout()
plt.savefig('yt_max_video_views_pie.png')  # Save the figure
plt.show()

# Plot 3: Line plot - Maximum Video Views by Category
```



```python
plt.tight_layout()
plt.savefig('yt_max_video_views_bar.png')  # Save the figure
plt.show()

# Plot 2: Pie chart - Distribution of Max Video Views by Category
plt.figure(figsize=(8, 8))
colors = plt.cm.Paired(range(len(data)))  # Color map for pie chart
plt.pie(data['Max Video Views'], labels=data['Category'], autopct='%1.1f%%', colors=colors, startangle=140)
plt.title('Distribution of Max Video Views by Category', fontsize=14)
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is drawn as a circle.
plt.tight_layout()
plt.savefig('yt_max_video_views_pie.png')  # Save the figure
plt.show()

# Plot 3: Line plot - Maximum Video Views by Category
plt.figure(figsize=(10, 6))
plt.plot(data['Category'], data['Max Video Views'], marker='o', color='green', linestyle='-', linewidth=2)
plt.title('Maximum Video Views Trend by Category', fontsize=14)
plt.xlabel('Category', fontsize=12)
plt.ylabel('Max Video Views', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('yt_max_video_views_line.png')  # Save the figure
plt.show()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\SURESH PATEL\VirtualBox VMs\cloudera> python yt.py
              Category  Max Video Views
0      Autos & Vehicles          5000000
1               Comedy         20000000
2            Education         15000000
3        Entertainment        110000000
4      Film & Animation         45000000
5               Gaming        150000000
6                Music        320000000
7       News & Politics         33000000
8        People & Blogs         20000000
9  Science & Technology         80000000
PS C:\Users\SURESH PATEL\VirtualBox VMs\cloudera>
```

Visualization



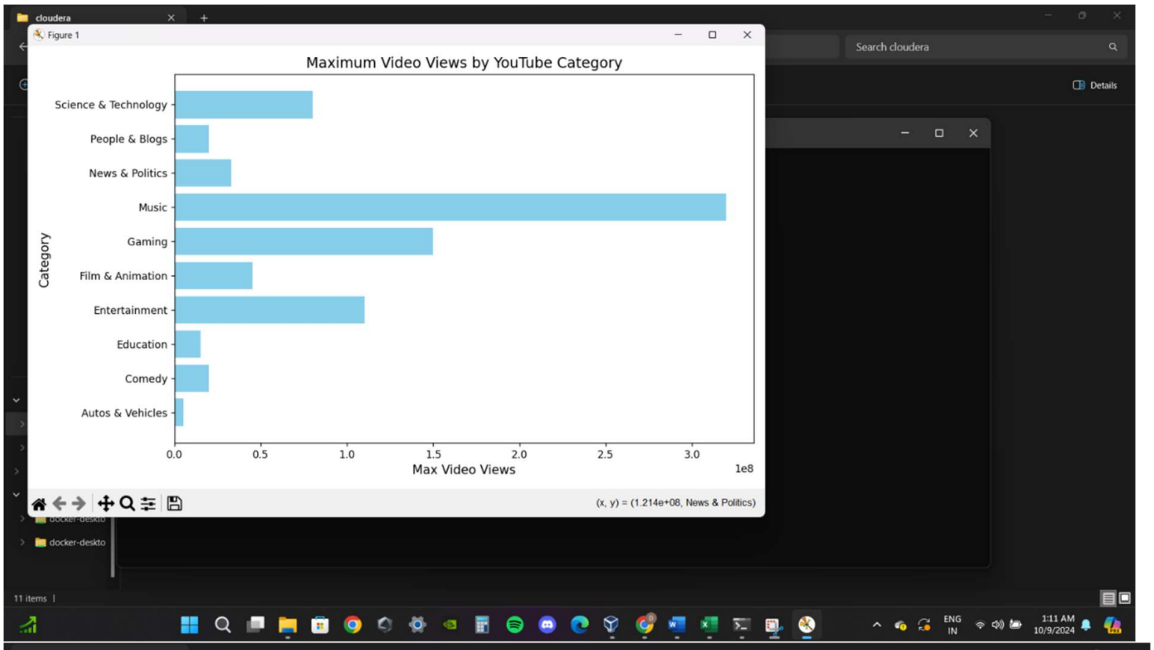


Figure 1
Maximum Video Views by YouTube Category
Science & Technology
People & Blogs
News & Politics
Music
Gaming
Film & Animation
Entertainment
Education
Comedy
Autos & Vehicles
Category
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e8
Max Video Views
(x, y) = (1.214e+08, News & Politics)

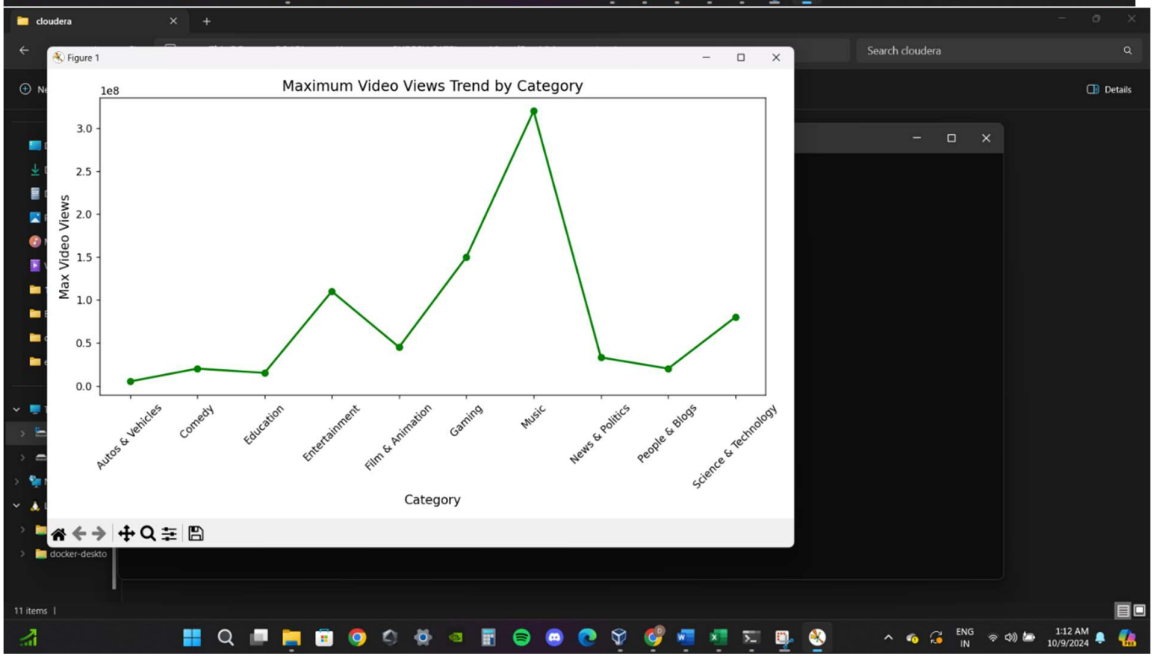
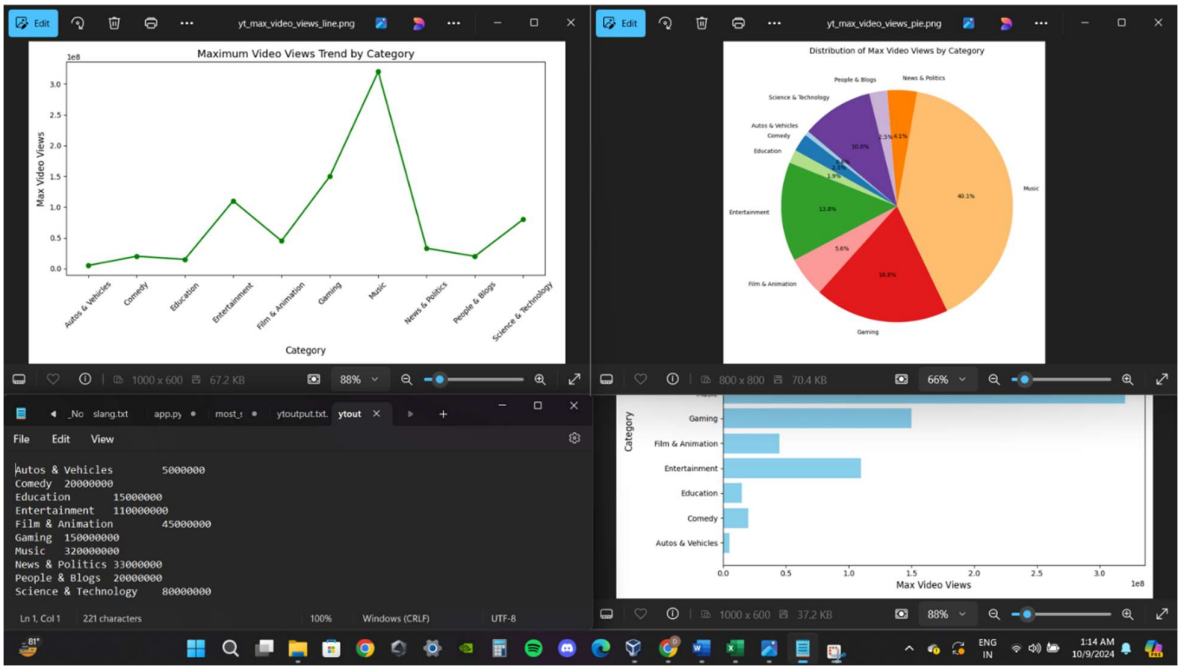


Figure 1
Maximum Video Views Trend by Category
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Max Video Views
Autos & Vehicles
Comedy
Education
Entertainment
Film & Animation
Gaming
Music
News & Politics
People & Blogs
Science & Technology
Category



1. Maximum Video Views by Category:

The analysis of maximum video views by category allows us to understand content popularity across various genres on YouTube. By using horizontal bar charts, we can easily identify which categories garner the most views, offering insights into viewer preferences and content trends. This visual representation facilitates quick comparisons between categories, highlighting the most successful niches.

2. Distribution of Max Video Views by Category:

The pie chart visualization effectively illustrates the proportional distribution of maximum video views among different YouTube categories. This representation provides a clear overview of the market share held by each category, helping stakeholders recognize dominant content areas and potential gaps in the market. By visualizing data in this format, we can emphasize the relative popularity of each category, aiding in content strategy decisions.

3. Trend Analysis of Maximum Video Views:

The line plot showcasing the trend of maximum video views by category enables a deeper analysis of how popularity evolves over time. By plotting data points connected with lines, we can observe fluctuations, peaks, and troughs in viewership. This visualization allows us to

detect seasonal patterns, emerging trends, and shifts in audience engagement, supporting predictive analytics and strategic content planning.

4. Interactive Data Exploration:

Leveraging interactive visualizations in tools like Python's Matplotlib enhances user engagement with the data. Stakeholders can explore the visualized results more dynamically, facilitating a better understanding of viewer behavior and preferences. Interactive elements allow users to filter data, zoom in on specific trends, or compare different categories, fostering a more comprehensive exploration of the dataset.

5. Conclusion of findings :

The overall visual analysis provides critical insights into the YouTube ecosystem, helping to decode audience behavior and content consumption patterns. By utilizing various visualization techniques, we empower content creators and marketers with actionable insights, enabling them to refine their strategies, optimize content delivery, and align with user preferences effectively.

This visualization theory integrates the findings from the code and provides a structured understanding of how different visualizations contribute to the analysis of YouTube data. Let me know if you need any adjustments or additional information!

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

Conclusion:

The YouTube data analysis pipeline effectively demonstrates how big data technologies like MySQL, HBase, Hive, and MapReduce can be combined to process large datasets for insightful analysis. Through the integration of distributed storage and computation frameworks, the pipeline allows for efficient data ingestion, storage, processing, and visualization. The use of Python's Pandas and Matplotlib further enhances the ability to generate comprehensive visual reports. This solution addresses the challenges of handling large-scale YouTube data, such as extracting meaningful insights about trends in video views across different categories.

Future Scope:

1. Scalability with Spark: The current system uses MapReduce for large-scale data processing, but transitioning to Apache Spark could significantly reduce processing

time due to its in-memory computing capabilities, making the pipeline faster and more efficient for even larger datasets..

2. Real-Time Analysis with Kafka: Incorporating real-time data streaming solutions like Apache Kafka could enable real-time analytics on YouTube data, allowing for immediate insights into trends and user engagement.

3. Machine Learning Integration: Applying machine learning algorithms to the YouTube dataset could open up possibilities for predictive analytics. For example, predicting the growth of certain content categories, audience preferences, or identifying potential viral videos.

4. Dashboard Integration: Incorporating tools like Tableau or Power BI to visualize the results could provide an interactive dashboard for stakeholders to monitor YouTube data trends continuously, enabling decision-makers to get real-time insights.

5. Support for Unstructured Data: Extending the pipeline to process other unstructured data sources (such as video metadata, user comments, etc.) could offer a more holistic view of YouTube content, including sentiment analysis and user behavior patterns.

6. Ethical Considerations: Address ethical considerations such as data privacy, algorithmic bias, and transparency in data usage to ensure responsible and equitable practices in data-driven decision-making.

These enhancements can transform the current data pipeline into a more robust and real-time system that addresses modern big data needs for YouTube analysis.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Written Assignment 1

Q1 Write briefly on Big data characteristics, Hadoop architecture, Hadoop ecosystem.

Ans1 Big Data characteristics:-

1) Volume - Big data is characterised by sheer volume of data generated daily. This data can range from terabytes to petabytes.
2) Velocity - Data is generated and collected at an unprecidented speed.
3) Variety - Big data encompasses various data types including structured and unstructured data.
4) Veracity - Big data can be noisy and unreliable. Veracity refers to the quality and trustworthiness of data. It is essential to handle inaccurate data.
5) Value - Extracting valuable insights from big data is a primary objective. This requires advanced analytics and processing of turning data into actionable information.
6) Variability - Data can exhibit seasonable or temporal patterns Understanding data variability is crucial for accurate analysis.
7) Complexity - Big data often involves complex relationships within the data, requiring the sophestrated algorithm and tools for analysis.

Hadoop Architecture:-

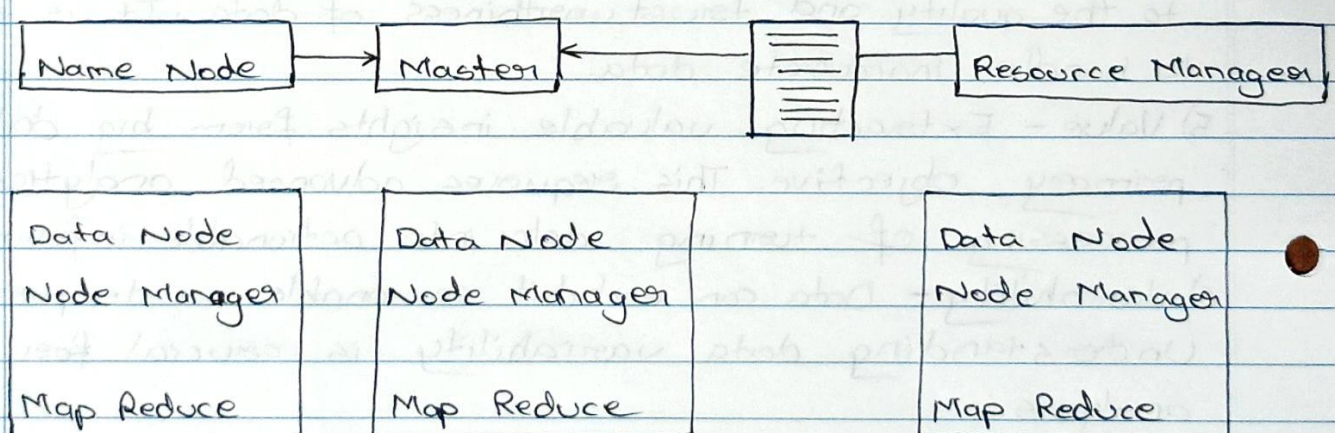
1) HDFS - Hadoop Distributed File System is a storage component of hadoop. It divides large files into smaller blocks and stores multiple copies of each block on different

nodes on a cluster.

2) Map Reduce - It is a programming model and processing engine for distributed data processing. It divides task into two phases mapping (data processing) and reducing (aggregation).

3) YARN (Yet Another Resource Negotiator) - YARN is the resource management layer of Hadoop. It allocates resources and schedules task across the cluster, making it possible to run various applications alongside Hadoop such as spark and Hive.

4) Common Utilities - Hadoop includes a set of utilities and libraries for managing and operating the cluster.



Hadoop Ecosystem:-

The hadoop ecosystem consists of various projects and tools that extend the capabilities of the core hadoop framework.

1) Apache Hive - A data warehousing and SQL like query language for Hadoop.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

2) Apache Pig - A high level platform for creating MapReduce programs with a simpler scripting language.
3) Apache HBase - A NoSQL database that produces real time/write access to Hadoop data.
4) Apache Spark - A fast in memory data processing engine that can replace or complement MapReduce for certain workloads.
5) Apache Kafka - A distributed streaming platform for handling real time data feeds and stream processing.
6) Apache sqoop - A tool for efficiently transferring data between Hadoop and structured data stores like relational databases.
7) Apache ZooKeeper - A centralised service for maintaining configuration information, naming and providing distributed synchronization.
8) Apache Mahout - A machine learning library for creating scalable, machine learning algorithms.
9) Apache Oozie - A workflow scheduler for managing hadoop job and processes.
10) Apache Flume - A data ingestion tool for collecting, aggregation and moving large amounts of streaming data into hadoop.

AD
5/9/24.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Written Assignment 2

Q1 Mining Social Network Graphs
- clustering of social graph using Nigvan-Newman algorithm.
- Direct discovery of communities is social graph using Clique percolation method (CPM).

Ans 1 Mining social network graphs in Big Data Analytics.

Mining social network graphs is a critical task in big data analytics, especially when dealing with large scale networks like Facebook, Twitter or LinkedIn. It involves extracting valuable insights, patterns and communities from these graphs to understand the user behaviour, target marketing efforts detect anomalies and more. Here we will focus on two specific aspects, clustering using the Nigvan-Newman algorithm. and direct discovery of communities using clique percolation.

- Clustering of social graph using Newman-Girvan algorithms-

Newman-Girvan Algorithm- The Newman Girvan algorithm also known as Girvan Newman algorithm is a widely used technique for community detection in social graphs. It is based on the concept of edge betweenness centrality which quantifies the importance of an edge in connecting different communications.

Algorithm Steps-

1) Calculate the betweenness centrality of all edges in the social graph. This measures how often an edge lies on the shortest path between two nodes.

2) Remove the edges with the highest betweenness centrality. This step involves iteratively identifying and removing edges, that bridge different communities.
3) Calculate the modularity of the graph decreases.

Advantages-
1) The Newman-Girvan algorithm is effective in detecting communities of various sizes and shapes, makes it suitable for analyzing the diverse social network structures.
2) Direct discovery of communities in social graphs using clique prelocation.

- Clique Percolation Method -
The clique percolation method is a direct approach to discovering overlapping communities within social graphs. Unlike traditional clustering, which assumes non overlapping communities, this method acknowledges that individuals can belong to multiple communities simultaneously.

Algorithm steps-
1) Identify all cliques of a given size(k) in the social graph.
2) Create a clique graph where each node is representing a clique and hence there is an edge between two nodes in their corresponding cliques that overlap by (k-1) nodes or more.
3) Apply percolation theory to clique graph, where nodes (cliques) that percolate represent overlapping communities.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Advantages-

The clique method is particularly useful for capturing overlapping communities in social graphs, where individuals ~~are~~ often participate in multiple social codes. It provides a more realistic view of social network structures.

Girvan Newman:-

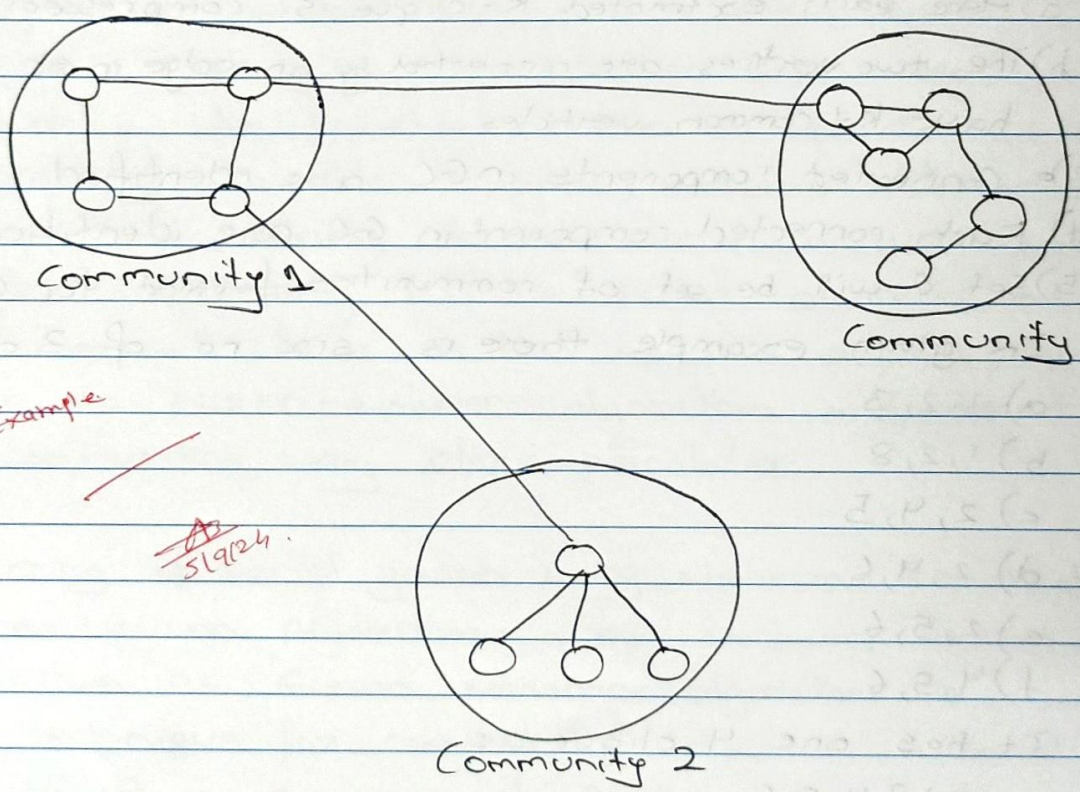
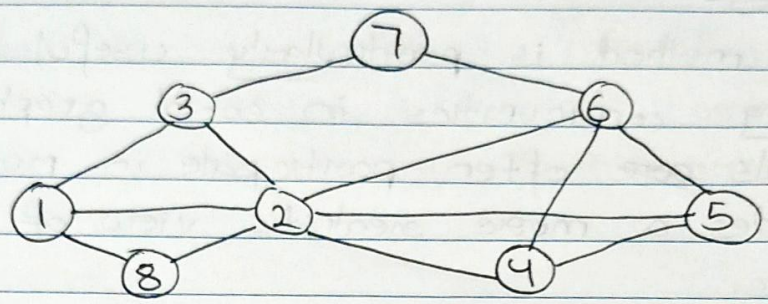


diagram / Example

5/9/24

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CPM:-



1) All K cliques are extracted in graph G are extracted.
2) A new clique graph GC is created-
   a) Here each extracted K-clique is compressed as one vertex.
   b) The two vertices are connected by an edge in GC if they have K-1 common vertices.
3) Connected components in GC are identified.
4) Each connected component in GC are identified.
5) Set C will be set of communities formed for G

In given example there is six no of 3 cliques are:

a) 1, 2, 3
b) 1, 2, 8
c) 2, 4, 5
d) 2, 4, 6
e) 2, 5, 6
f) 4, 5, 6

It has one 4 clique as:

g: 2, 4, 5, 6

∴ New graph