**732\_Saksham**

**Msc DSAI (Sem -II)**

**Big Data Analytics**

**INDEX**

| **Sr No** | **Practical Name** |
| --- | --- |
| **1** | **Basic Commands of Linux and Hadoop** |
| **2** | **WordCount Application using Java and MapReduce** |
| **3** | **Character Counting Application using Python and MapReduce** |
| **4** | **Whether Analysis Application using Python and MapReduce** |
| **5** | **Matrix Multiplication Application using Python and MapReduce** |
| **6** | **Using Hive Tool** |
| **7** | **Using Pig Tool and creating Pig Latin Scripts** |
| **8** | **Using HBase Tool** |
| **9** | **Using Document-based Database - MongoDB** |
| **10** | **Using Impala Tool** |

**Practical : 01**

**Aim :** Basic Commands of Linux and Hadoop

Here are some basic commands for both Linux and Hadoop:

Linux Commands:

1. ls - List files in the current directory.

Syntax: ls

1. cd - Change to a different directory.

Syntax: cd directory\_name

1. mkdir - Create a new directory.

Syntax: mkdir directory\_name

1. cp - Copy a file.

Syntax: cp source\_file destination\_file

1. mv - Move a file.

Syntax: mv source\_file destination\_file

1. rm - Remove a file.

Syntax: rm file\_name

1. pwd - Print the current working directory.

Syntax: pwd

1. cat - Display the contents of a file.

Syntax: cat file\_name

1. grep - Search for a specific string in a file.

Syntax: grep search\_string file\_name

Hadoop Commands:

1. hdfs dfs -ls - List files in the HDFS.

Syntax: hdfs dfs -ls

1. hdfs dfs -put - Copy a file from the local file system to HDFS.

Syntax: hdfs dfs -put source\_file destination\_directory

1. hdfs dfs -get - Copy a file from HDFS to the local file system.

Syntax: hdfs dfs -get source\_file destination\_directory

1. hdfs dfs -mkdir - Create a new directory in HDFS.

Syntax: hdfs dfs -mkdir directory\_name

1. hdfs dfs -rm - Remove a file from HDFS.

Syntax: hdfs dfs -rm file\_name

1. hdfs dfs -mv - Move a file in HDFS.

Syntax: hdfs dfs -mv source\_file destination\_file

1. hdfs dfs -cat - Display the contents of a file in HDFS.

Syntax: hdfs dfs -cat file\_name

1. hdfs dfs -du - Display the size of a file or directory in HDFS.

Syntax: hdfs dfs -du file\_name/directory\_name

1. hdfs dfs -chmod - Change the permissions of a file or directory in HDFS.

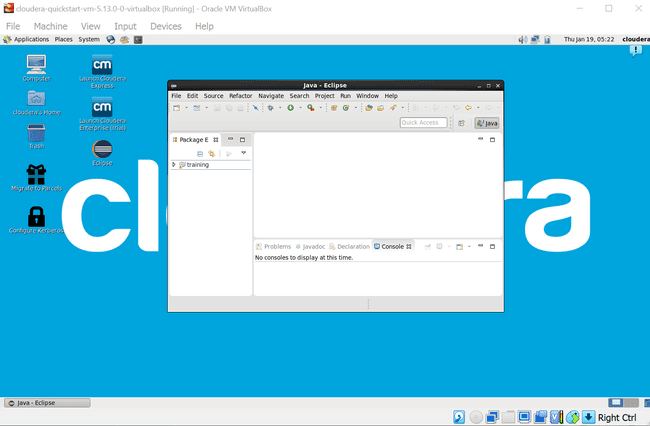
Syntax: hdfs dfs -chmod permission file\_name/directory\_name

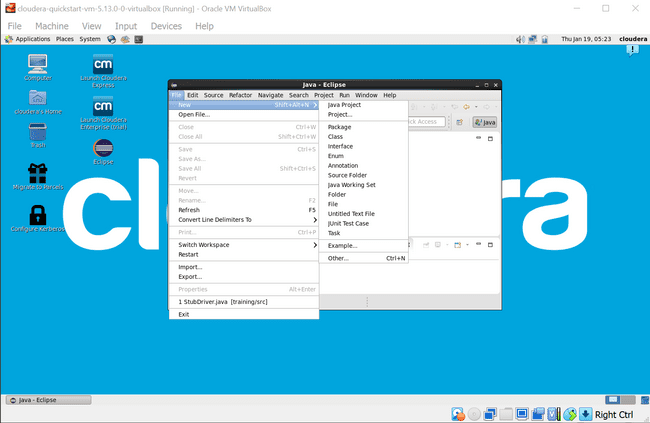
Overall, these are some of the basic Linux and Hadoop commands that you can use to navigate and manage files in the file system. There are many more commands available in both Linux and Hadoop that you can use depending on your needs.

**Practical : 02**

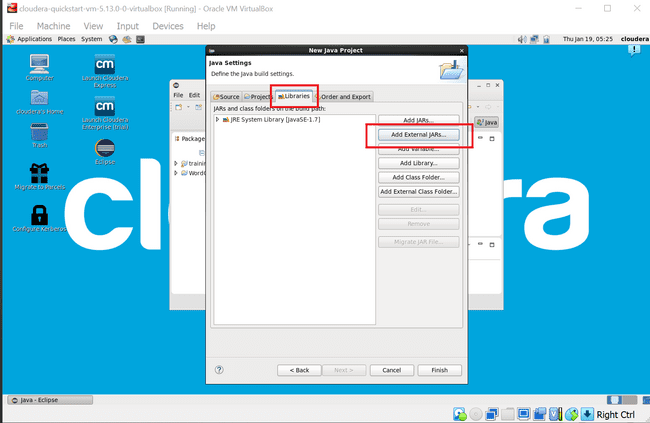
Aim : To Implement WordCount problem using Hadoop MapReduce in Eclips.

Step 1 : Run your cloudera system. Open Eclips.



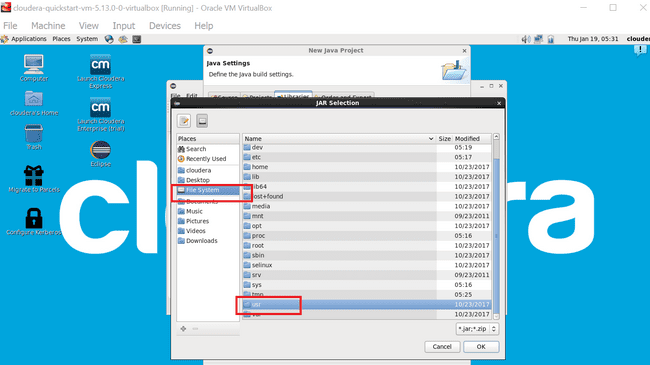
Step 2 : Click on File > New > java project. Give Project Name (“WordCount”). 

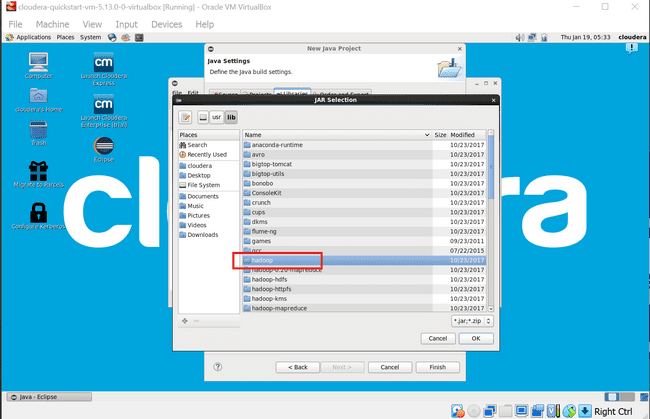
Step 3 : Click on Libraries tab. Then click on Add External JARs… Tab To add Hadoop Libraries.



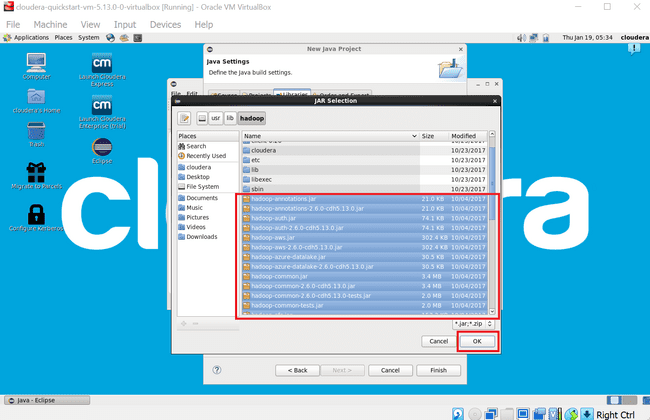
Step 4 : Follow this steps : Click on File System -> usr -> lib -> hadoop (Select all the libraries (JAR files) -> Click OK.

Again Click on Add External JARs… -> client -> select all jar files -> ok -> Finish.

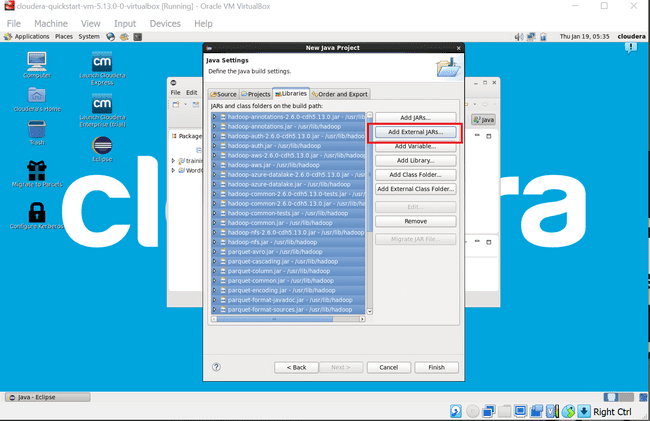


Step : 5

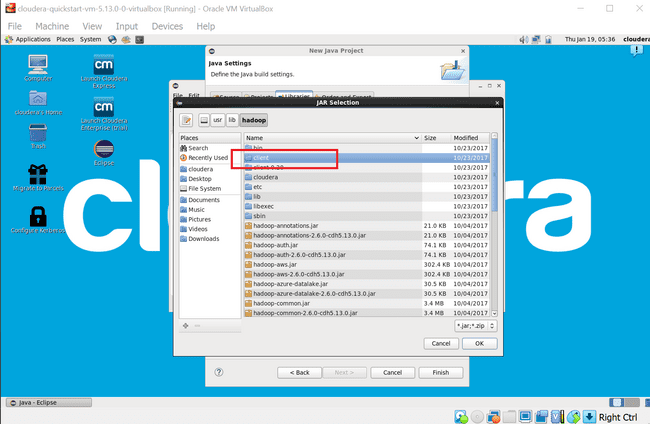
Step : 6



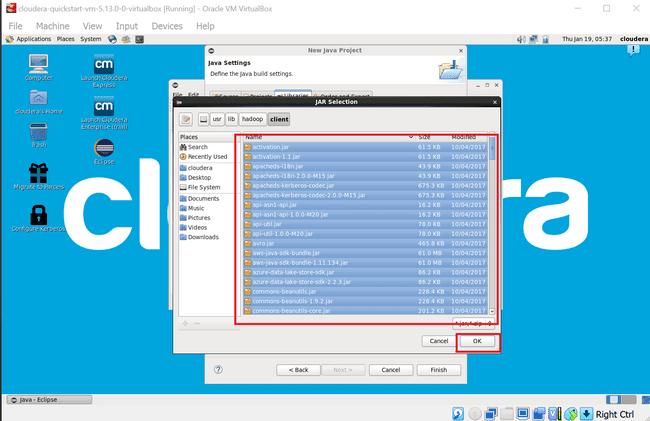
Step : 7



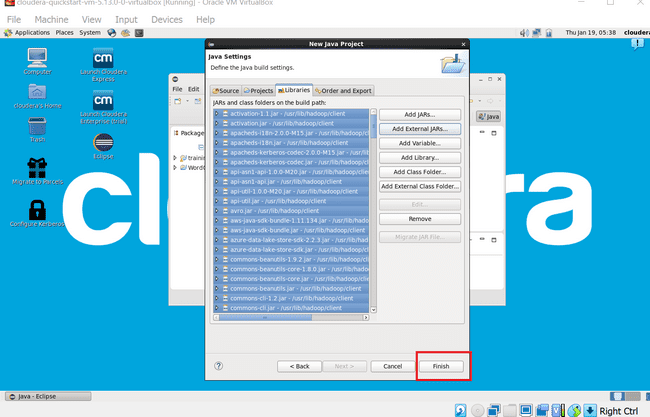
Step : 8



Step : 9



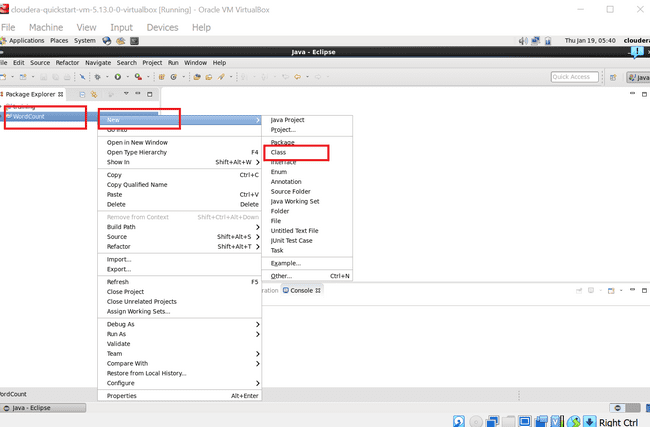
Step : 10



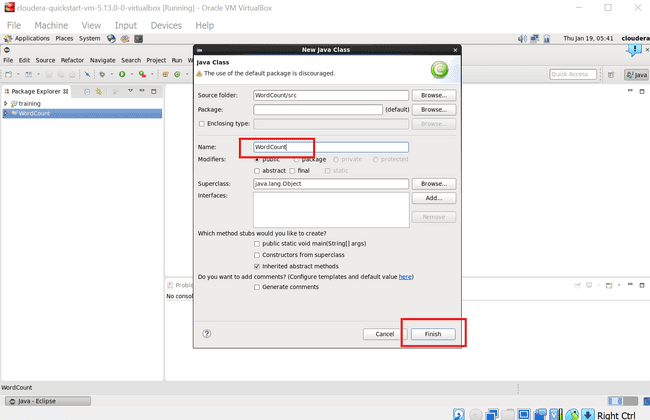
Step 11 : Now you will see project name “WordCount” on sidebar. Right Click on Project name “WordCount” -> New -> class.

For package write name “WordCloud” -> Finish.

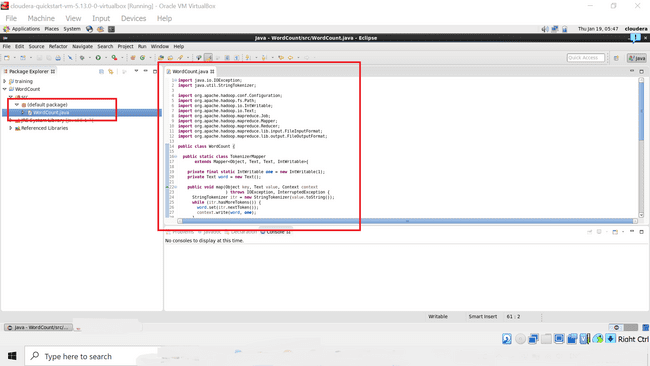
WordCount.java window will pop up.



Step : 12



Step 13 : Write your code in window(WordCount.java).



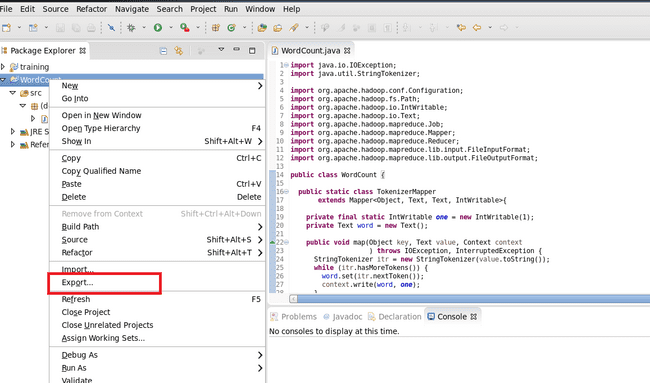
Step 14 : Right Click on the project name

WordCount -> Export -> Java -> JAR File -> Next ->

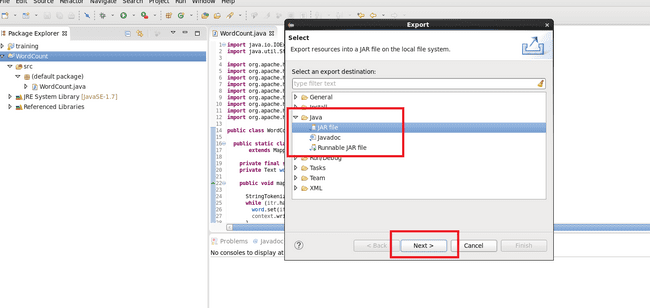
For select the export destination for JAR file:

browse -> Name : WordCount.jar -> save in folder -> cloudera -> Finish -> OK

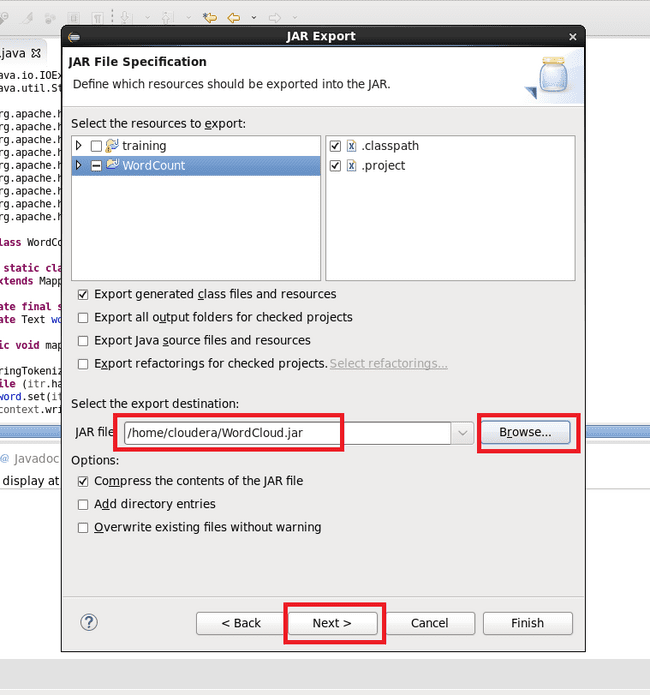
(\*\* I used “WordCloud.jar” here)



Step : 15



Step : 16



Step 17 : Now open terminal . follow this commands.

[cloudera@quickstart ~]$ ls

cloudera-manager cm\_api.py Desktop Documents Downloads eclipse

enterprise-deployment.json express-deployment.json kerberos lib

Music parcels Pictures Public Templates Videos WordCloud.jar

Workspace

[cloudera@quickstart ~]$ pwd

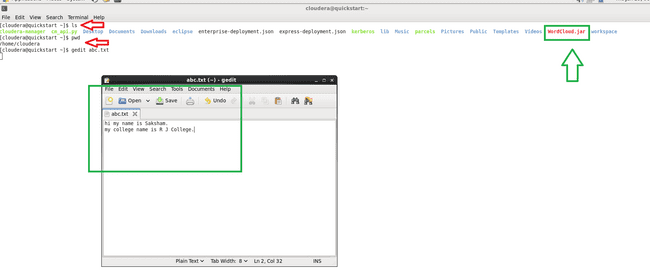
/home/cloudera

[cloudera@quickstart ~]$ gedit abc.txt

[cloudera@quickstart ~]$ cat abc.txt

hi my name is Saksham.

my college name is R J College.



Step 18 :

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 6 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -mkdir /inputdir

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 7 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:04 /inputdir

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -put /home/cloudera/abc.txt /inputdir/

[cloudera@quickstart ~]$ hdfs dfs -ls /inputdir

Found 1 items

-rw-r--r-- 1 cloudera supergroup 55 2023-01-19 06:05 /inputdir/abc.txt

[cloudera@quickstart ~]$ hdfs dfs -cat /inputdir/abc.txt

hi my name is Saksham.

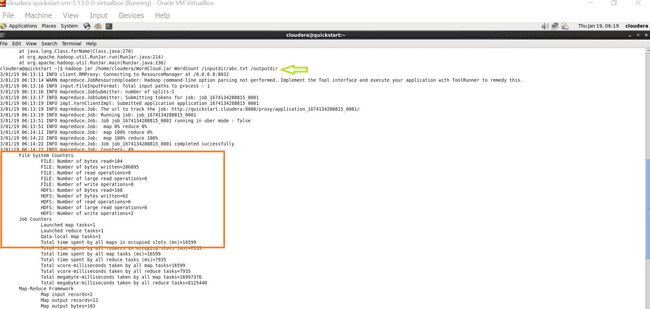
my college name is R J College.



Step 19 :

[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCloud.jar WordCount /inputdir/abc.txt /outputdir

It will Start executing the program.



Step 20 :

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 8 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:05 /inputdir

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:14 /outputdir

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -ls /outputdir

Found 2 items

-rw-r--r-- 1 cloudera supergroup 0 2023-01-19 06:14 /outputdir/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 62 2023-01-19 06:14 /outputdir/part-r-00000

**Final Output :**

[cloudera@quickstart ~]$ hdfs dfs -cat /outputdir/part-r-00000

College. 1

J 1

R 1

Saksham. 1

college 1

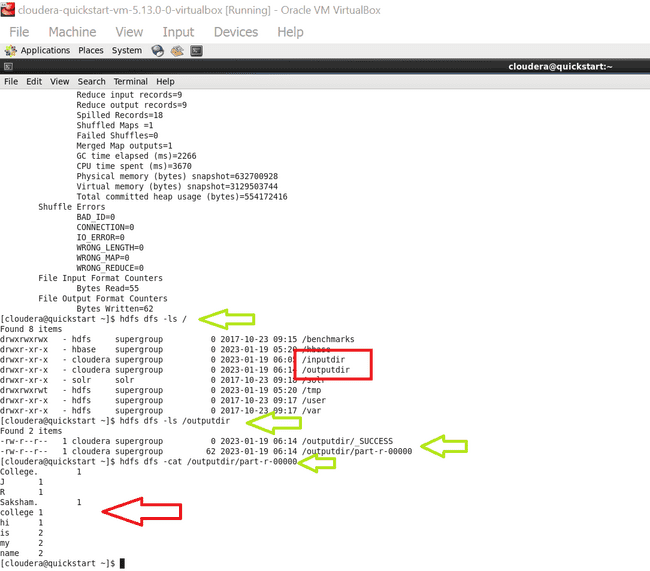
hi 1

is 2

my 2

name 2

[cloudera@quickstart ~]$



*Thank you.*

**Practical : 03**

**Aim : Character Counting Application using Eclips and MapReduce**

Step 1 : Run your cloudera system. Open Eclips.

Step 2 : Click on File > New > java project. Give Project Name (“CharCount”).

Step 3 : Click on Libraries tab. Then click on Add External JARs… Tab To add Hadoop Libraries.

Step 4 : Follow this steps :

Click on File System -> usr -> lib -> hadoop (Select all the libraries (JAR files) -> Click OK.

Again Click on Add External JARs… -> client -> select all jar files -> ok -> Finish.

Step 5 : Now you will see project name “CharCount” on sidebar.

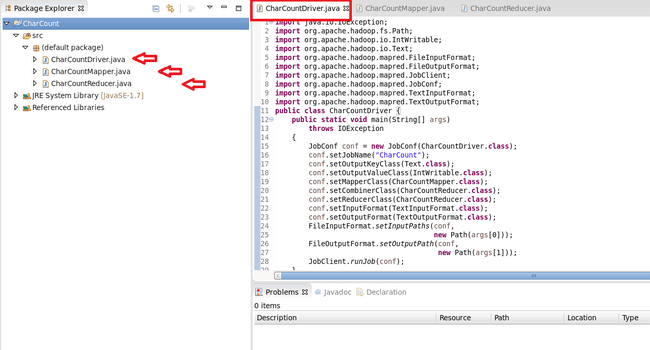
Right Click on Project name “CharCount” -> New -> class.

Create 3 new classes named :

**CharCountDriver**(having the main function),

**CharCountMapper**,

**CharCountReducer.**

****

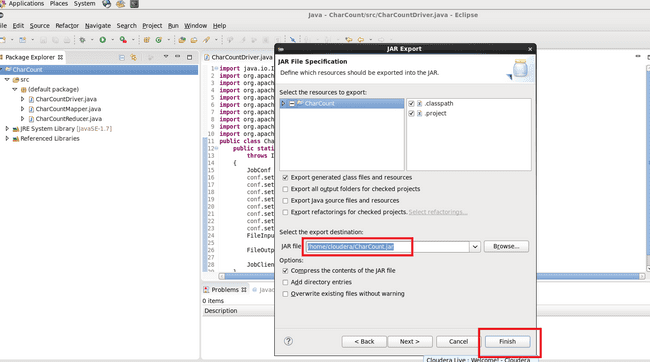
Step 6 : Write your code in java windows.

Step 7 : Right Click on the project name

CharCount -> Export -> Java -> JAR File -> Next ->

For select the export destination for JAR file:

browse -> Name : CharCount.jar -> save in folder -> cloudera -> Finish -> OK



Step 8 : Now open terminal . follow this commands :

[cloudera@quickstart ~]$ ls

abc.txt CharCount cloudera-manager Desktop Downloads

enterprise-deployment.json kerberos Music Pictures Templates

WordCloud.jar

abc.txt~ CharCount.jar cm\_api.py Documents eclipse

express-deployment.json lib parcels Public Videos

workspace

[cloudera@quickstart ~]$ pwd

/home/cloudera

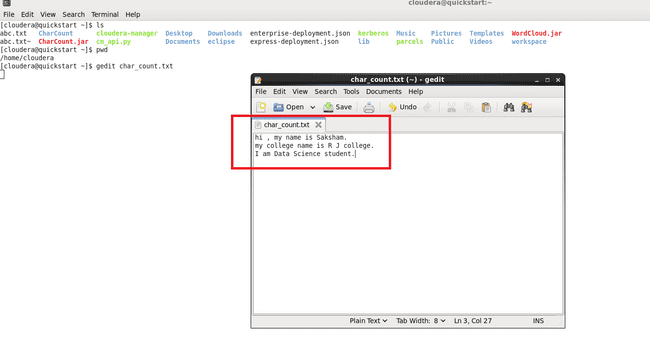
[cloudera@quickstart ~]$ gedit char\_count.txt

[cloudera@quickstart ~]$ cat char\_count.txt

hi , my name is Saksham.

my college name is R J college.

I am Data Science student.



[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 8 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:05 /inputdir

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:14 /outputdir

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -mkdir /char\_inputdir

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 9 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - cloudera supergroup 0 2023-01-24 07:32 /char\_inputdir

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:05 /inputdir

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:14 /outputdir

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -put /home/cloudera/char\_count.txt /char\_inputdir/

[cloudera@quickstart ~]$ hdfs dfs -ls /char\_inputdir

Found 1 items

-rw-r--r-- 1 cloudera supergroup 84 2023-01-24 07:34 /char\_inputdir/char\_count.txt

[cloudera@quickstart ~]$ hdfs dfs -cat /char\_inputdir/char\_count.txt

hi , my name is Saksham.

my college name is R J college.

I am Data Science student.

[cloudera@quickstart ~]$ hadoop jar /home/cloudera/CharCount.jar CharCountDriver /char\_inputdir/char\_count.txt /char\_outputdir

*-----It will start executing the program.---------*

*File System Counters*

*FILE: Number of bytes read=292*

*FILE: Number of bytes written=431746*

*FILE: Number of read operations=0*

*FILE: Number of large read operations=0*

*FILE: Number of write operations=0*

*HDFS: Number of bytes read=350*

*HDFS: Number of bytes written=100*

*HDFS: Number of read operations=9*

*HDFS: Number of large read operations=0*

*HDFS: Number of write operations=2*

*Job Counters*

*Launched map tasks=2*

*Launched reduce tasks=1*

*Data-local map tasks=2*

*Total time spent by all maps in occupied slots (ms)=38087*

*Total time spent by all reduces in occupied slots (ms)=7773*

*Total time spent by all map tasks (ms)=38087*

*Total time spent by all reduce tasks (ms)=7773*

*Total vcore-milliseconds taken by all map tasks=38087*

*Total vcore-milliseconds taken by all reduce tasks=7773*

*Total megabyte-milliseconds taken by all map tasks=39001088*

*Total megabyte-milliseconds taken by all reduce tasks=7959552*

*Map-Reduce Framework*

*Map input records=3*

*Map output records=84*

*Map output bytes=501*

*Map output materialized bytes=298*

*Input split bytes=224*

*Combine input records=84*

*Combine output records=36*

*Reduce input groups=25*

*Reduce shuffle bytes=298*

*Reduce input records=36*

*Reduce output records=25*

*Spilled Records=72*

*Shuffled Maps =2*

*Failed Shuffles=0*

*Merged Map outputs=2*

*GC time elapsed (ms)=1921*

*CPU time spent (ms)=4060*

*Physical memory (bytes) snapshot=911654912*

*Virtual memory (bytes) snapshot=4693946368*

*Total committed heap usage (bytes)=704643072*

*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=126*

*File Output Format Counters*

*Bytes Written=100*

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 10 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 09:15 /benchmarks

drwxr-xr-x - cloudera supergroup 0 2023-01-24 07:34 /char\_inputdir

drwxr-xr-x - cloudera supergroup 0 2023-01-24 07:45 /char\_outputdir

drwxr-xr-x - hbase supergroup 0 2023-01-19 05:20 /hbase

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:05 /inputdir

drwxr-xr-x - cloudera supergroup 0 2023-01-19 06:14 /outputdir

drwxr-xr-x - solr solr 0 2017-10-23 09:18 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-19 05:20 /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 ~]$ hdfs dfs -ls /char\_outputdir

Found 2 items

-rw-r--r-- 1 cloudera supergroup 0 2023-01-24 07:45 /char\_outputdir/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 100 2023-01-24 07:45 /char\_outputdir/part-00000

[cloudera@quickstart ~]$ hdfs dfs -cat /char\_outputdir/part-00000

**Output :-**

3

15

, 1

. 3

D 1

I 1

J 1

R 1

S 2

a 7

c 4

d 1

e 9

g 2

h 2

i 4

k 1

l 4

m 6

n 4

o 2

s 4

t 3

u 1

y 2

[cloudera@quickstart ~]$

**Practical : 04**

**Aim :** Whether Analysis Application using Python and MapReduce

**Aim : Weather Analysis using MapReduce.**

**Step 1 : Open terminal > Add the weather.txt file and create mapper.py and reducer.py.**

**[cloudera@quickstart Desktop]$ mkdir weather**

**[cloudera@quickstart Desktop]$ cd weather**

**[cloudera@quickstart weather]$ touch weather.txt**

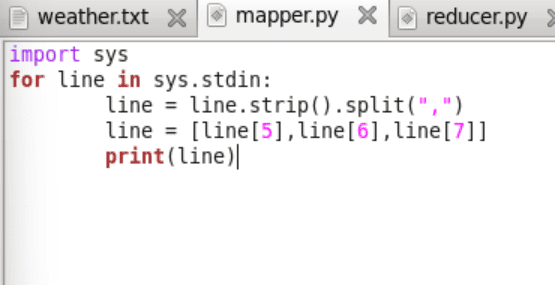
**[cloudera@quickstart weather]$ ls**

**weather.txt**

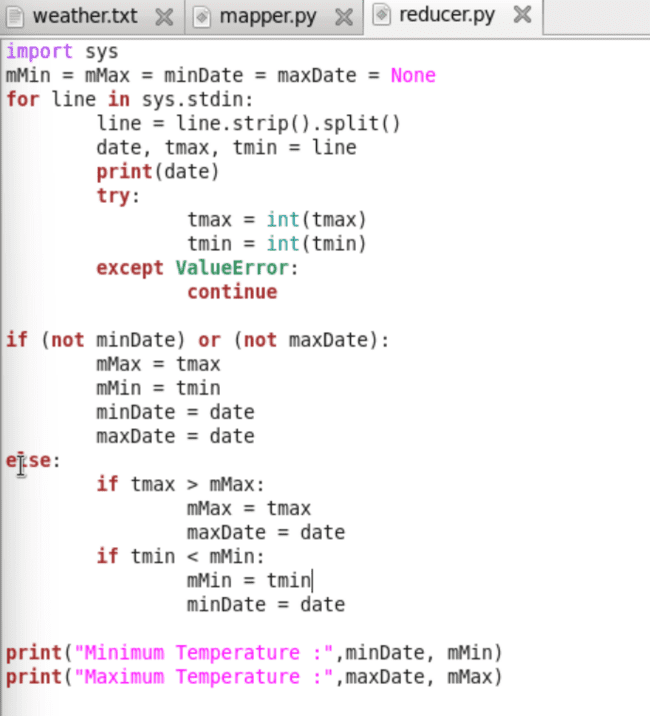
**[cloudera@quickstart weather]$ touch mapper.py**

**[cloudera@quickstart weather]$ touch reducer.py**

**Step 2 : Open mapper.py and write the code.**

****

**Step 3 : Open reducer.py and write the code.**

****

**Step 4 : Running the mapper function with weather.txt file to check the output.**

**[cloudera@quickstart weather]$ cat weather.txt | python mapper.py**

**Output:**

**20100101 -178 -311**

**20100102 -244 -322**

**20100103 -194 -289**

**20100104 -167 -200**

**20100105 -133 -167**

**20100106 -133 -172**

**20100107 -150 -278**

**20100108 -233 -328**

**20100109 -233 -322**

**20100110 -117 -244**

**20100111 -67 -128**

**20100112 -78 -122**

**20100113 -17 -89**

**20100114 39 -72**

**20100115 -67 -72**

**20100116 22 -50**

**20100117 33 -44**

**20100118 6 -172**

**20100119 -56 -183**

**20100120 -67 -139**

**20100121 -67 -94**

**20100122 -44 -67**

**20100123 -6 -44**

**20100124 0 -11**

**20100125 -11 -161**

**20100126 -161 -233**

**20100127 -167 -222**

**20100128 -167 -283**

**20100129 -189 -283**

**20100130 -156 -267**

**20100131 -150 -272**

**Step 5: The mapper function will put these values in the stream which will be used by the reducer function and give the output**

**[cloudera@quickstart weather]$ cat weather.txt | python mapper.py |python reducer.py**

**Output:**

**['20100101', '-178', '-311']**

**20100101**

**('Minimum Temperature:', '20100101', -311)**

**('Maximum Temperature:', '20100101', -178)**

**['20100102', '-244', '-322']**

**20100102**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100101', -178)**

**['20100103', '-194', '-289']**

**20100103**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100101', -178)**

**['20100104', '-167', '-200']**

**20100104**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100104', -167)**

**['20100105', '-133', '-167']**

**20100105**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100105', -133)**

**['20100106', '-133', '-172']**

**20100106**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100105', -133)**

**['20100107', '-150', '-278']**

**20100107**

**('Minimum Temperature:', '20100102', -322)**

**('Maximum Temperature:', '20100105', -133)**

**['20100108', '-233', '-328']**

**20100108**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100105', -133)**

**['20100109', '-233', '-322']**

**20100109**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100105', -133)**

**['20100110', '-117', '-244']**

**20100110**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100110', -117)**

**['20100111', '-67', '-128']**

**20100111**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100111', -67)**

**['20100112', '-78', '-122']**

**20100112**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100111', -67)**

**['20100113', '-17', '-89']**

**20100113**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100113', -17)**

**['20100114', '39', '-72']**

**20100114**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100115', '-67', '-72']**

**20100115**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100116', '22', '-50']**

**20100116**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100117', '33', '-44']**

**20100117**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100118', '6', '-172']**

**20100118**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100119', '-56', '-183']**

**20100119**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100120', '-67', '-139']**

**20100120**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100121', '-67', '-94']**

**20100121**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100122', '-44', '-67']**

**20100122**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100123', '-6', '-44']**

**20100123**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100124', '0', '-11']**

**20100124**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100125', '-11', '-161']**

**20100125**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100126', '-161', '-233']**

**20100126**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100127', '-167', '-222']**

**20100127**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100128', '-167', '-283']**

**20100128**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100129', '-189', '-283']**

**20100129**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100130', '-156', '-267']**

**20100130**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**['20100131', '-150', '-272']**

**20100131**

**('Minimum Temperature:', '20100108', -328)**

**('Maximum Temperature:', '20100114', 39)**

**[cloudera@quickstart weather]$ ^C**

**[cloudera@quickstart weather]$**

**Practical : 05**

**Aim :**Matrix Multiplication Application using Python and MapReduce

Step 1 : Open terminal.

Create two text files of matrix m1.txt and m2.txt.

[cloudera@quickstart ~]$ cd ~/Desktop

[cloudera@quickstart Desktop]$ mkdir matrix\_mul

[cloudera@quickstart Desktop]$ cd matrix\_mul

[cloudera@quickstart matrix\_mul]$ gedit m1.txt

m1

1 2 3

4 5 6

[cloudera@quickstart matrix\_mul]$ gedit m2.txt

m2

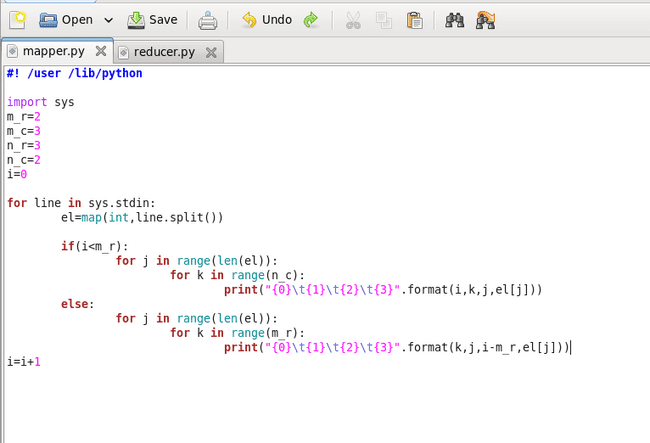
7 8

9 10

11 12

Step 2 : Create Mapper file mapper.py

[cloudera@quickstart matrix\_mul]$ gedit mapper.py



[cloudera@quickstart matrix\_mul]$ cat m1.txt m2.txt | python mapper.py

Mapper Output :

0 0 0 1

0 1 0 1

0 0 1 2

0 1 1 2

0 0 2 3

0 1 2 3

0 0 0 4

0 1 0 4

0 0 1 5

0 1 1 5

0 0 2 6

0 1 2 6

0 0 0 7

0 1 0 7

0 0 1 8

0 1 1 8

0 0 0 9

0 1 0 9

0 0 1 10

0 1 1 10

0 0 0 11

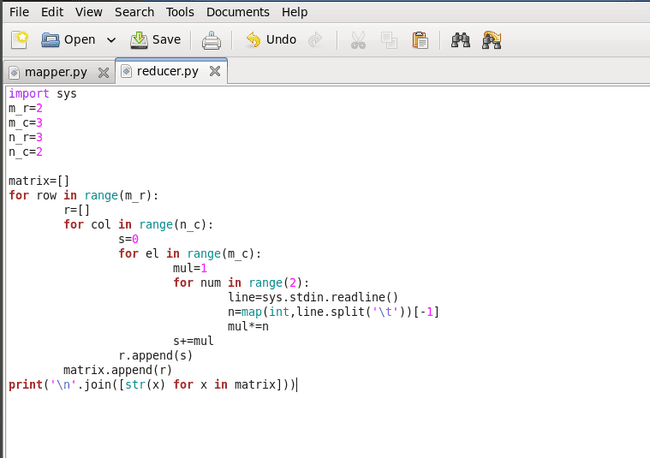
0 1 0 11

0 0 1 12

0 1 1 12

Step 3 : Create Reducer file reducer.py

[cloudera@quickstart matrix\_mul]$ gedit reducer.py



Step 4 : to get matrix multiplication :

[cloudera@quickstart matrix\_mul]$ cat m1.txt m2.txt | python mapper.py | python reducer.py

**Output :**

[14, 77]

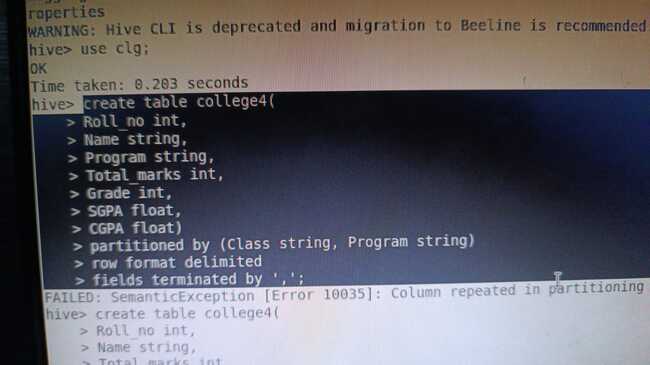
[194, 365]

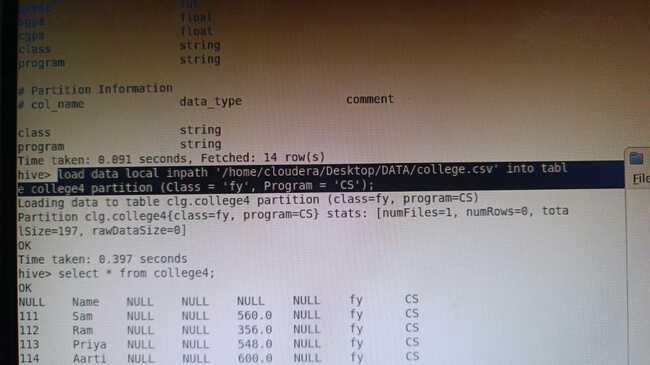
**Practical : 06**

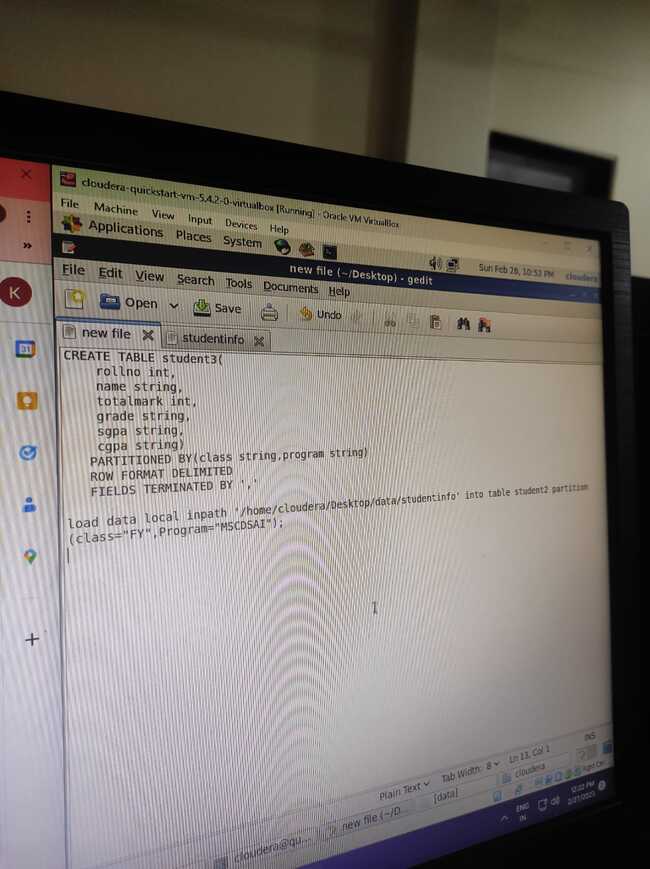
**Aim :** Using Hive Tool

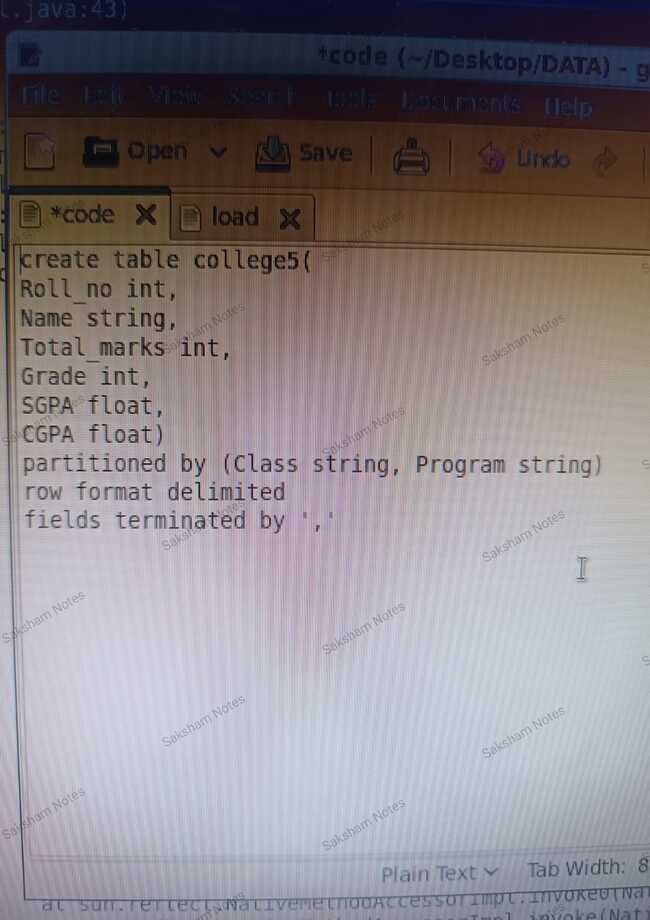
Hive is a data warehouse system that is used to query and analyse large datasets stored in the HDFS. Hive uses a query language called HiveQL, which is similar to SQL.

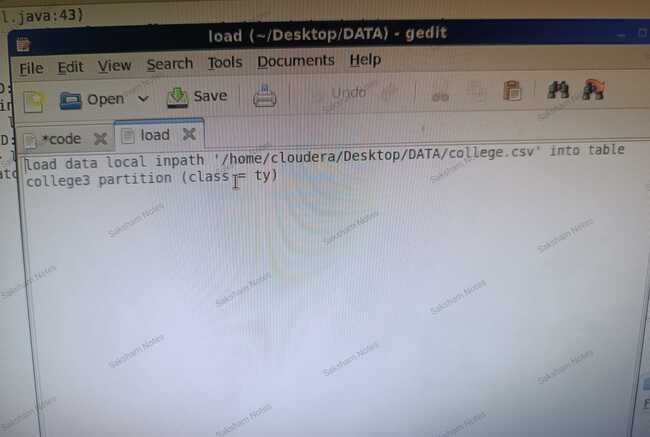
Follow Below steps :

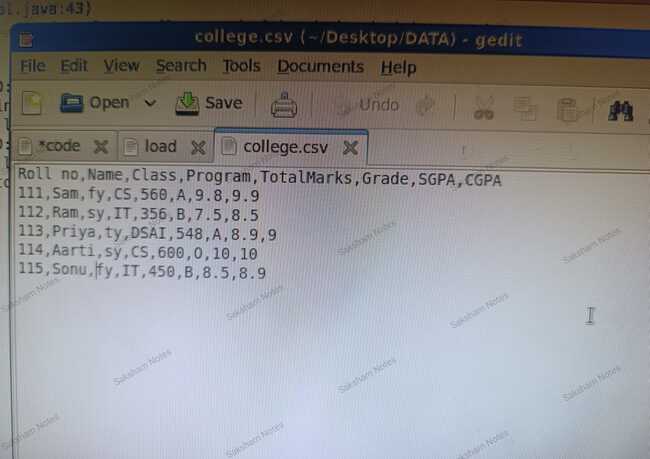


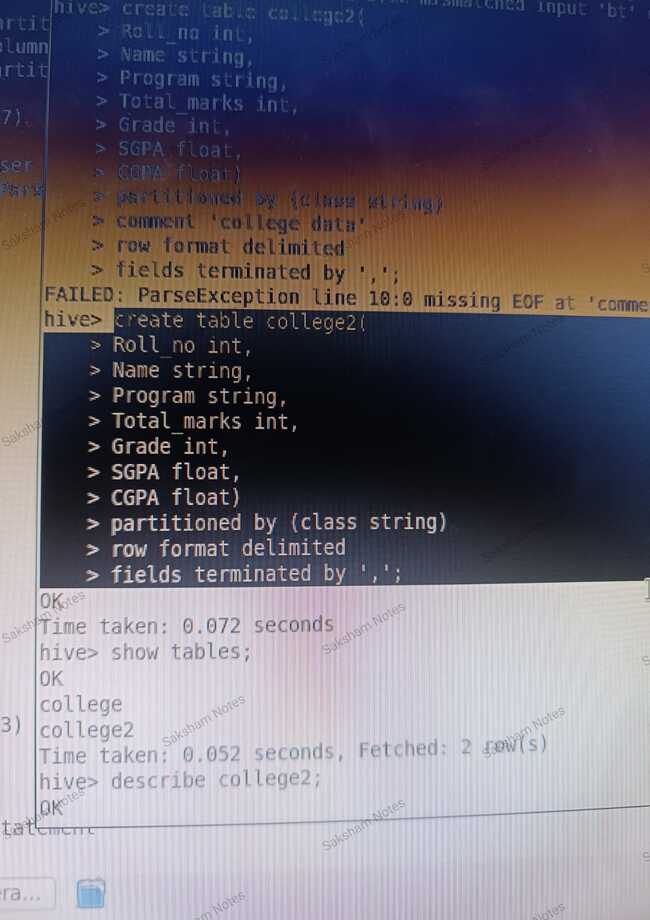


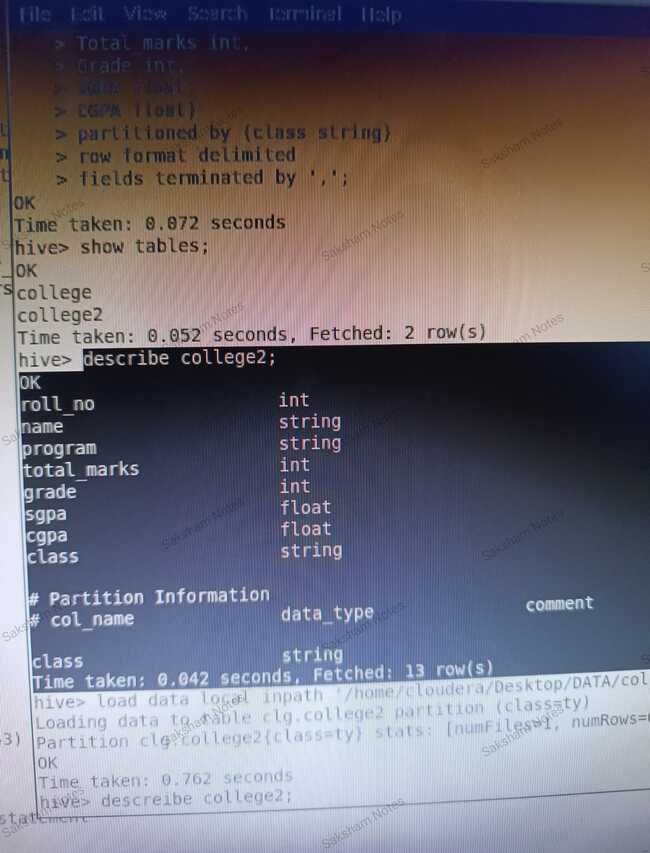


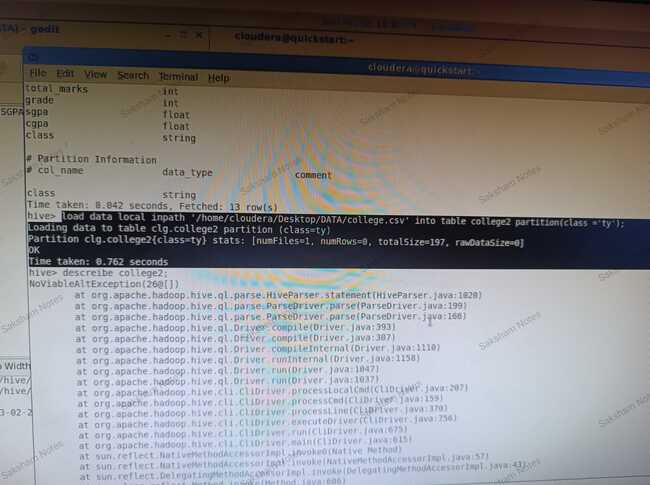


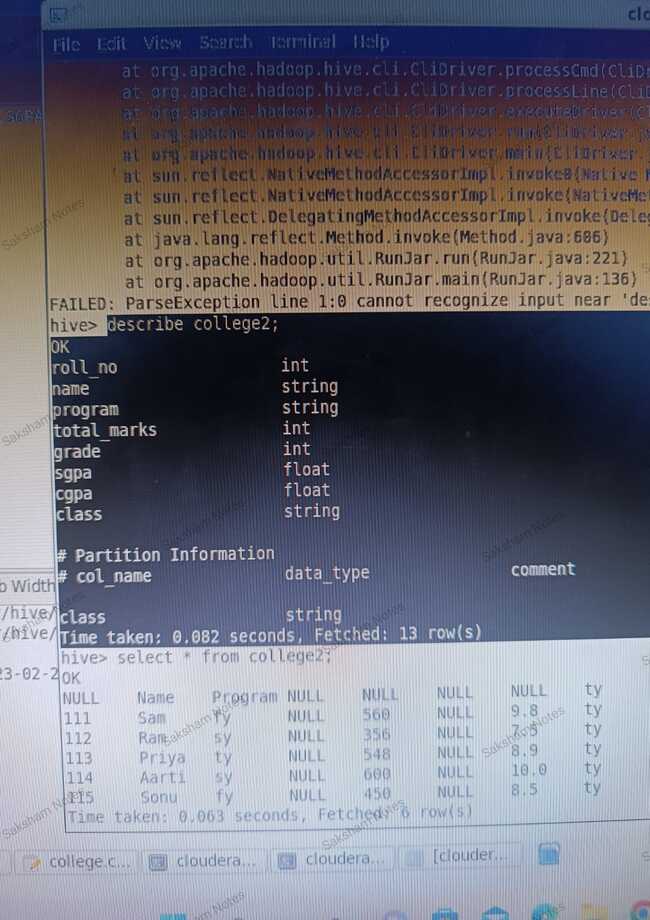


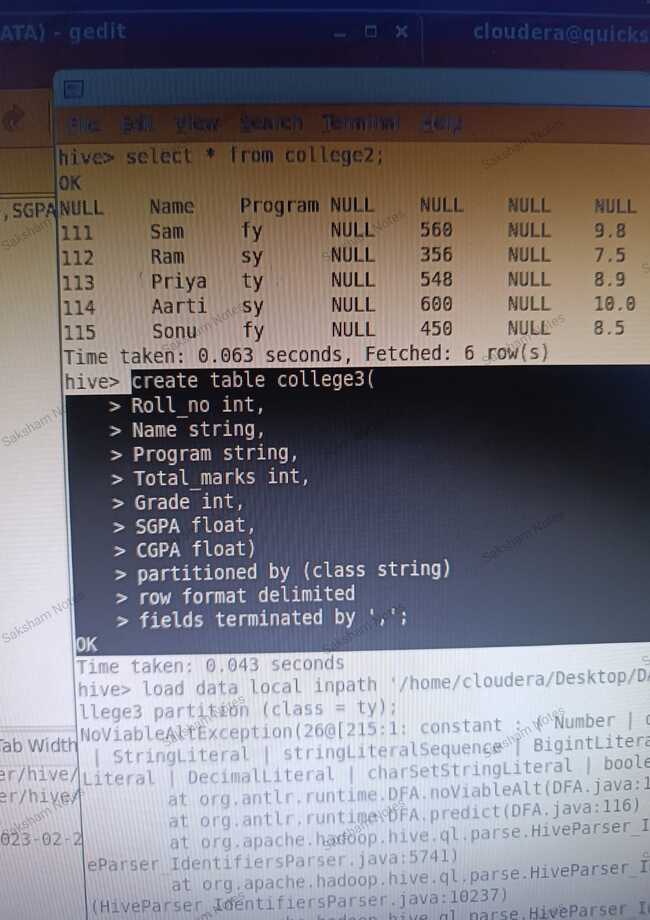


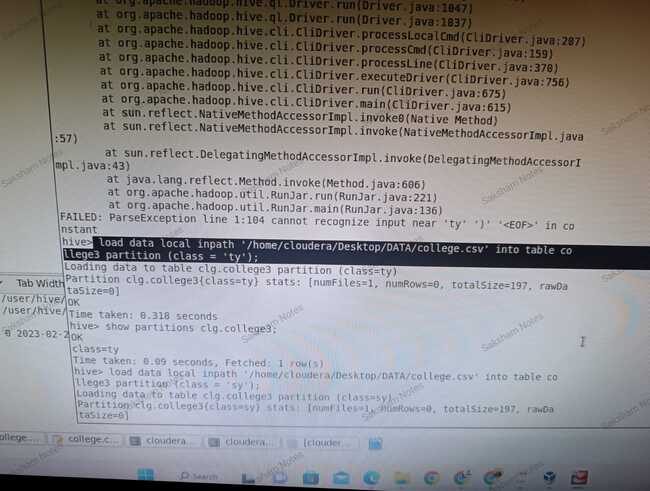


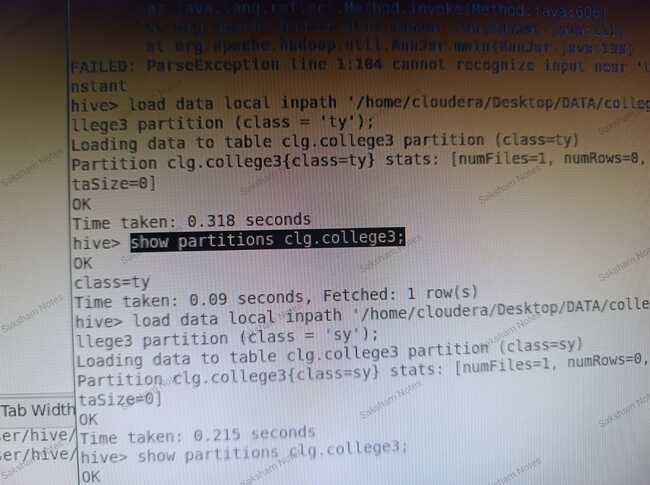


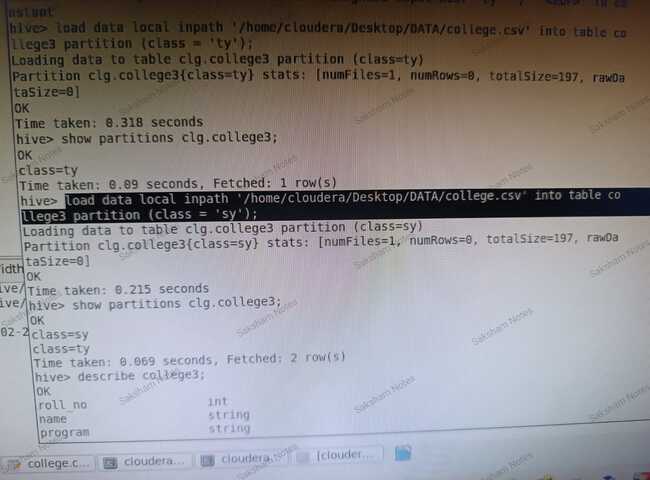


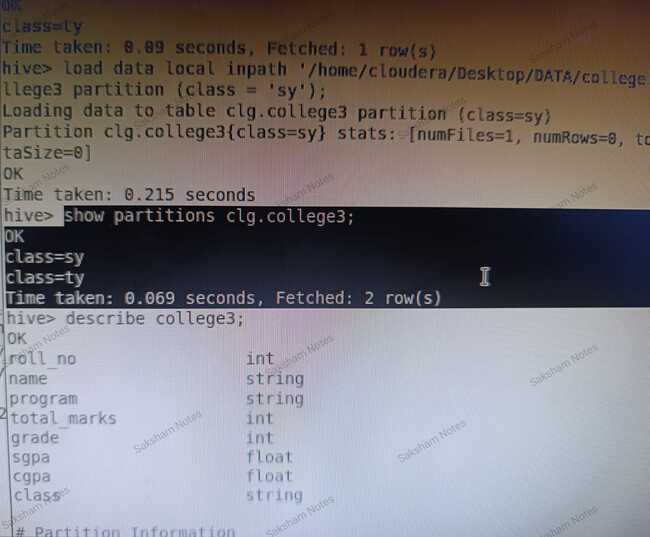


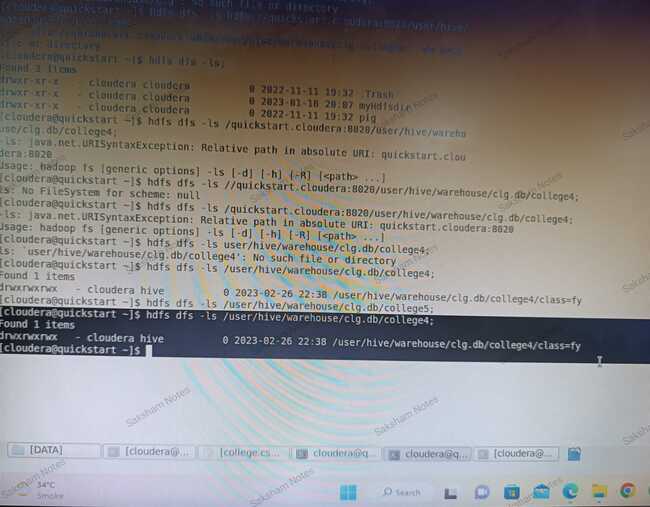


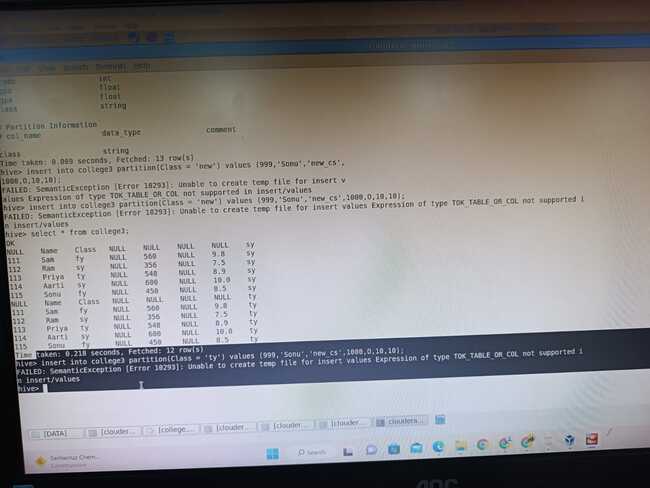












**Done**

**Practical : 07**

**Aim :** Using Pig Tool and creating Pig Latin Scripts

Pig is a high-level platform for creating MapReduce programs used in Hadoop. It provides a simple language called Pig Latin for expressing data analysis tasks. Pig Latin abstracts away the complexity of MapReduce programming and allows users to focus on the business logic of their data processing tasks.

To create Pig Latin scripts, you need to follow these steps:

1. Write a Pig Latin script: The Pig Latin script contains the logic for data processing. It is a high-level language that makes it easy to express complex data processing tasks. Pig Latin scripts are written in a text editor or an integrated development environment (IDE) and saved with a .pig file extension.
2. Load data: Before you can process data, you need to load it into Pig. You can load data from a local file system, HDFS, or other data sources. Pig supports several file formats such as CSV, JSON, Avro, and more.
3. Transform data: After loading the data, you can transform it using Pig Latin operators. Pig Latin supports many operators for filtering, grouping, sorting, and aggregating data. You can chain multiple operators together to create complex data transformations.
4. Store the results: Once you have processed the data, you can store the results back to HDFS or other data sources. Pig supports several storage formats such as Avro, SequenceFile, and more.

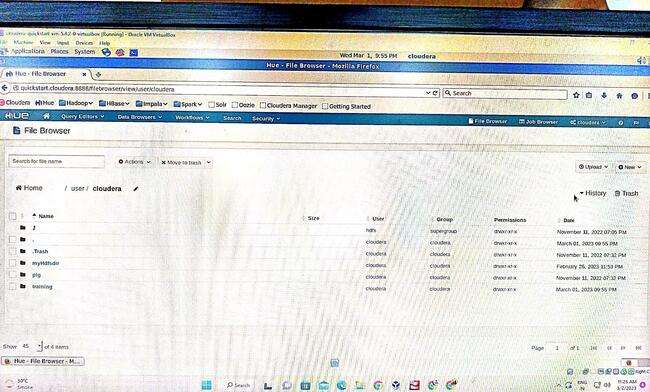
Some examples of Pig Latin operators are:

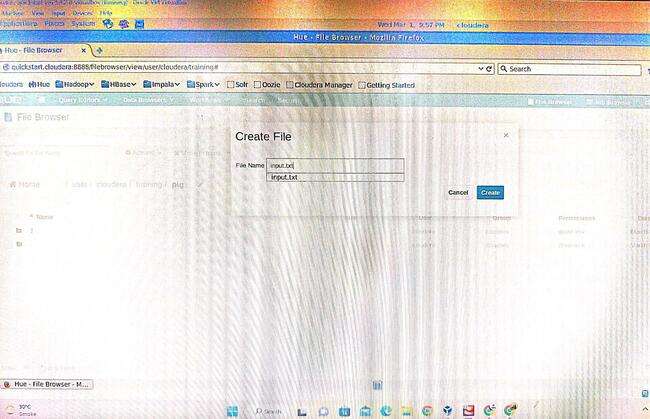
* LOAD: Used to load data from a file or data source.
* FILTER: Used to filter data based on a condition.
* GROUP: Used to group data by one or more fields.
* FOREACH: Used to apply a transformation to each record.
* DISTINCT: Used to remove duplicate records.
* JOIN: Used to join two or more relations based on a common field.

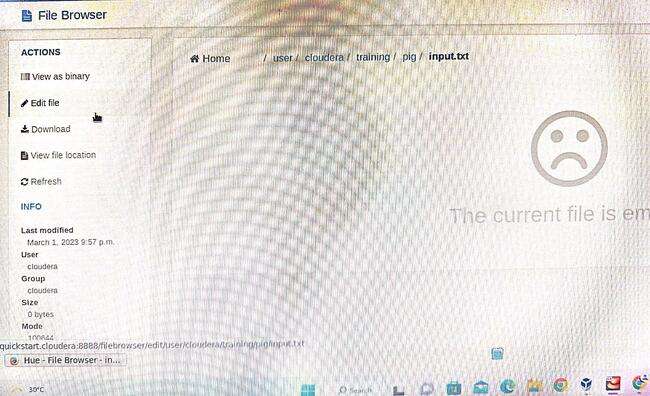
Pig also provides a tool called Pig Latin Grunt shell, which is a command-line interface for running Pig scripts. You can use the Grunt shell to run Pig scripts and interact with Pig Latin operators in real-time.

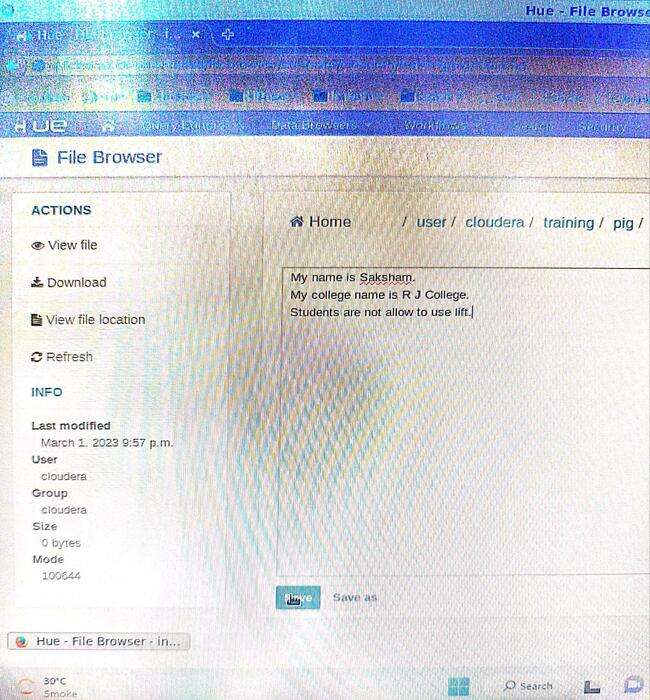
Overall, Pig is a powerful tool for data processing in Hadoop, and Pig Latin makes it easy to express complex data transformations.

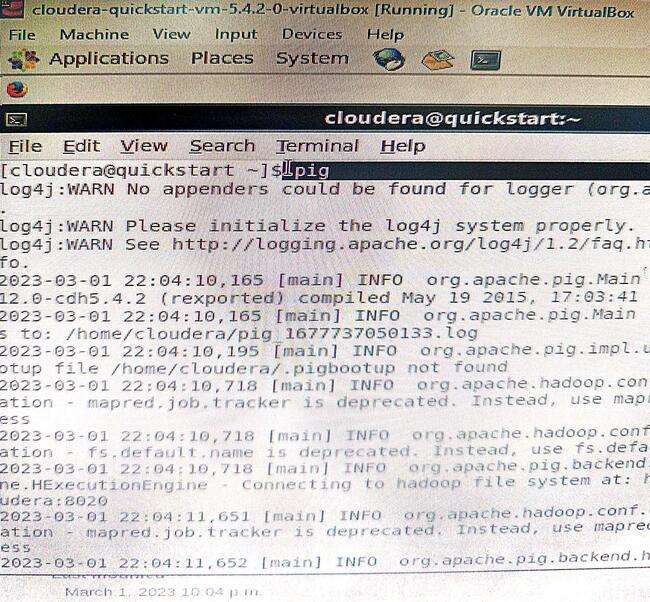
**Follow Below Steps :**

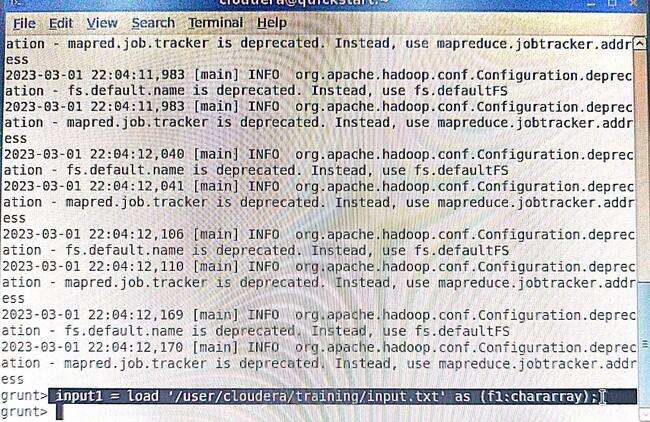
****

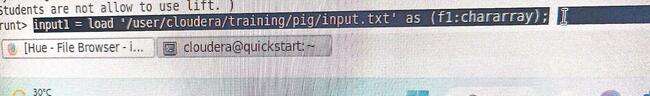
****

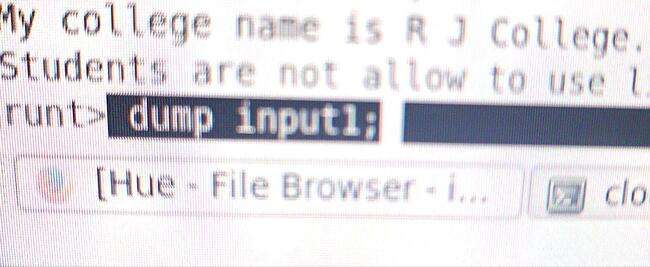
****

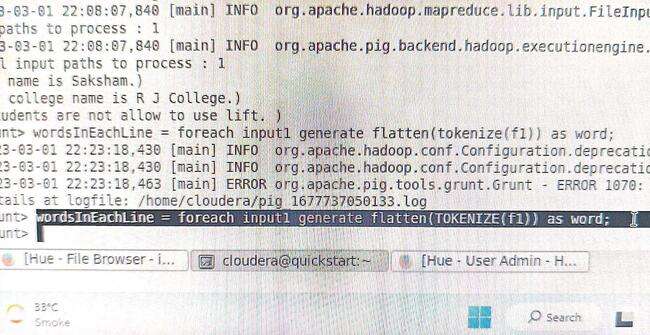
****

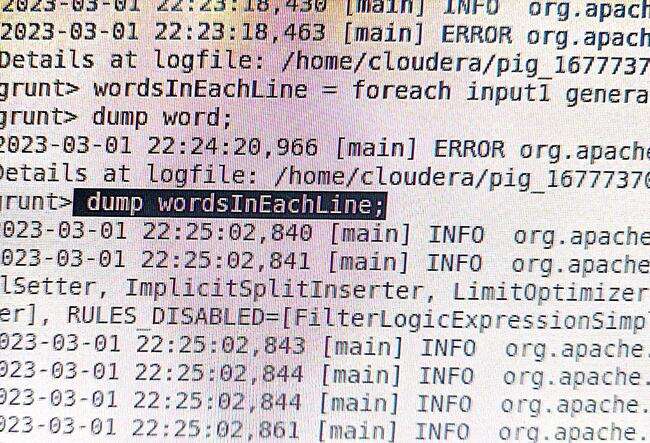
****

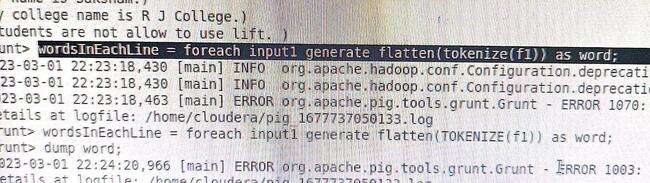
****

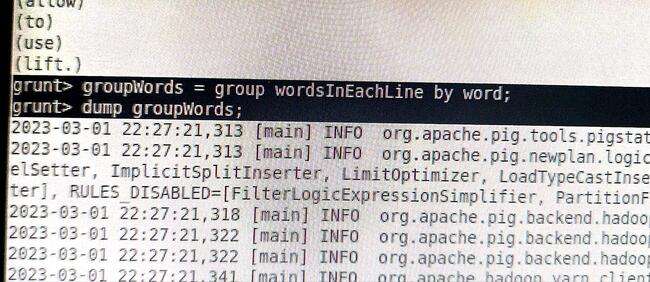
****

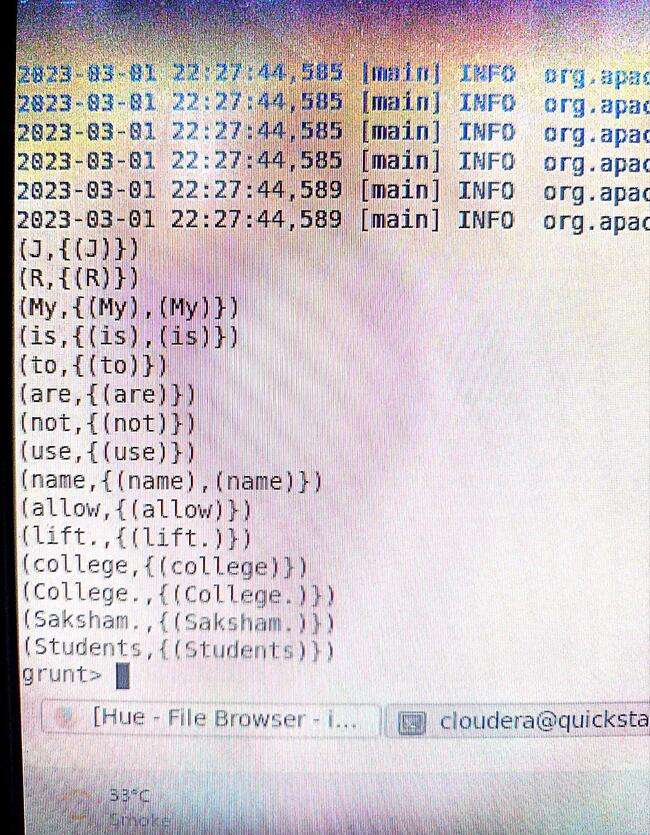
****

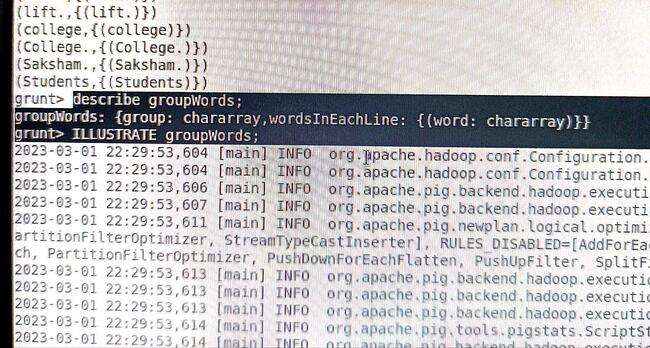
****

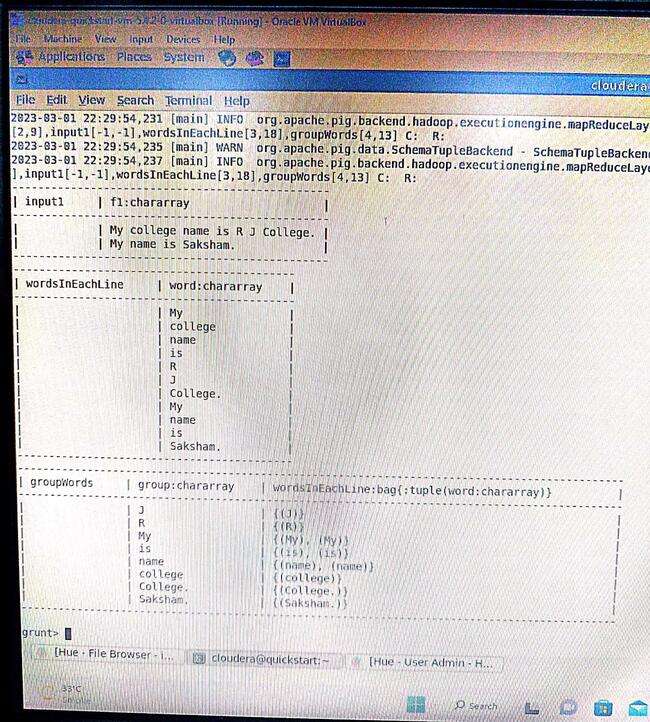
****

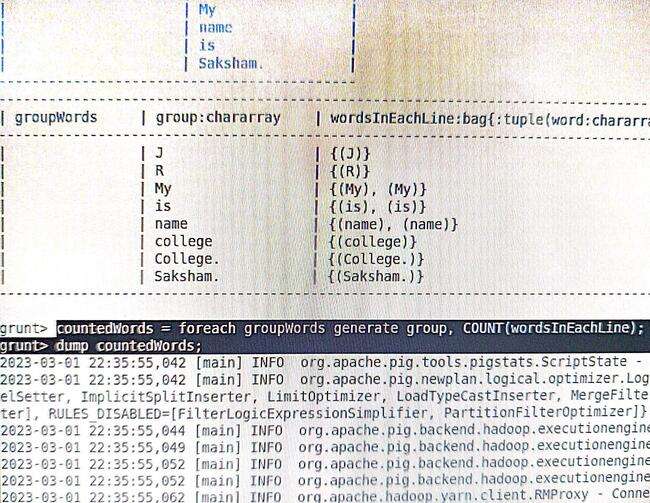
****

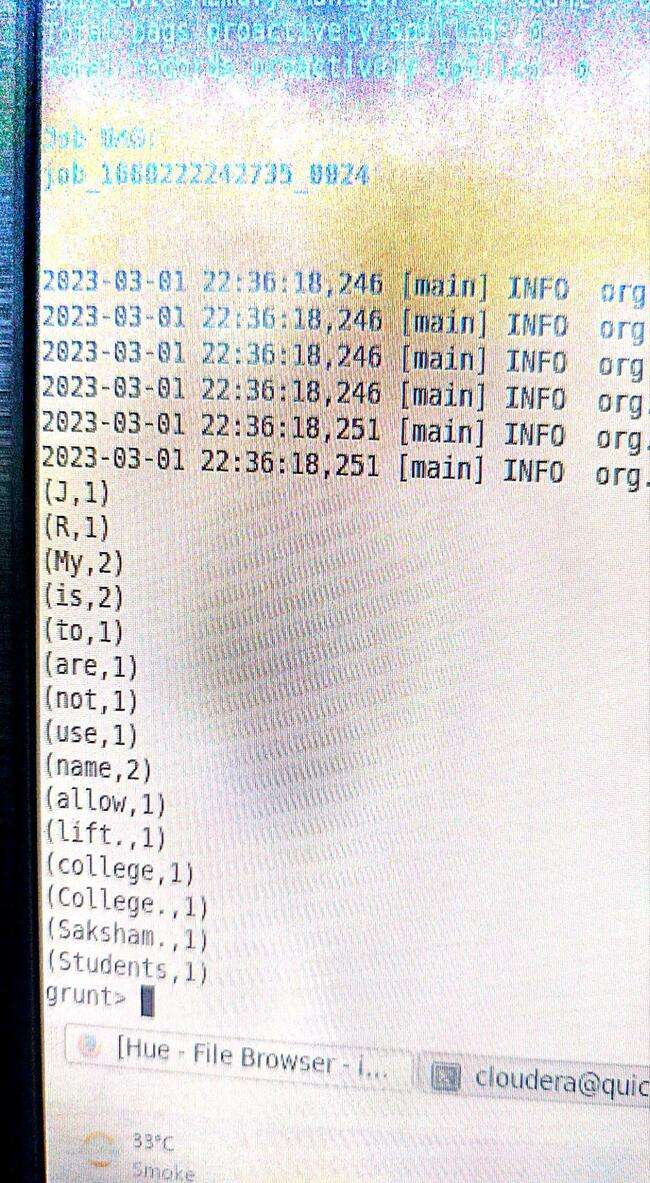
****

****

****

****

****

****

**Practical : 08**

**Aim :** Using HBase Tool

**HBase is a distributed, column-oriented database built on top of the Hadoop Distributed File System (HDFS). It is a NoSQL database that provides high scalability, fault-tolerance, and real-time data access. HBase is used to store and manage large-scale, semi-structured data.**

**HBase provides a command-line tool called the HBase shell, which is used to interact with the HBase database. The HBase shell provides a simple and interactive way to create, read, update, and delete data in HBase.**

**Some of the commonly used HBase shell commands are:**

**create:** Used to create a new table in HBase.

Syntax: create 'table\_name', 'column\_family'

**put:** Used to insert data into an existing table in HBase.

Syntax: put 'table\_name', 'row\_key', 'column\_family:column\_qualifier', 'value'

**get:** Used to retrieve data from an existing table in HBase.

Syntax: get 'table\_name', 'row\_key'

**scan:** Used to retrieve all data from an existing table in HBase.

Syntax: scan 'table\_name'

**delete:** Used to delete data from an existing table in HBase.

Syntax: delete 'table\_name', 'row\_key', 'column\_family:column\_qualifier'

**disable:** Used to disable a table in HBase.

Syntax: disable 'table\_name'

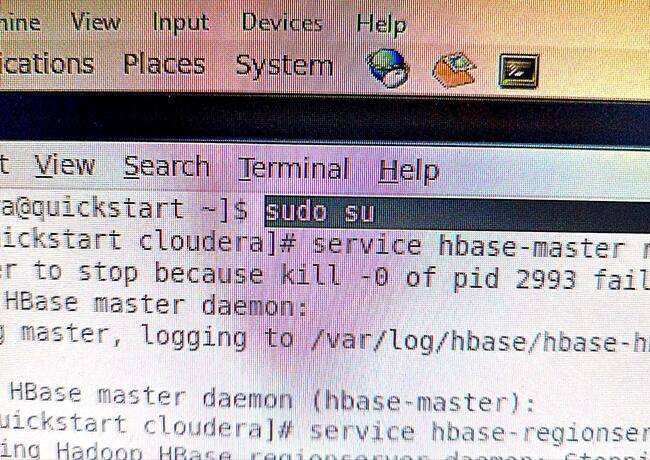
**drop:** Used to delete a table in HBase.

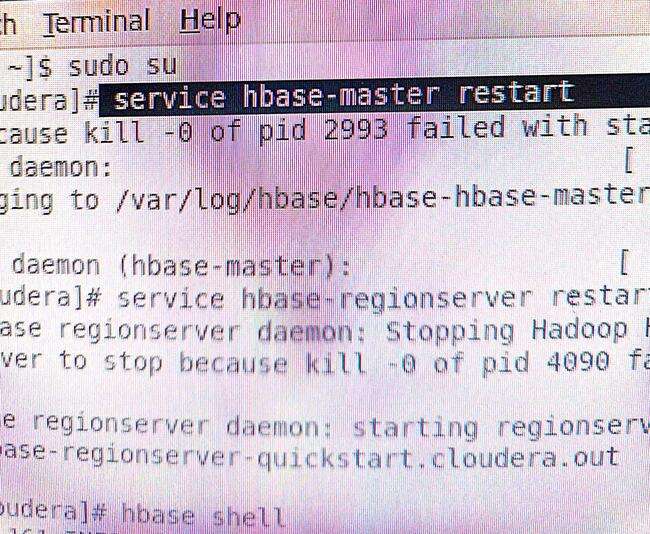
Syntax: drop 'table\_name'

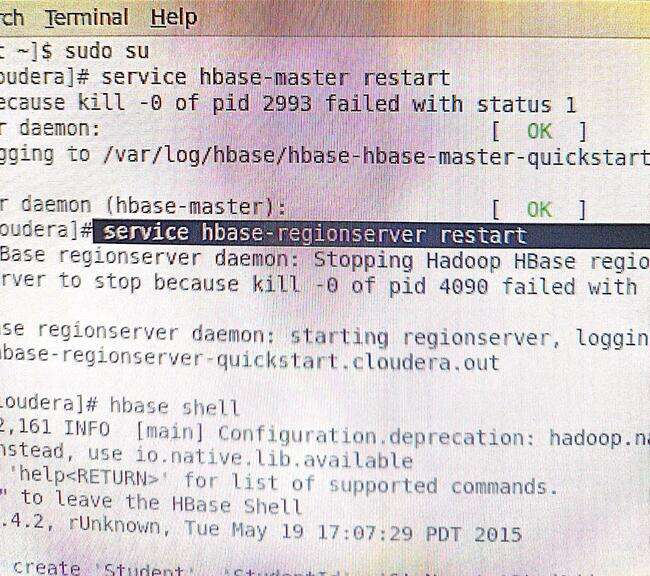
**Apart from the HBase shell, HBase also provides a web-based user interface called HBase Web UI, which can be used to monitor and manage HBase clusters. The HBase Web UI provides a graphical interface to view HBase tables, regions, and other system statistics.**

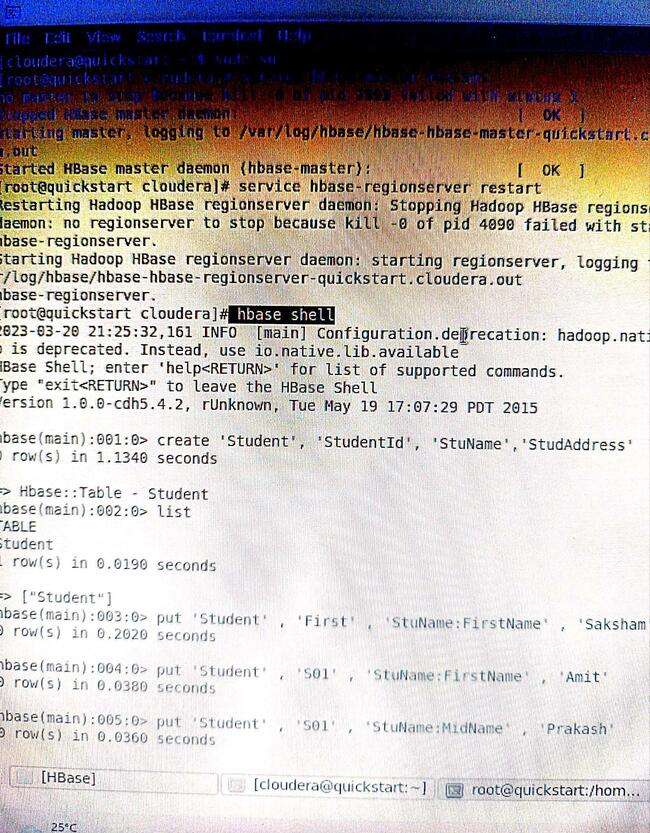
**Overall, HBase is a powerful NoSQL database that provides high scalability, fault-tolerance, and real-time data access. The HBase shell and HBase Web UI make it easy to interact with HBase and manage HBase clusters.**

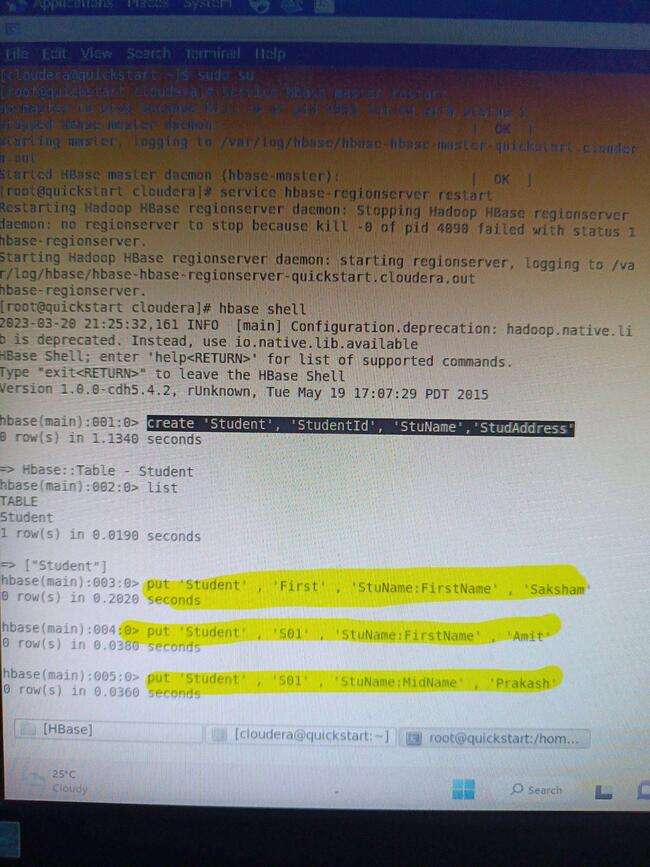
**Follow Below Steps:**

****

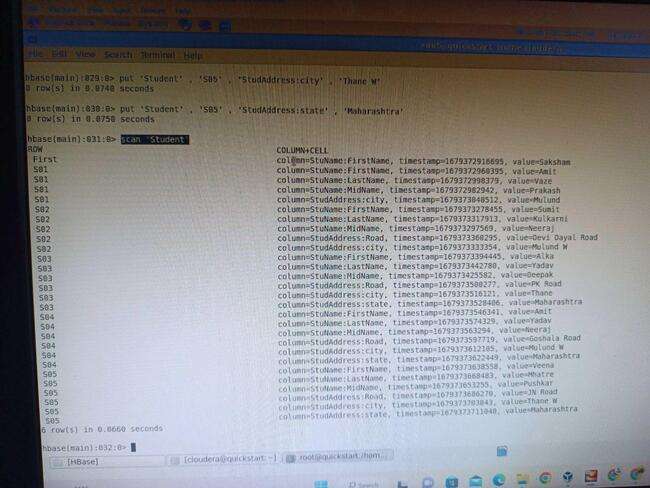
****

****

****

****

****

****

****

**Practical : 09**

**Aim :** Using Document-based Database - MongoDB

**What is MongoDB?**

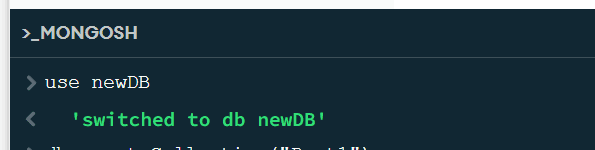
**MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++..**

**1.The use Command:**

**MongoDB use DATABASE\_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.**

**Basic syntax of use command is as follows-**

**use databasename**

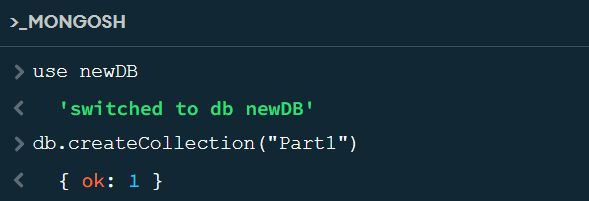
****

**2.The createCollection() Method**

**MongoDB db.createCollection(name, options) is used to create collection.**

**Basic syntax of createCollection() command is as follows −**

**db.createCollection(name, options);**

****

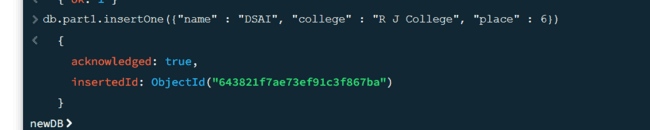
**3.The insert() Method**

**To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.**

**Syntax**

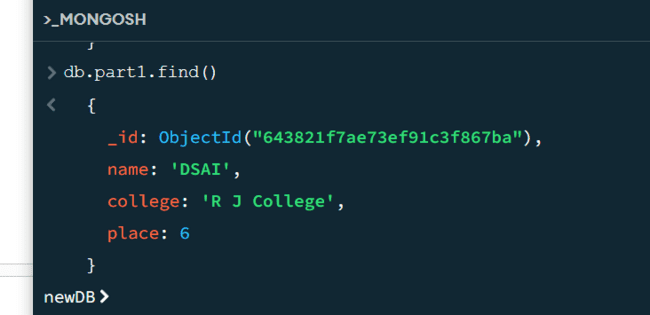
**The basic syntax of insert() command is as follows −**

**db.COLLECTION\_NAME.insert(document);**

****

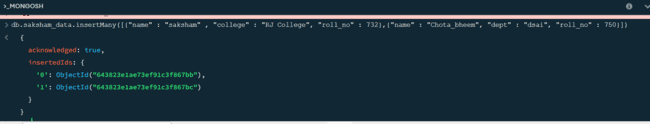
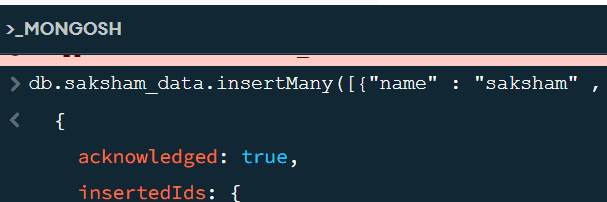
**4.The find() Method**

**MongoDB's find() method, explained in MongoDB Query Document accepts second optional parameter that is list of fields that you want to retrieve. In MongoDB, when you execute find() method, then it displays all fields of a document. To limit this, you need to set a list of fields with value 1 or 0. 1 is used to show the field while 0 is used to hide the fields.**

****

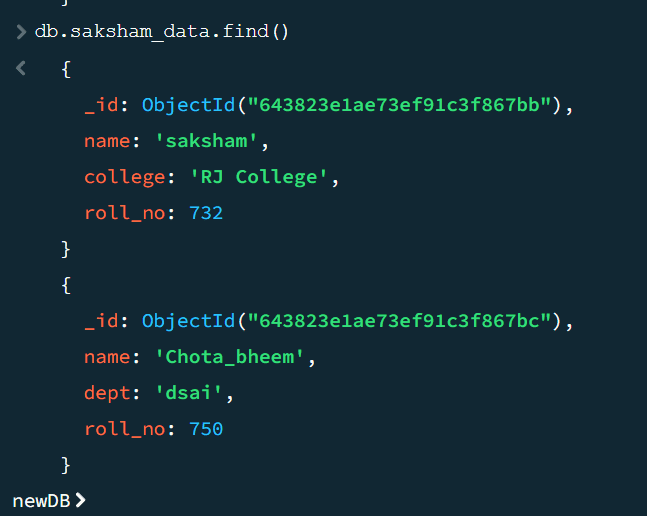
**5. inserMany() Method**

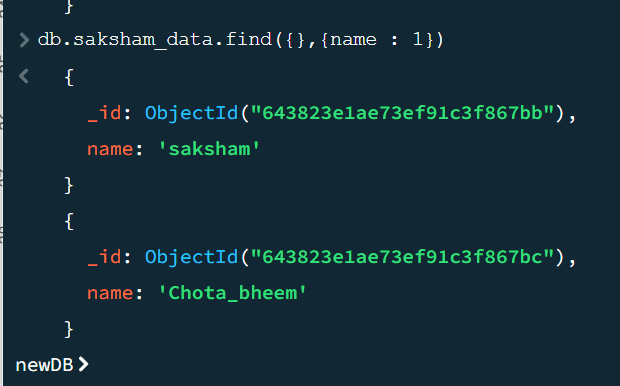
**The insertMany() method inserts one or more documents in the collection. It takes array of documents to insert in the collection.**

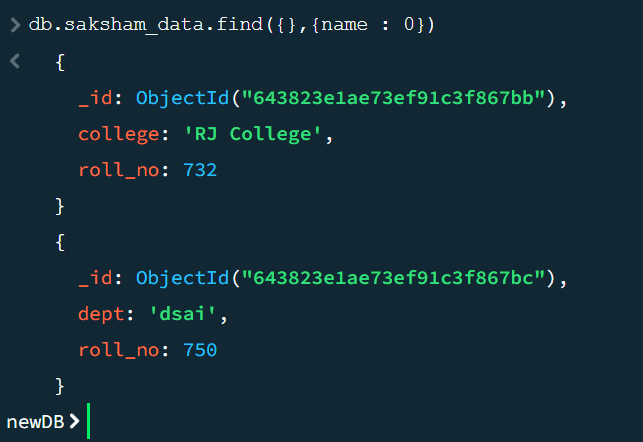
****

**6. find() method**

**In MongoDB, find() method is used to select documents in a collection and return a cursor to the selected documents. Cursor means a pointer that points to a document, when we use find() method it returns a pointer on the selected documents and returns one by one. If we want to return pointer on all documents then use empty() parameter that returns all documents one by one. It takes only some optional parameters. The first optional parameter is the selection criteria on which we want to return a cursor. To return all documents in a collection use empty document({}).**

****

****

****

**Practical : 10**

**Aim :** Using Impala Tool

Impala is an open-source, distributed SQL query engine that runs on top of the Hadoop Distributed File System (HDFS). Impala is designed for interactive SQL queries on large datasets and provides real-time querying of Hadoop data.

Impala supports a wide range of SQL commands and can be used with various Business Intelligence (BI) tools, such as Tableau and MicroStrategy. Impala provides a fast and flexible way to query data stored in Hadoop.

**Some of the key features of Impala are:**

**High Performance:** Impala uses a distributed architecture that enables it to perform SQL queries on large datasets in real-time. Impala is highly optimized for Hadoop and can execute SQL queries faster than other SQL-on-Hadoop engines.

**SQL Compatibility:** Impala supports a wide range of SQL commands and is compatible with standard SQL syntax. This makes it easy for SQL developers to use Impala without having to learn a new query language.

**Integration with Hadoop Ecosystem:** Impala is integrated with the Hadoop ecosystem and can be used with various Hadoop components such as HDFS, Hive, and HBase. This integration enables Impala to access data stored in these systems and perform queries on them.

**Security:** Impala provides a comprehensive security model that includes authentication, authorization, and audit logging. This ensures that data is protected and only authorized users can access it.

**Some of the commonly used Impala SQL commands are:**

**SELECT: Used to retrieve data from one or more tables.**

**Syntax: SELECT column1, column2 FROM table\_name**

**WHERE: Used to filter data based on a condition.**

**Syntax: SELECT column1, column2 FROM table\_name WHERE condition**

**JOIN: Used to combine data from two or more tables.**

**Syntax: SELECT column1, column2 FROM table1 JOIN table2 ON table1.column = table2.column**

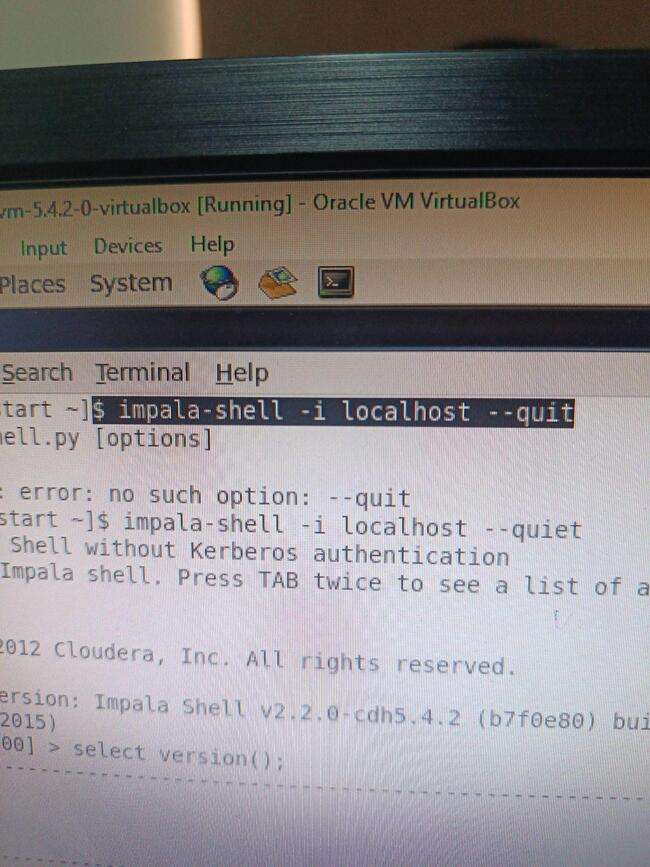
**GROUP BY: Used to group data based on one or more columns.**

**Syntax: SELECT column1, COUNT(column2) FROM table\_name GROUP BY column1**

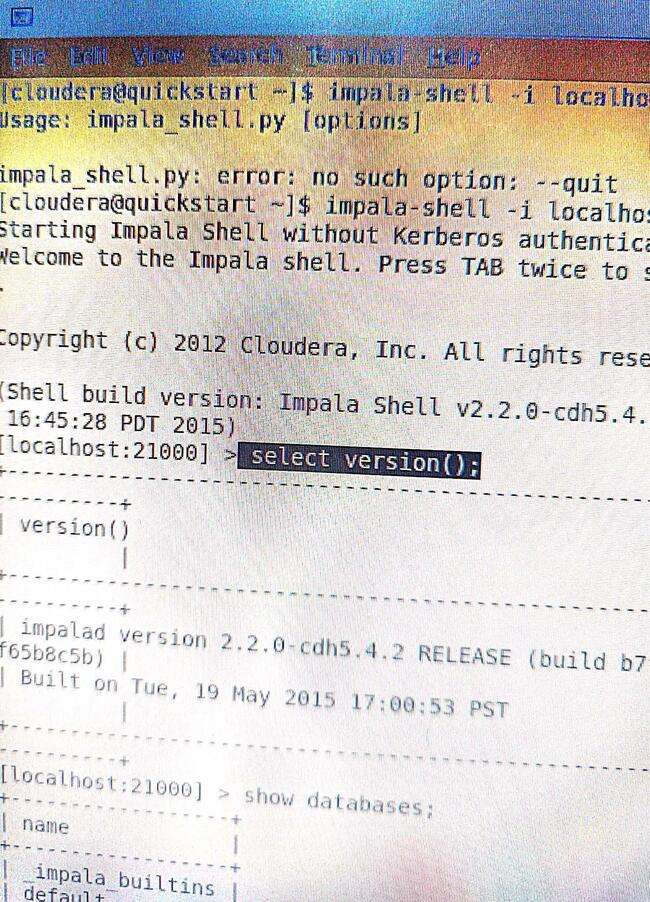
**ORDER BY: Used to sort data in ascending or descending order.**

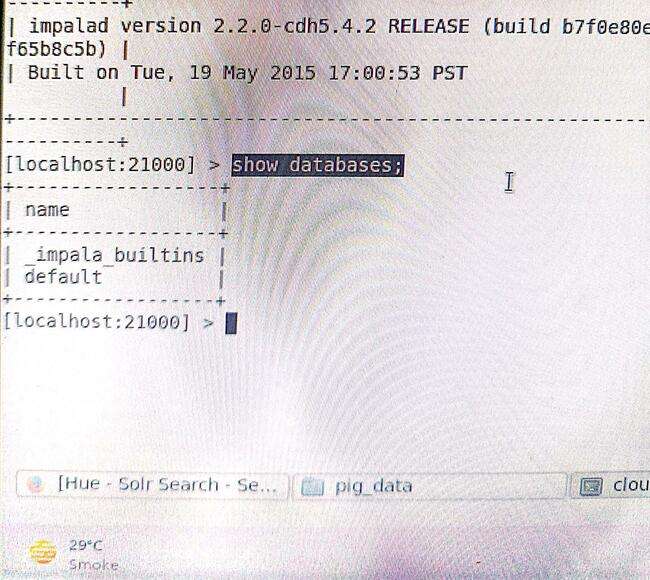
**Syntax: SELECT column1, column2 FROM table\_name ORDER BY column1 ASC/DESC**

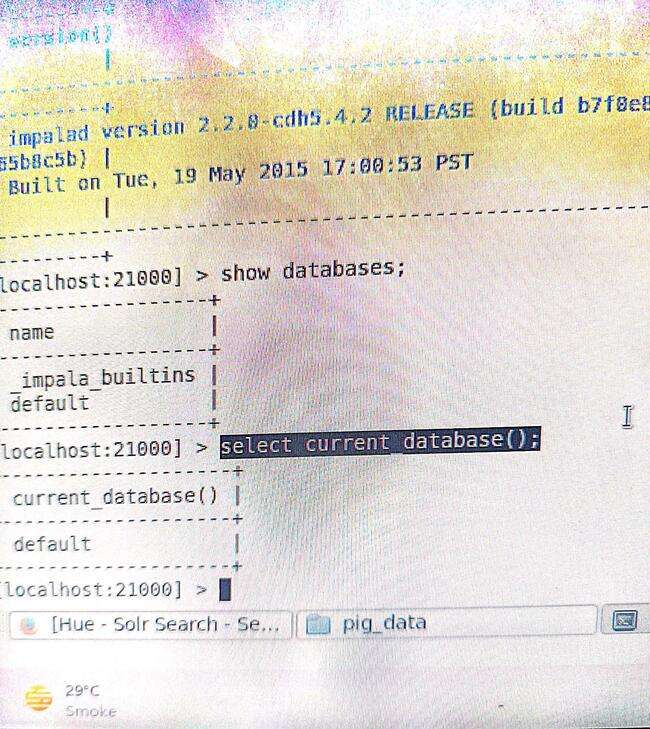
**Overall, Impala is a powerful SQL-on-Hadoop engine that provides high-performance querying of large datasets stored in Hadoop. Its SQL compatibility and integration with the Hadoop ecosystem make it a popular choice for interactive data querying and analysis.**

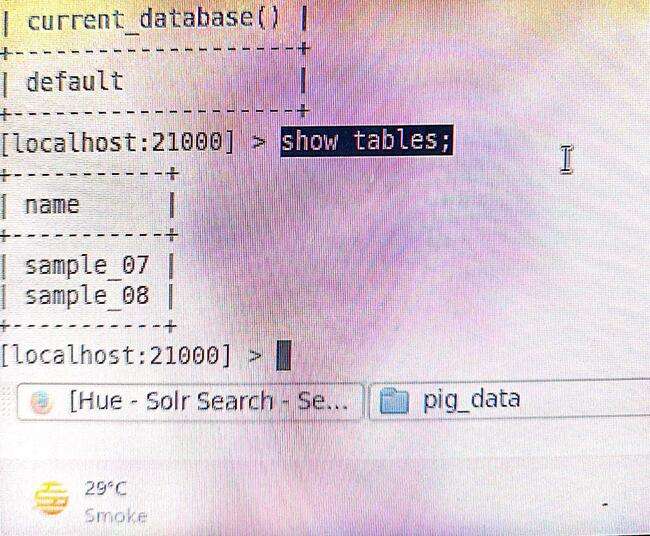
****

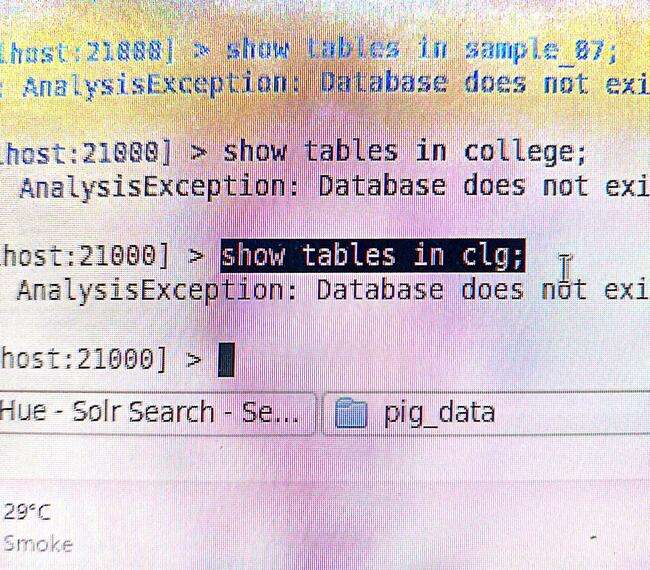
****

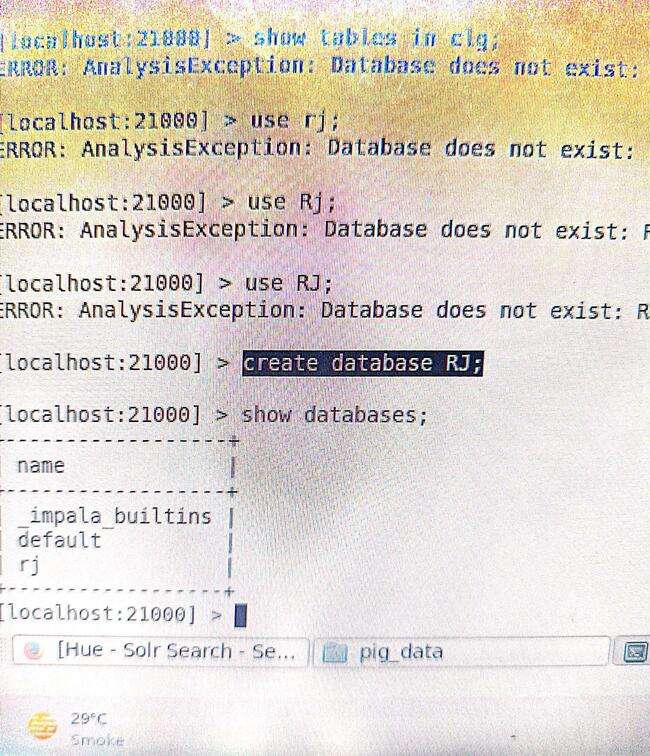
****

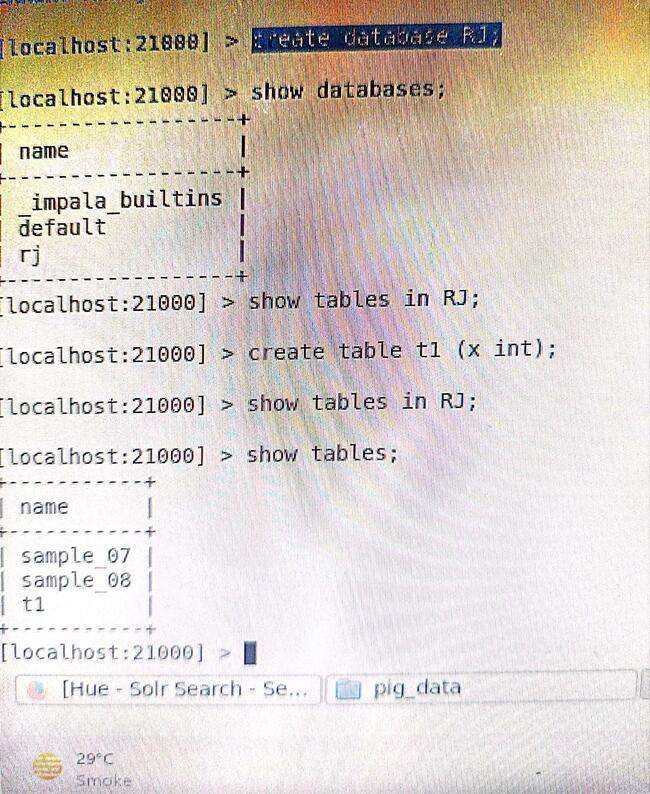
****

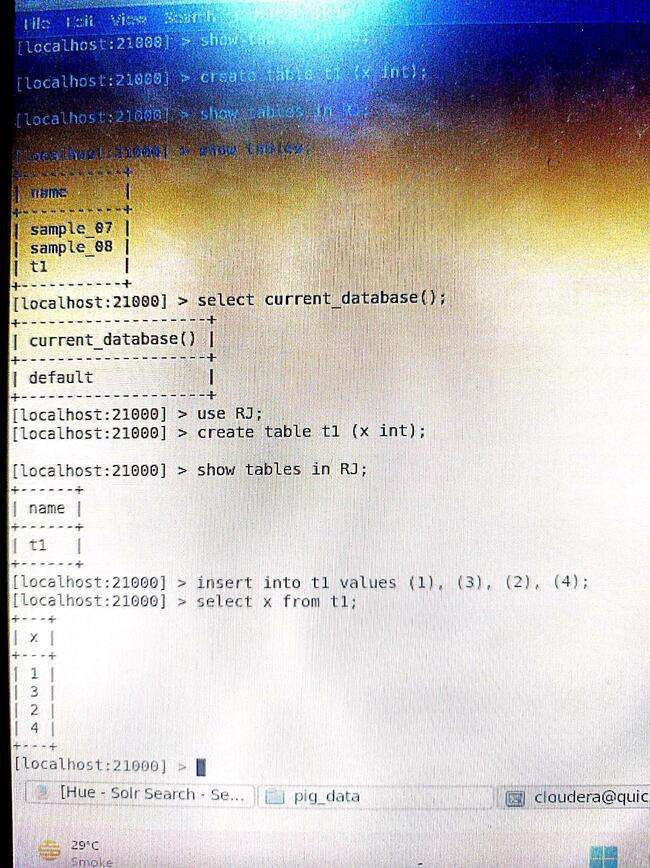
****

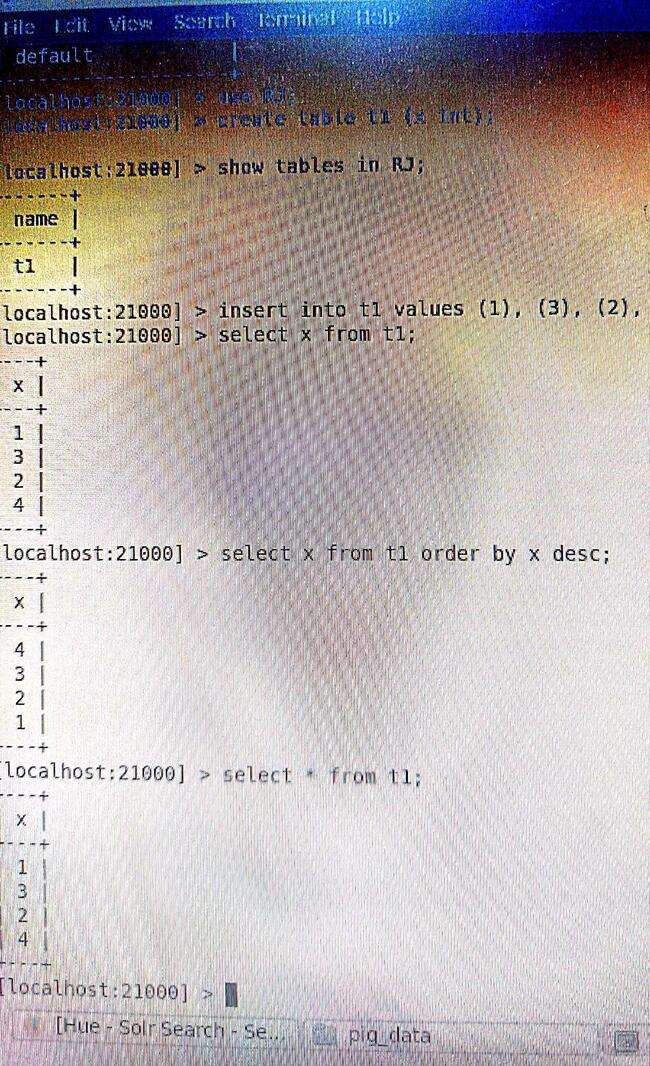
****

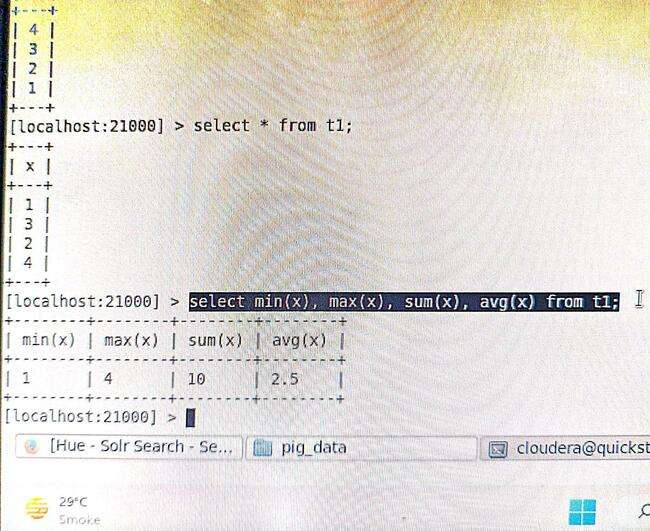
****

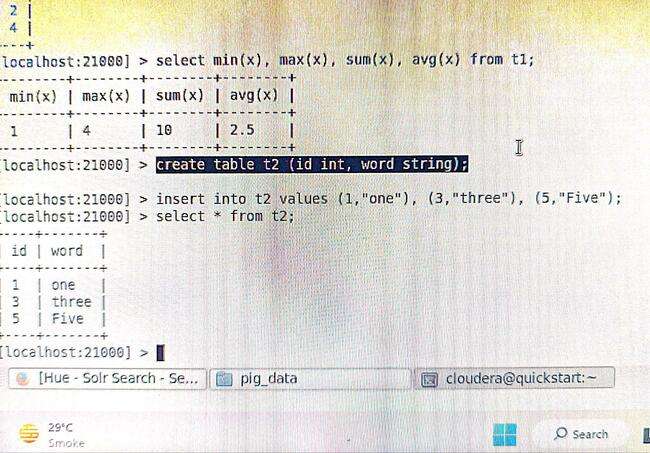
****

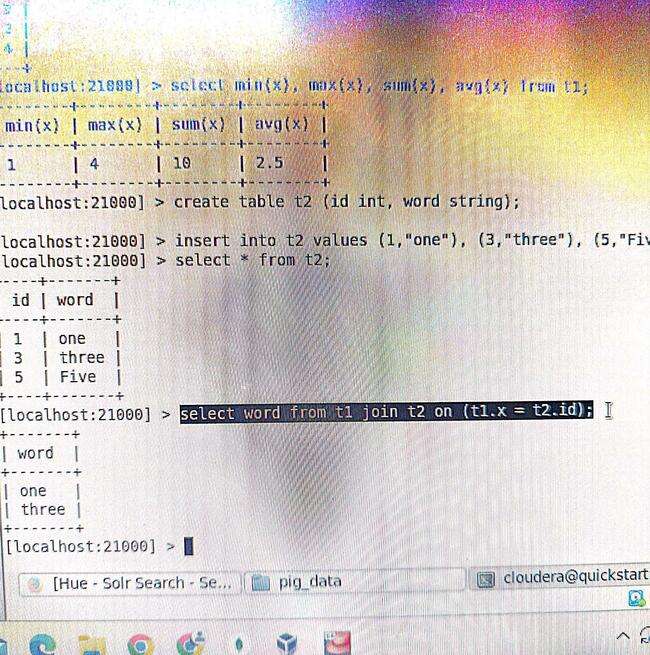
****

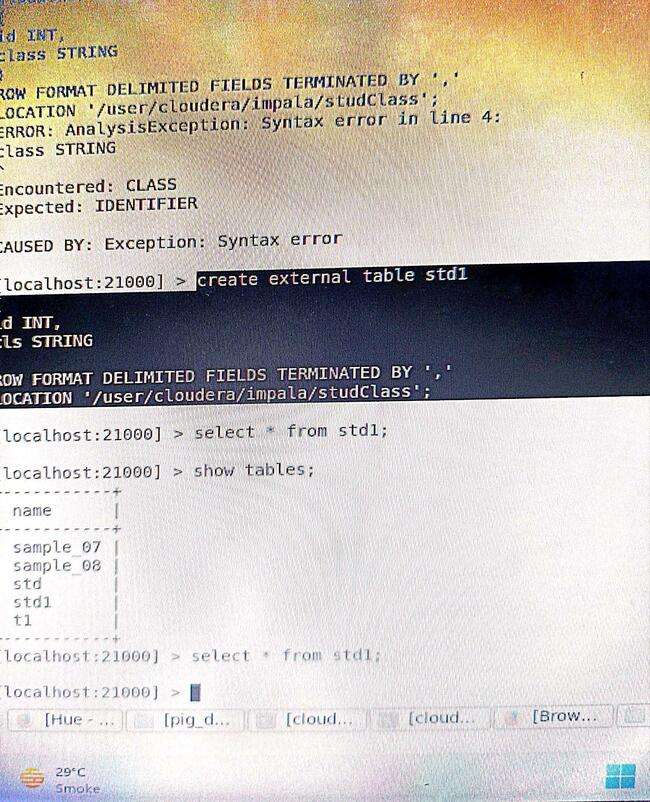
****

****

****

****

****

****

**Done**