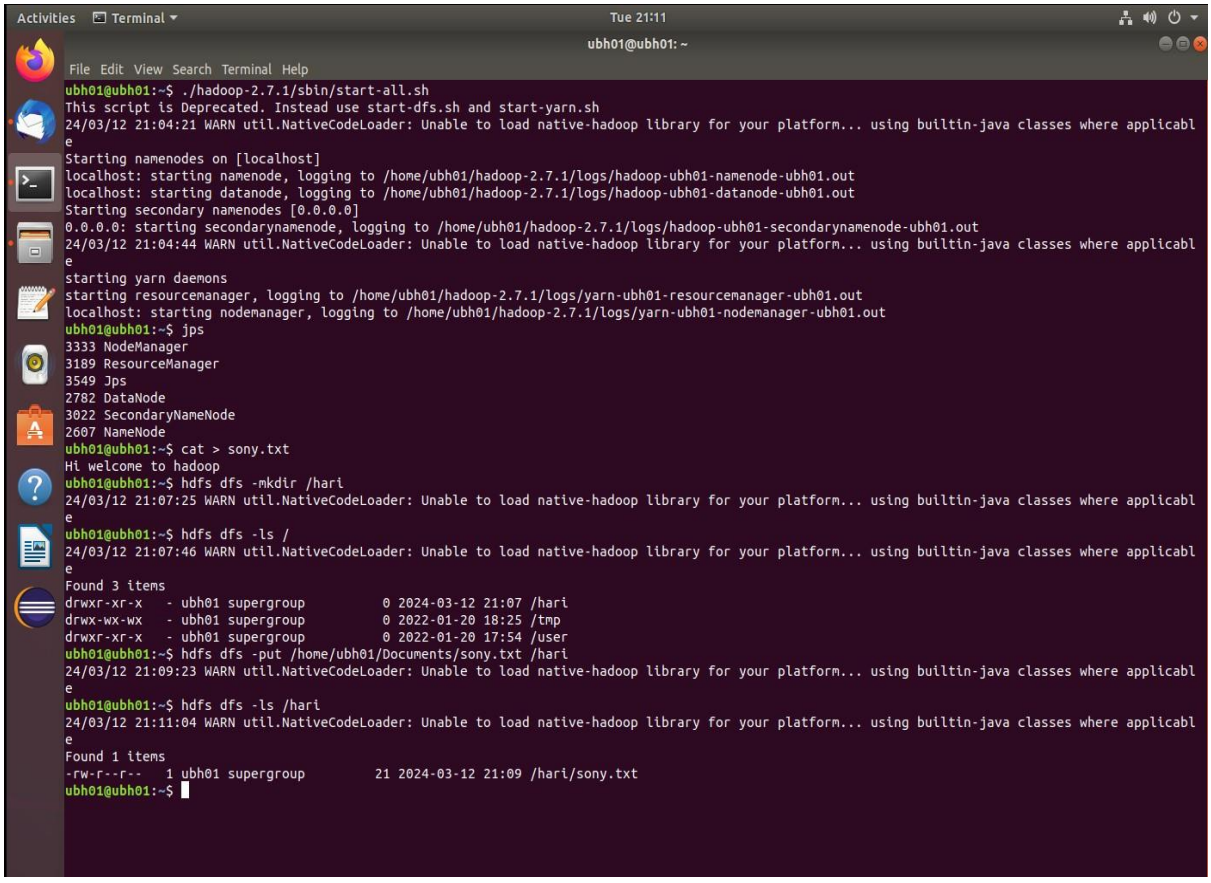
BIG DATA

CSDAIA24GP003
Chakka Sudheer Kumar
Emp-id:2320418

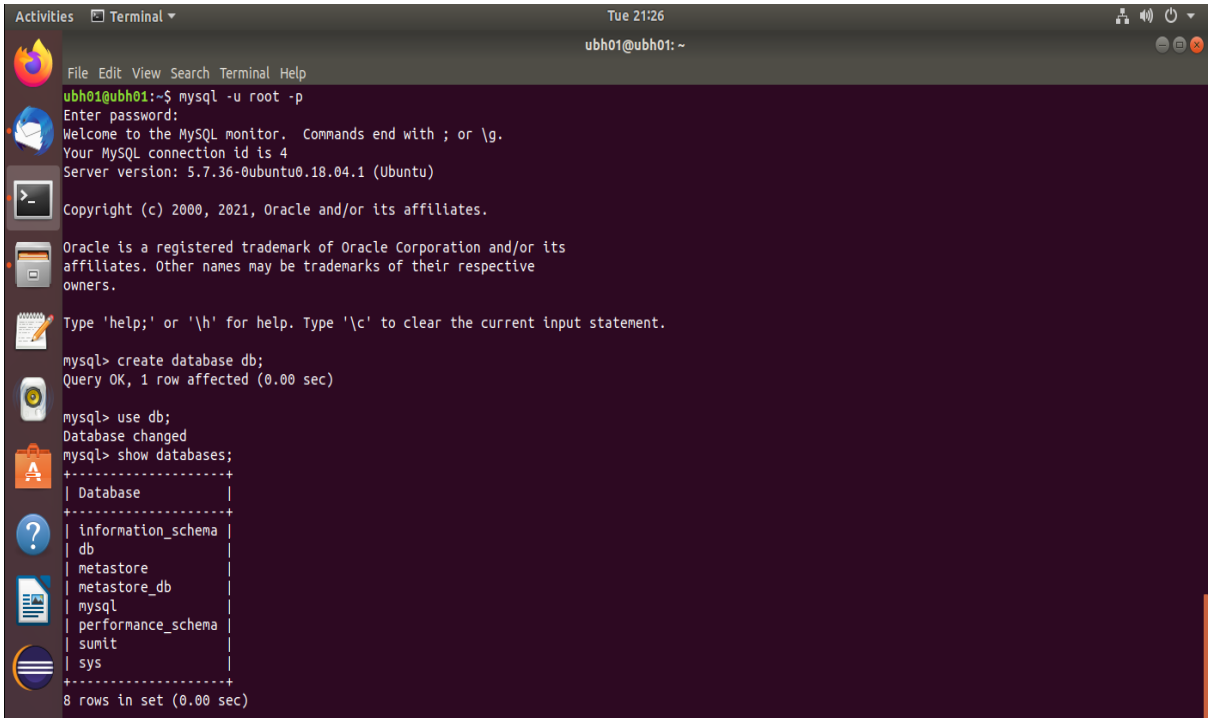
Hadoop

- Hadoop is an open-source framework used for storing and processing big data.
- It provides a distributed file system called Hadoop Distributed File System (HDFS).

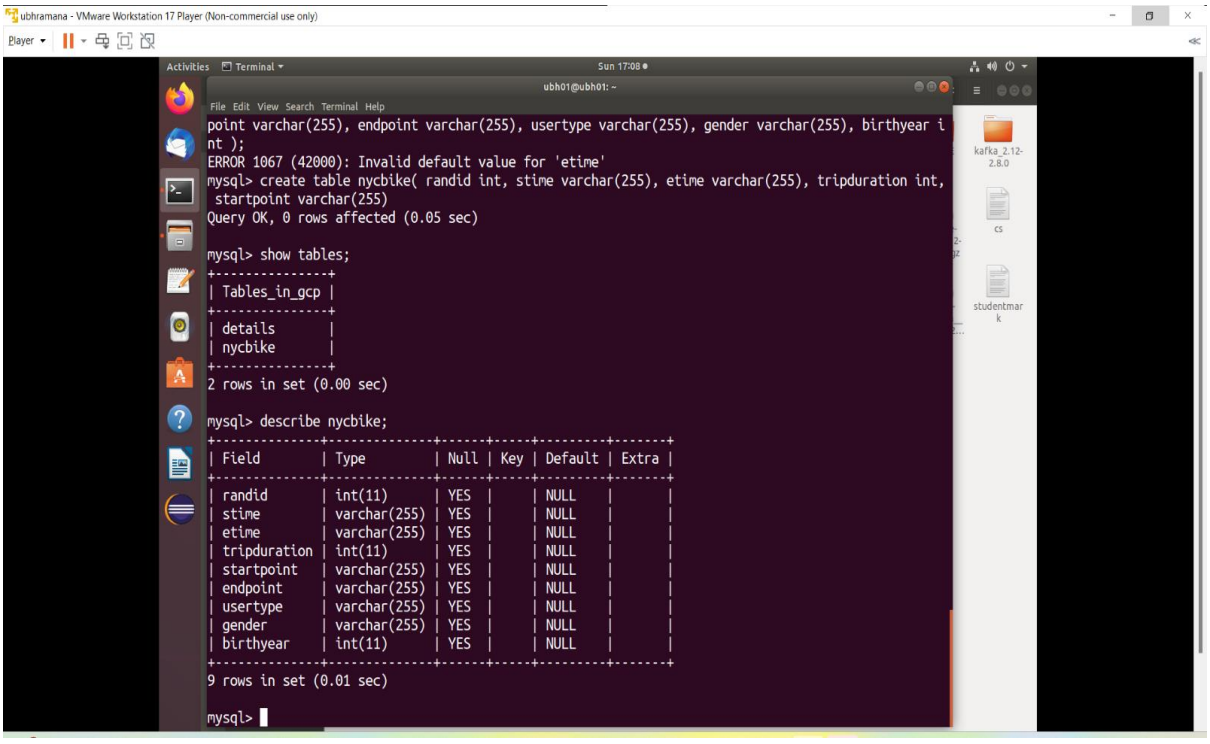
1. To start the Hadoop we use --**./hadoop-2.7.1/sbin/start-all.sh**
2. To check whether Hadoop daemons are running or not --- **jps**
3. Create a file in local -- **cat > filename.txt**
4. Create a directory in local --- **mkdir name**
5. Just check whether file is copied or not using --- **ls**



*** Start MYSQL and create a database with named db



- Created table with table name student
  Syntax:
  Create table student(name varchar(20),roll int primary key, age int);



- Inserted the values manually or load values from csv file as shown below:

```
mysql> load data local infile '/home/ubh01/new_york_city2.csv' into table nycbike fields terminate
d by ',' enclosed by '"' lines terminated by '\n' ignore 1 rows (randid,stime,etime,tripduration,s
tartpoint,endpoint,usertype,gender,birthyear);
Query OK, 837 rows affected, 98 warnings (0.28 sec)
Records: 837  Deleted: 0  Skipped: 0  Warnings: 98

mysql>
```

SQOOP:

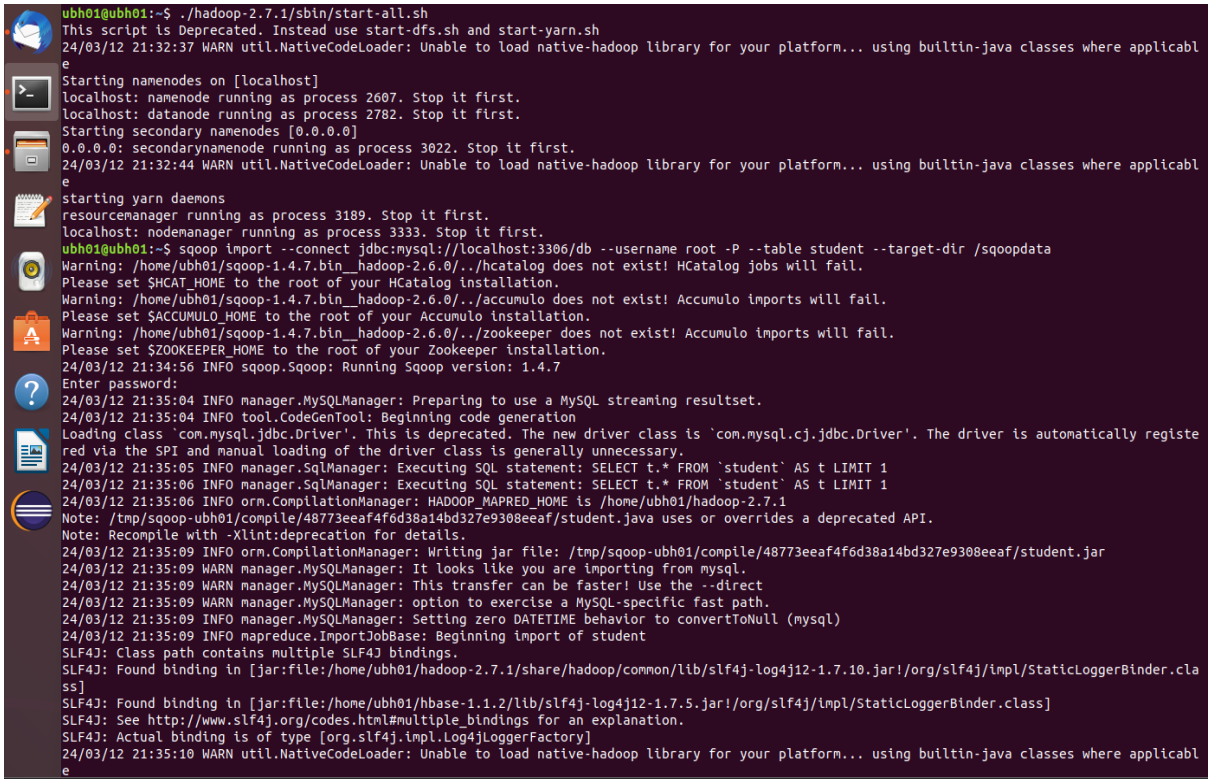
It is a command line tool that helps to import and export data from database.

- **Sqoop import:**
  Sqoop import is used to import the data from RDBMS to HDFS.

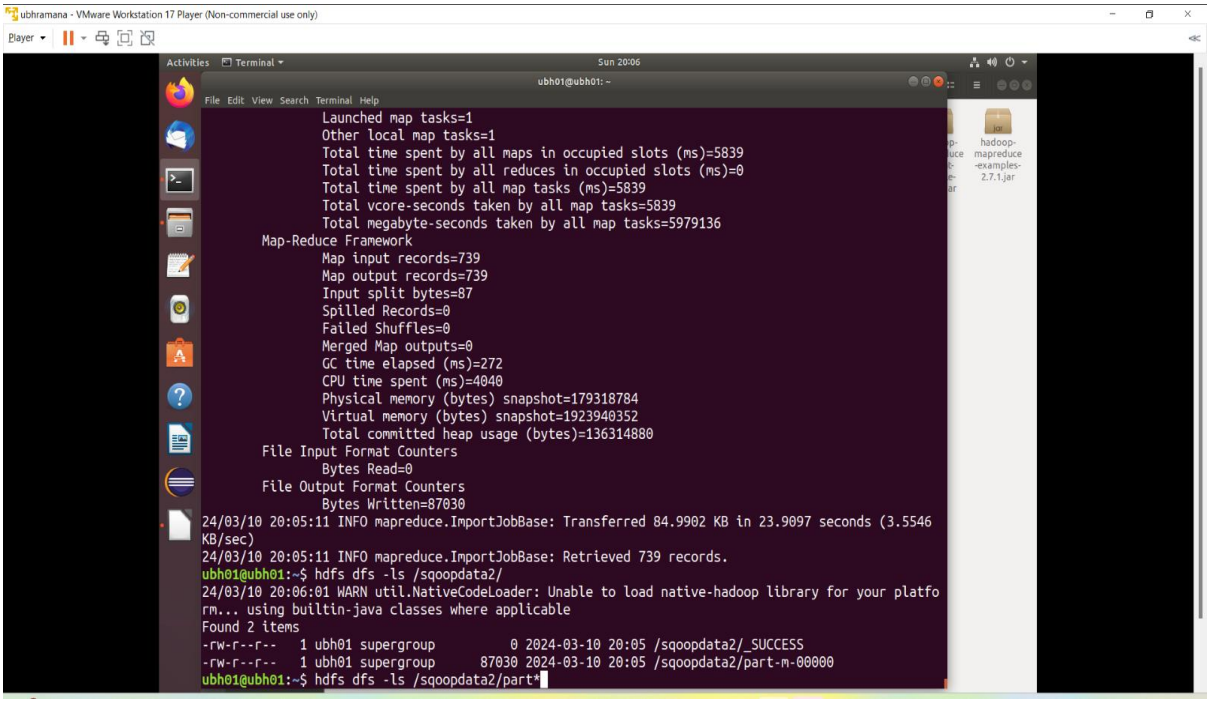
*****To start the Hadoop we use ./hadoop-2.7.1/sbin/start-all.sh**

- Using the below command the table nycbike from database gcp is imported to hdfs through sqoop import

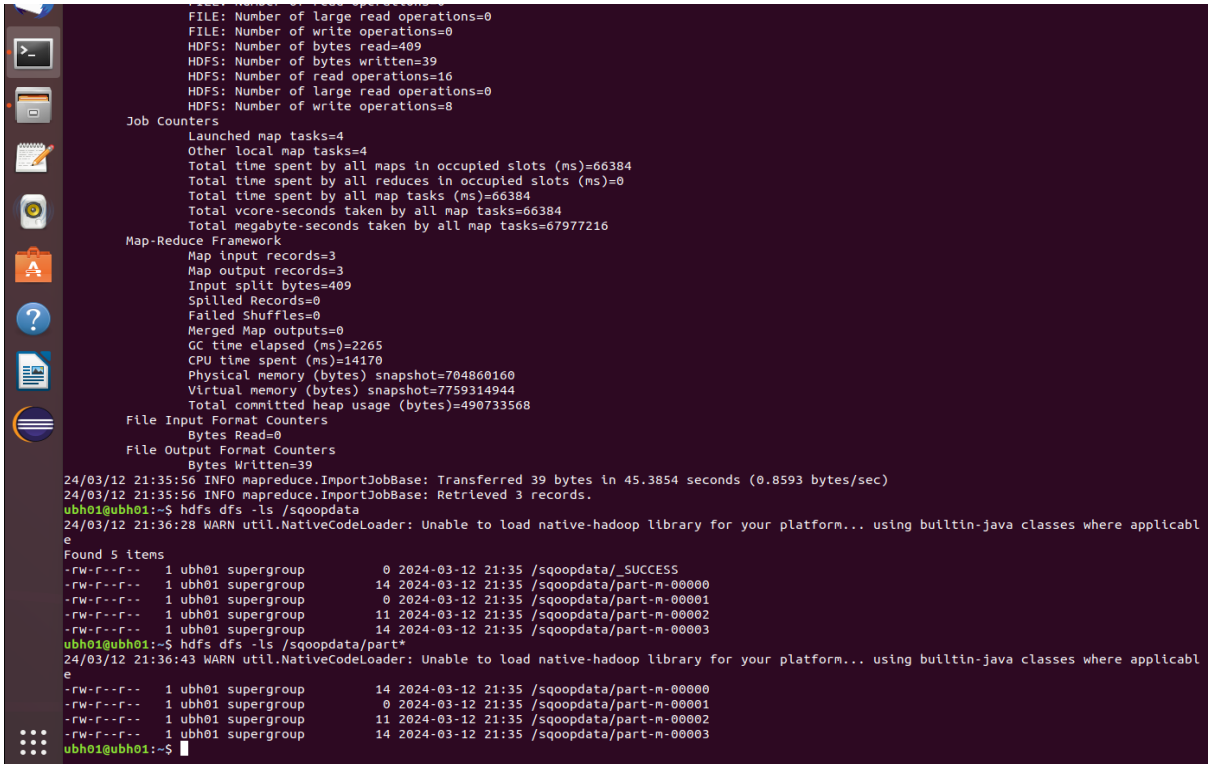
Syntax:
Sqoop import –connectjdbc:mysql://localhost:3306/gcp –username root -p –query –table student –target-dir /sqoopdata



- Listing the output files of sqoopdata file which includes _SUCCESS and PART FILES using ls command.

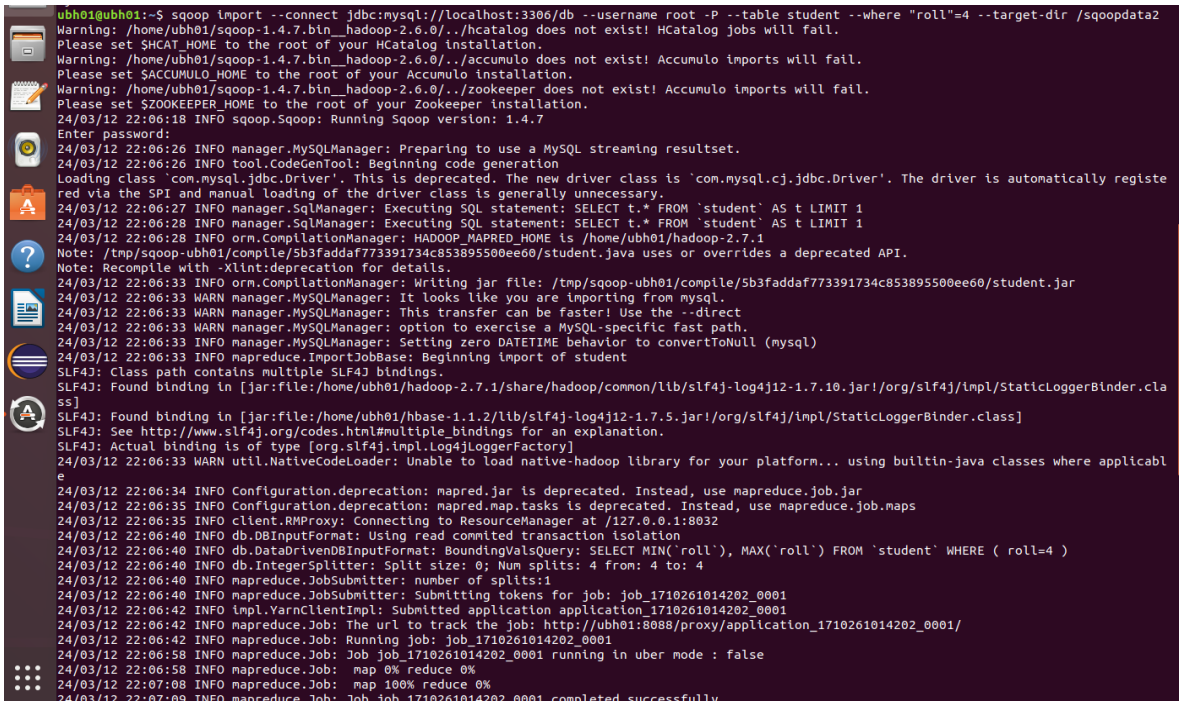


- Display the content inside PART FILE using cat command.

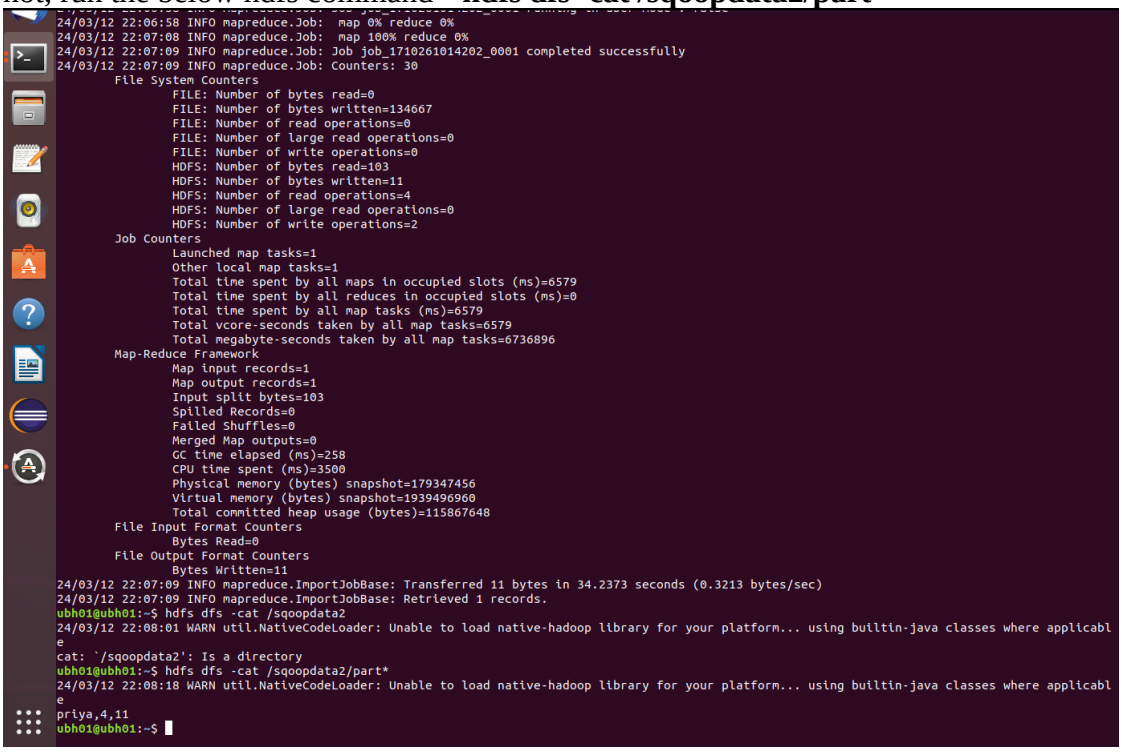


- Sqoop import command with **where** clause.
  Syntax:

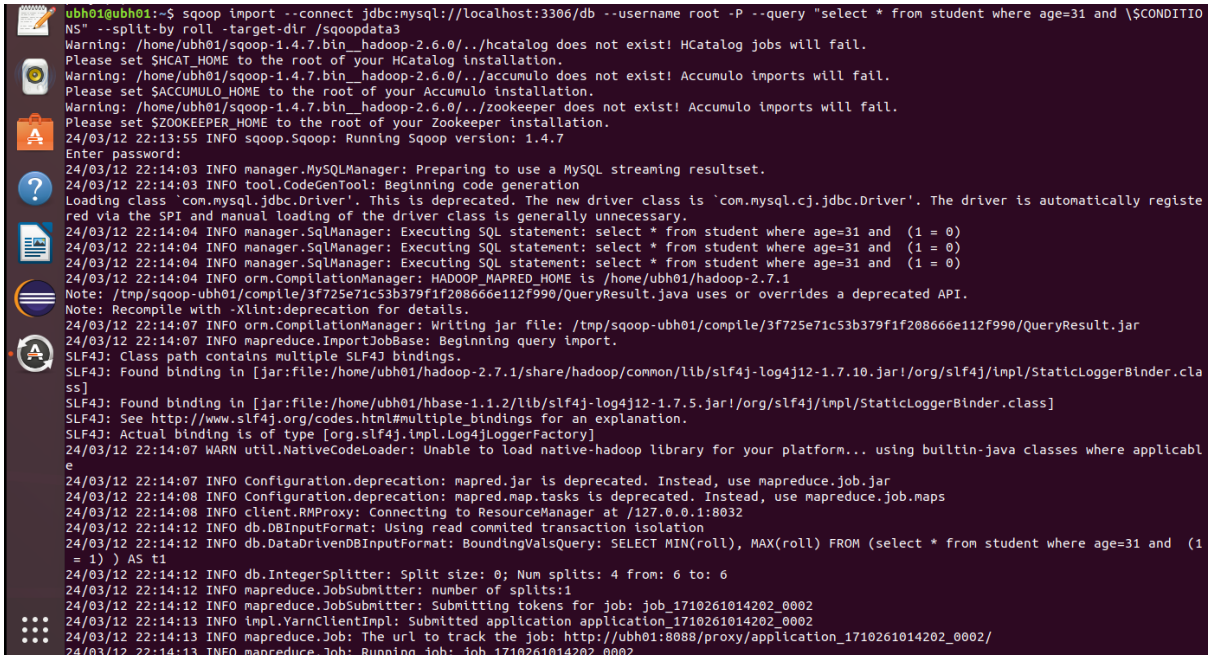
Sqoop import –connect jdbc:mysql://localhost:3306/db –username root -p –query “select * from student where ‘roll’=4 –target-dir /sqoopdata2



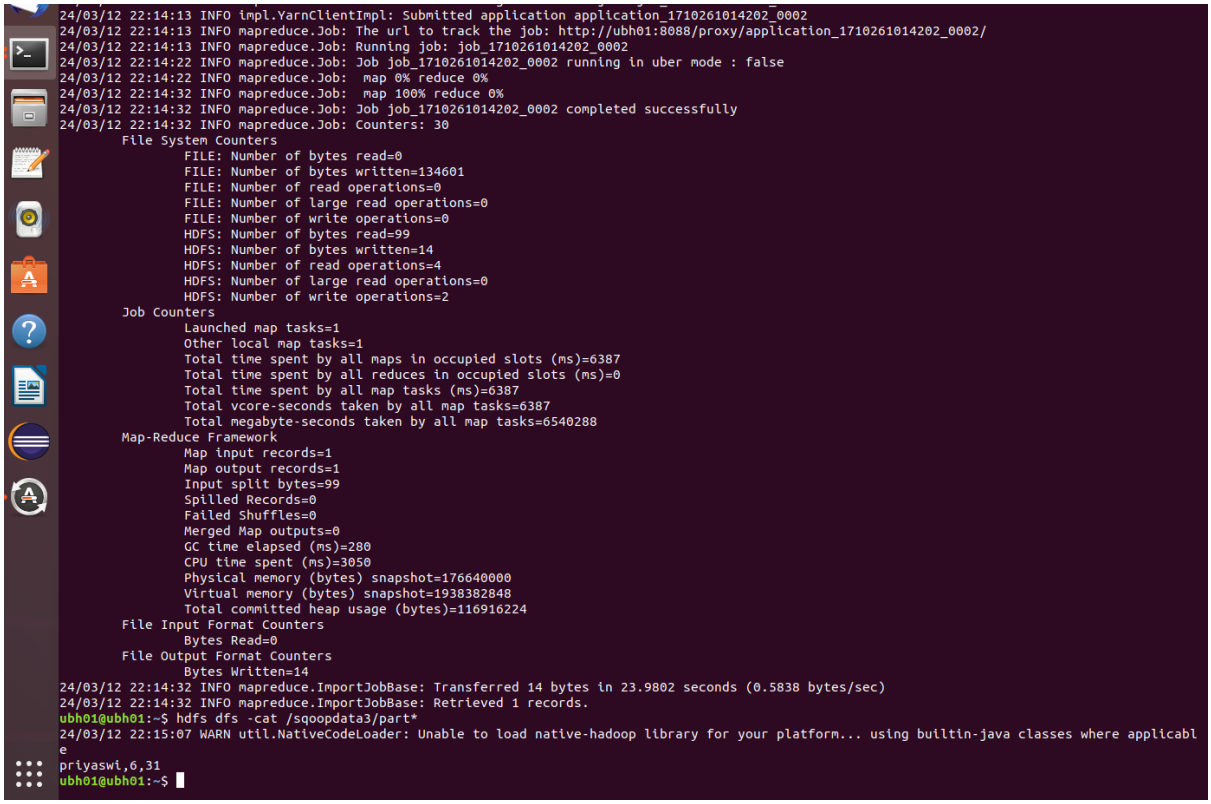
- If we want to verify whether values that satisfies where condition imported to hdfs or not, run the below hdfs command **--hdfs dfs -cat /sqoopdata2/part***



- Sqoop import with –query command where we will specify mysql query and \$CONDITIONS that is a placeholder in query which divide the data with more where clauses internally and –split-by is used to split the data by particular column that is primary key.

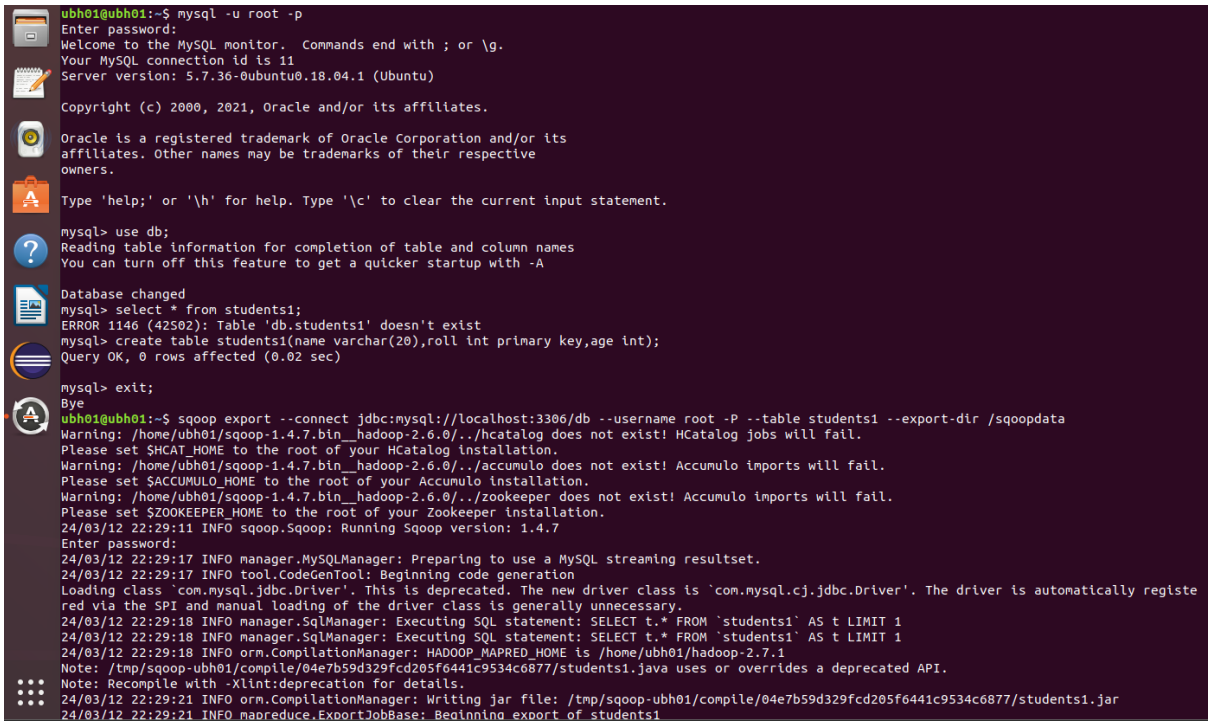


- If we want to verify whether values that satisfies QUERY imported to hdfs or not, run the below hdfs command **--hdfs dfs -cat /sqoopdata3/part***

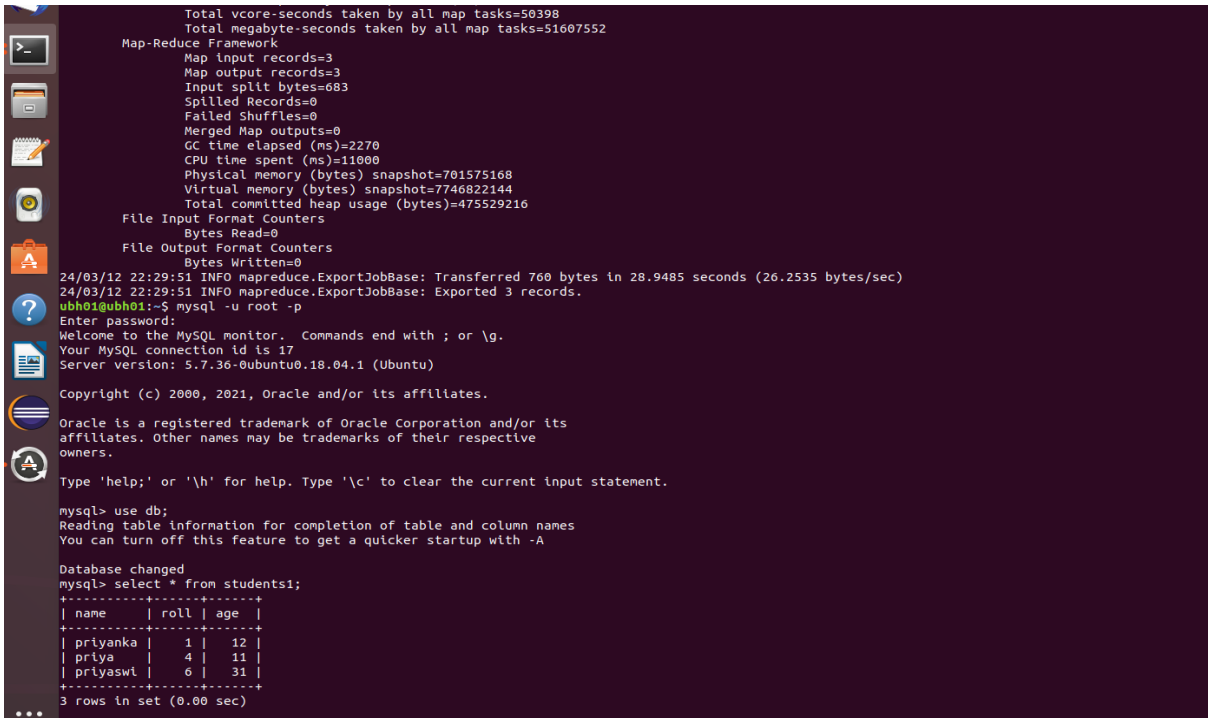


- **Sqoop export**:

  - Sqoop export is used to import the data from HDFS to RDBMS.
  - Here we have to mention export keyword and –export-directory and before this we have to create a empty schema or table in mysql.



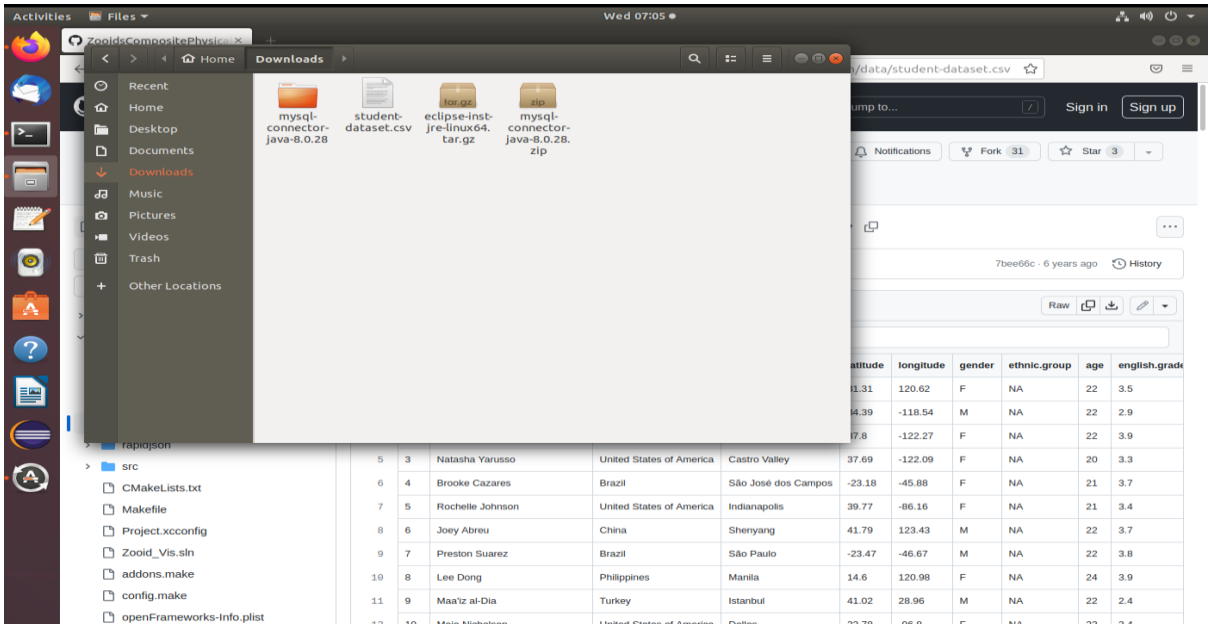
- If we want to verify whether values are exported from hdfs to mysql or not by running the below command in mysql—select * from students1;



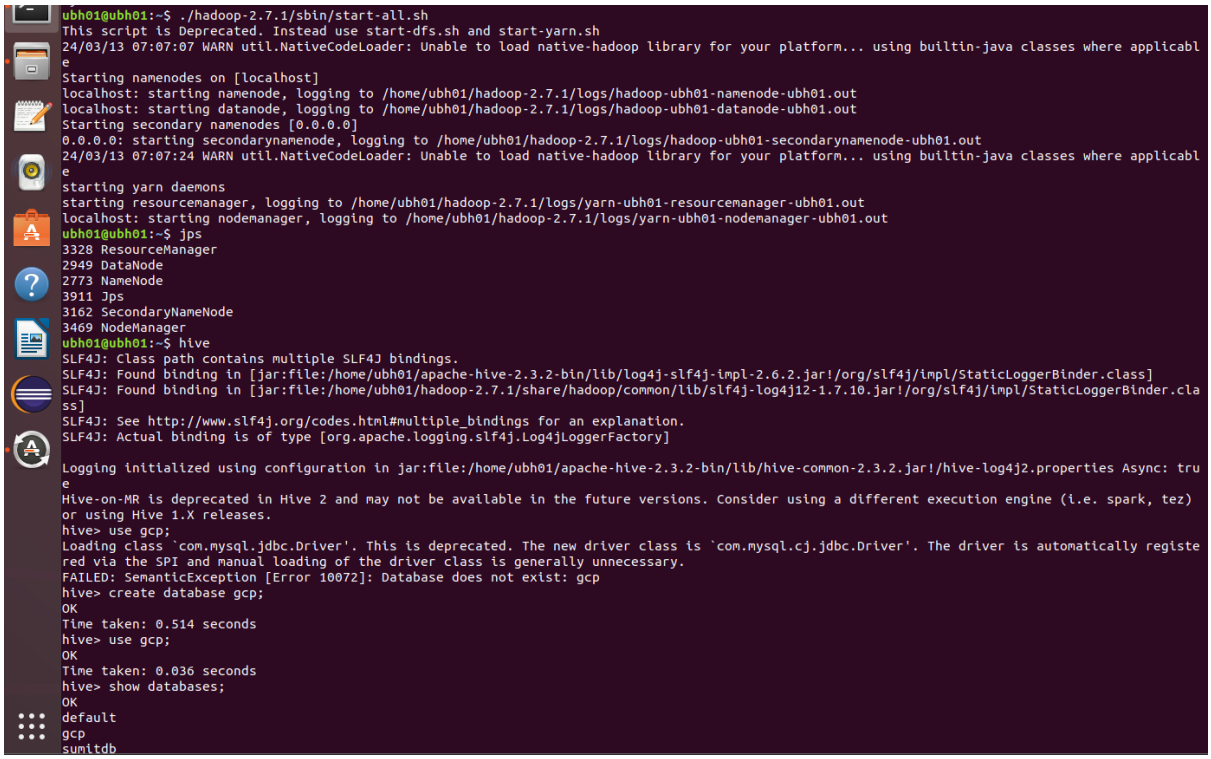
HIVE

Hive is a data warehouse system which is used for querying and analysing large datasets stored in HDFS.

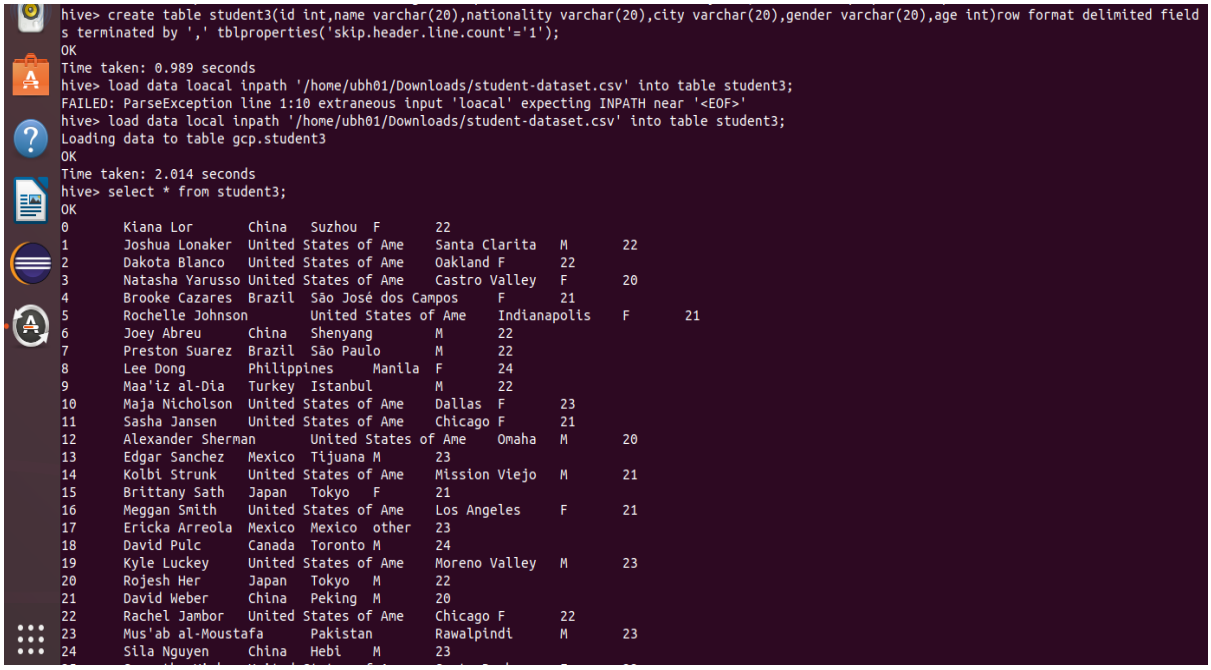
In hive here we are downloading the dataset(student-dataset.csv) from github.



- To entering into the hive, use **hive** command.
- Create a database and use database.



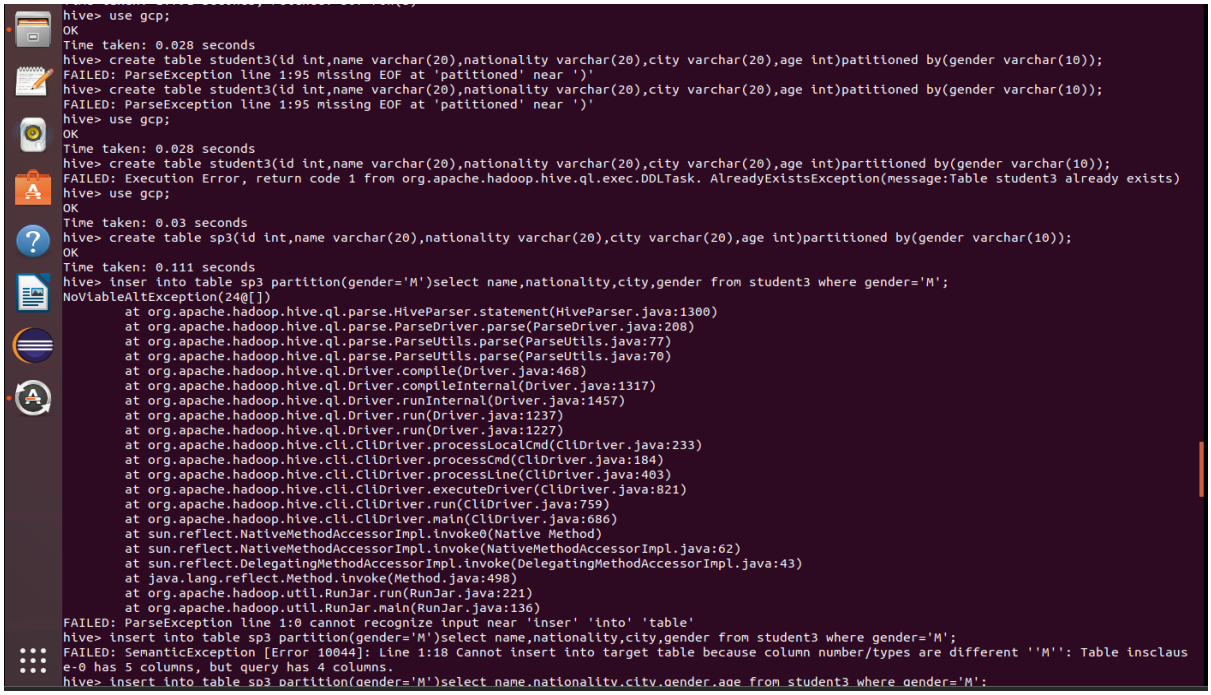
- Here we are creating a hive table and inserted values into table using load command.
- To verify whether the data is loaded are not, use command**--select * from student3;**



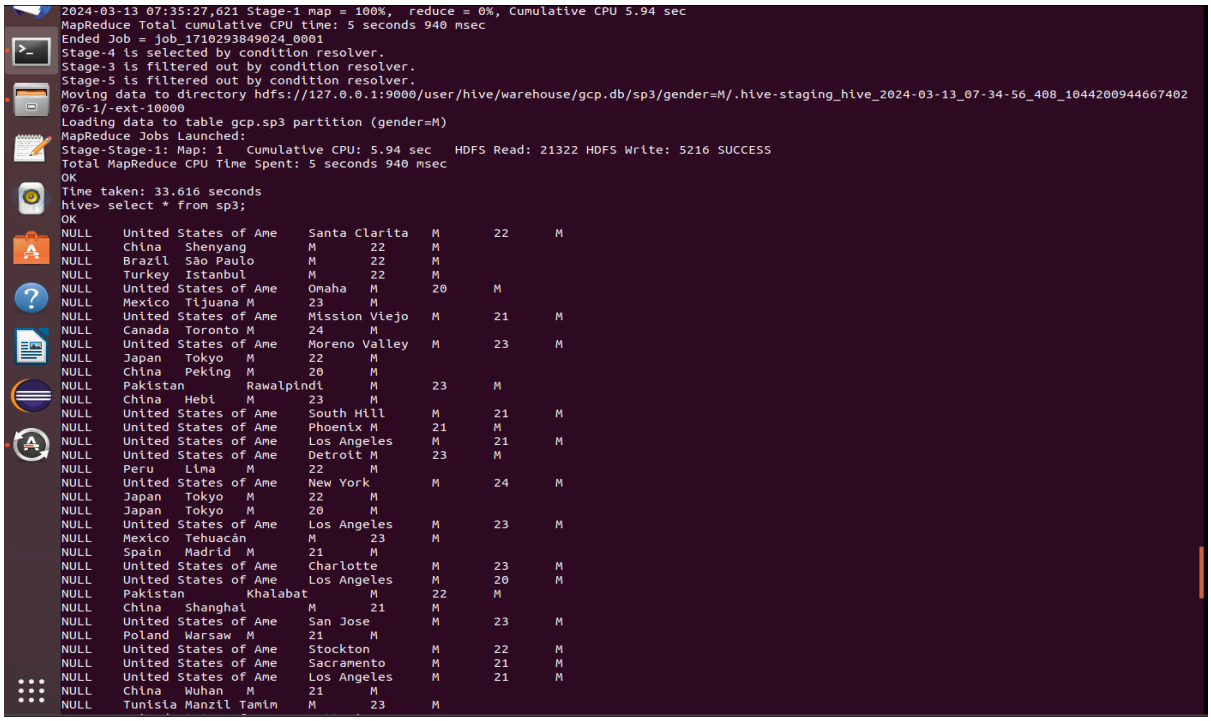
Hive Partitioning:

Create a partition table and follow either static or dynamic partitioning to insert values from main table following conditions.

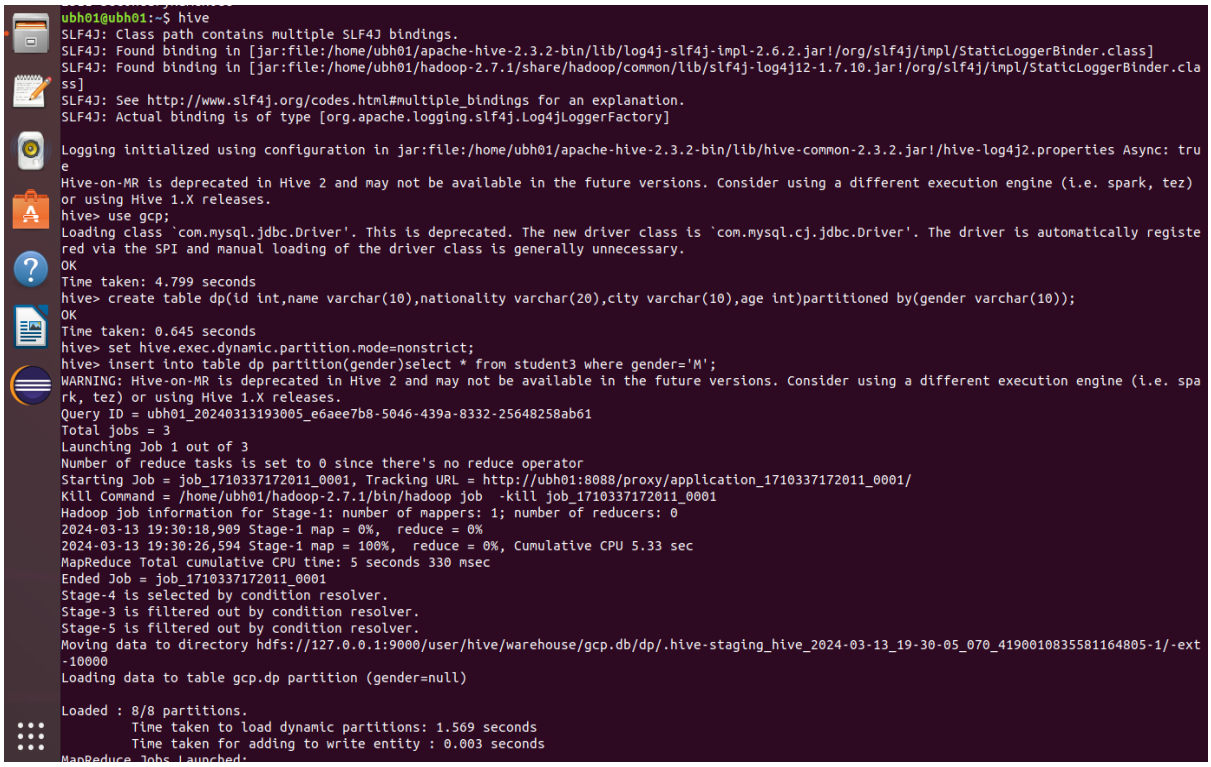
- Static partitioning:
  - It is setup when the table is created and remains static, meaning the partitions don't change unless explicitly altered by the user.
  - Here we are inserting values of a main table into partition table using static partitioning.



- Here is the partition table for partition gender = ’M’ shown below



- **Dynamic partitioning**:
  - It involves automatically creating partitions for data based on the values of specified columns during the insertion process.
  - Here we are inserting values of a main table into partition table using dynamic partitioning.



- To verify whether the data is loaded are not, use command**--select * from dp;**

```
hive> select * from dp;
OK
173     Sean Bruso      United Kingdom  London  NULL    19
12      Alexander       United States of Ame    Omaha   NULL    20
21      David Webe      China   Peking  NULL    20
46      Alan Trinh      Japan   Tokyo   NULL    20
58      Cameron St      United States of Ame    Los Angele      NULL    20
94      Devon Mira      Colombia        Pereira NULL    20
98      Jason Hund      United States of Ame    Los Angele      NULL    20
106     Ryan Barre      United States of Ame    Chicago NULL    20
121     James Rice      Canada  Toronto NULL    20
150     Abdul Jabb      Egypt   Cairo   NULL    20
196     Dylan Bell      United States of Ame    Tucson  NULL    20
235     Ryan Tyler      United States of Ame    Los Angele      NULL    20
268     Bryant Ron      Mexico  Aguascalie      NULL    20
269     Garrett He      United States of Ame    Mobile  NULL    20
281     Dyllan Pai      United States of Ame    Sugar Land      NULL    20
283     Jesse Carb      Colombia        Bogotá  NULL    20
288     Weldon Hig      United States of Ame    New York        NULL    20
302     Austin Haa      United States of Ame    Columbus        NULL    20
304     Zachary Mu      United States of Ame    Los Angele      NULL    20
14      Kolbi Stru      United States of Ame    Mission Vi      NULL    21
27      Brandon Ba      United States of Ame    South Hill      NULL    21
32      Michael Be      United States of Ame    Phoenix NULL    21
33      Sean Rozga      United States of Ame    Los Angele      NULL    21
52      Zachary Br      Spain   Madrid  NULL    21
61      Colin Lemo      China   Shanghai        NULL    21
65      Cameron Ha      Poland  Warsaw  NULL    21
71      Joseph Smi      United States of Ame    Sacramento      NULL    21
75      Joseph Sni      United States of Ame    Los Angele      NULL    21
76      Sourinthon      China   Wuhan   NULL    21
86      Myles Vaug      United States of Ame    Waukegan        NULL    21
87      Juan Guerr      Mexico  Mexico  NULL    21
123     Gareth New      United States of Ame    Los Angele      NULL    21
124     Austin Har      United States of Ame    Las Vegas       NULL    21
130     Casey Buhr      United States of Ame    Phoenix NULL    21
```

Bucketing:

It is a method of dividing data within partitions into fixed-size buckets based on the hash value of a specified column. It helps distribute data evenly, Improving the query performance by reducing the data.

- Create a bucket table and mention how many buckets you need.
- Now insert main table values into bucket table using--

insert into table buckettablename select * from maintablename

```
hive> create table bk1(id int,name varchar(10),nationality varchar(20),city varchar(10),gender varchar(10),age int)clustered by(city) into 3 bucke
ts;
OK
Time taken: 0.152 seconds
hive> insert into table bk1 select * from student3;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spa
rk, tez) or using Hive 1.X releases.
Query ID = ubh01_20240313193834_92da57ee-472d-4c3c-9f82-0ce2d3a770f3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
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_1710337172011_0002, Tracking URL = http://ubh01:8088/proxy/application_1710337172011_0002/
Kill Command = /home/ubh01/hadoop-2.7.1/bin/hadoop job  -kill job_1710337172011_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 3
2024-03-13 19:38:44,882 Stage-1 map = 0%,  reduce = 0%
2024-03-13 19:38:53,377 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.14 sec
2024-03-13 19:39:08,365 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 10.13 sec
2024-03-13 19:39:12,154 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 16.47 sec
2024-03-13 19:39:14,826 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 23.57 sec
MapReduce Total cumulative CPU time: 23 seconds 570 msec
Ended Job = job_1710337172011_0002
Loading data to table gcp.bk1
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 3   Cumulative CPU: 23.57 sec   HDFS Read: 36661 HDFS Write: 14006 SUCCESS
Total MapReduce CPU Time Spent: 23 seconds 570 msec
OK
Time taken: 42.604 seconds
hive>
```

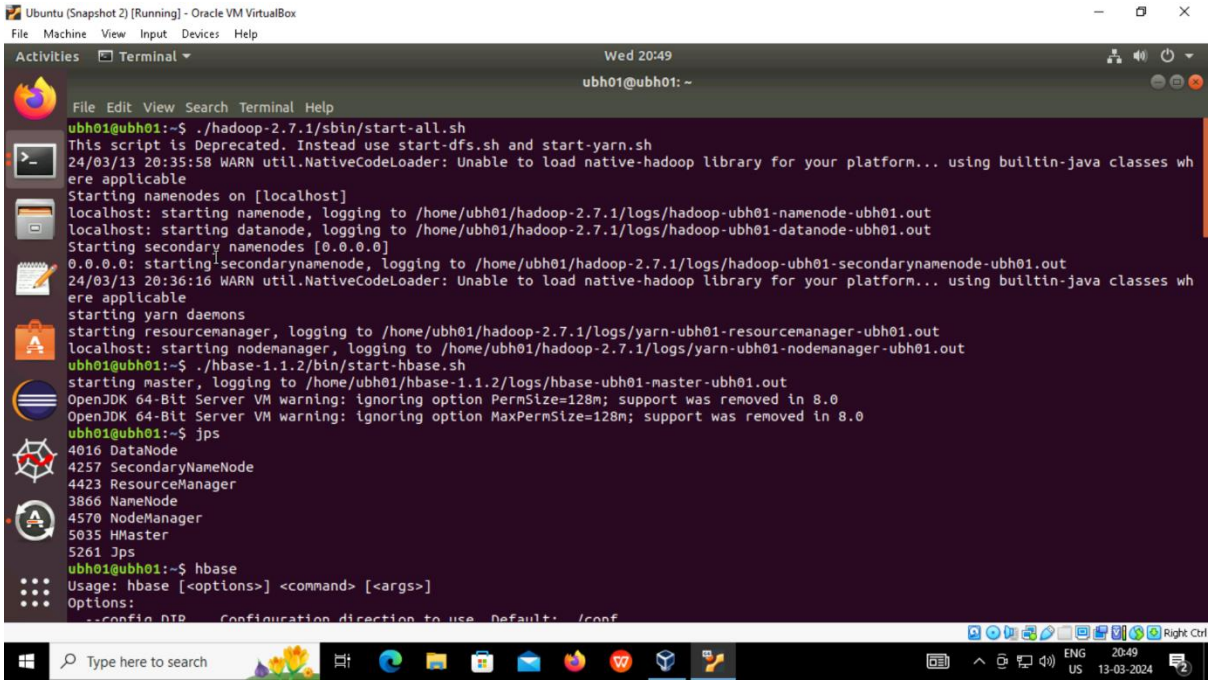
Check for bucket creation in hive directory of hdfs:

```
hive> exit;
ubh01@ubh01:~$ hdfs dfs -ls /user/hive/warehouse/gcp.db
24/03/13 19:40:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicabl
e
Found 5 items
drwxrwxr-x   - ubh01 supergroup          0 2024-03-13 19:37 /user/hive/warehouse/gcp.db/bk
drwxrwxr-x   - ubh01 supergroup          0 2024-03-13 19:39 /user/hive/warehouse/gcp.db/bk1
drwxrwxr-x   - ubh01 supergroup          0 2024-03-13 19:30 /user/hive/warehouse/gcp.db/dp
drwxrwxr-x   - ubh01 supergroup          0 2024-03-13 07:34 /user/hive/warehouse/gcp.db/sp3
drwxrwxr-x   - ubh01 supergroup          0 2024-03-13 07:23 /user/hive/warehouse/gcp.db/student3
```

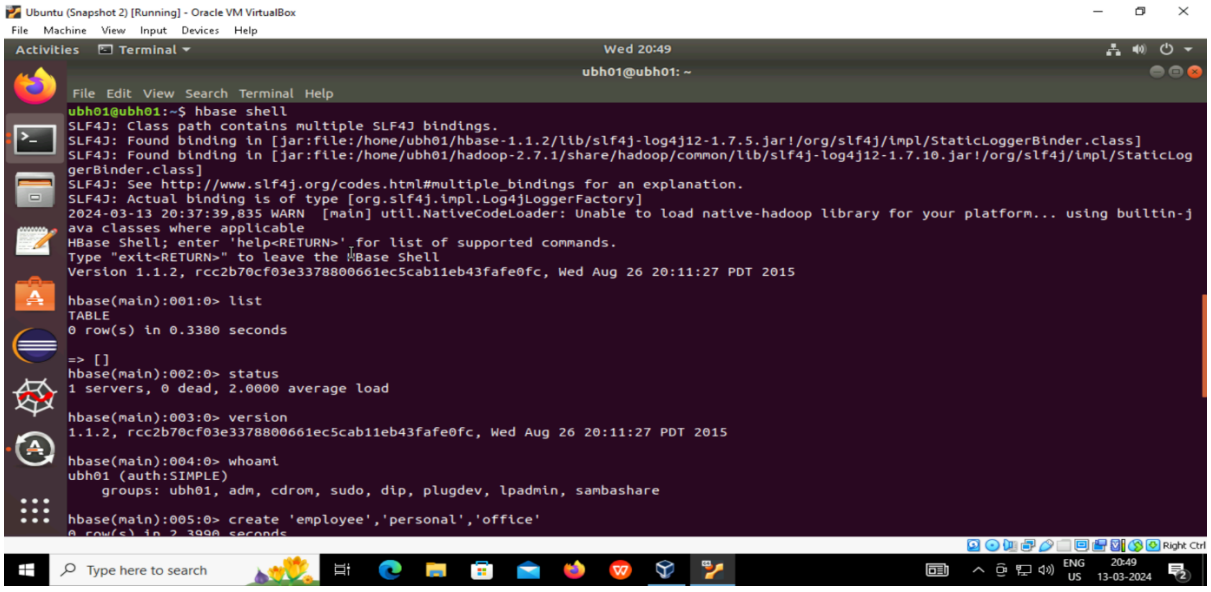
HBase:

HBase is an open-source, distributed, column-oriented database built on top of the HDFS. It provides real-time read/write access to large datasets, with horizontal scalability and fault tolerance.

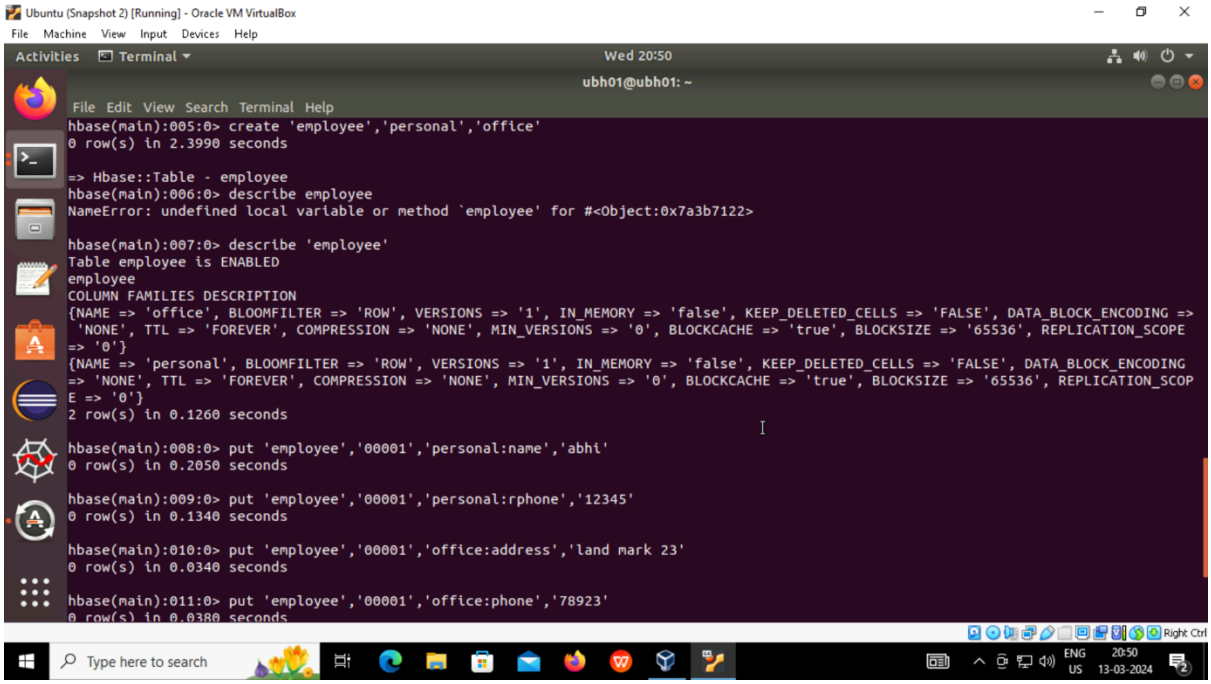
- Start Hadoop service by running Hadoop start command
- Then start HBase service by running command –**hbase**



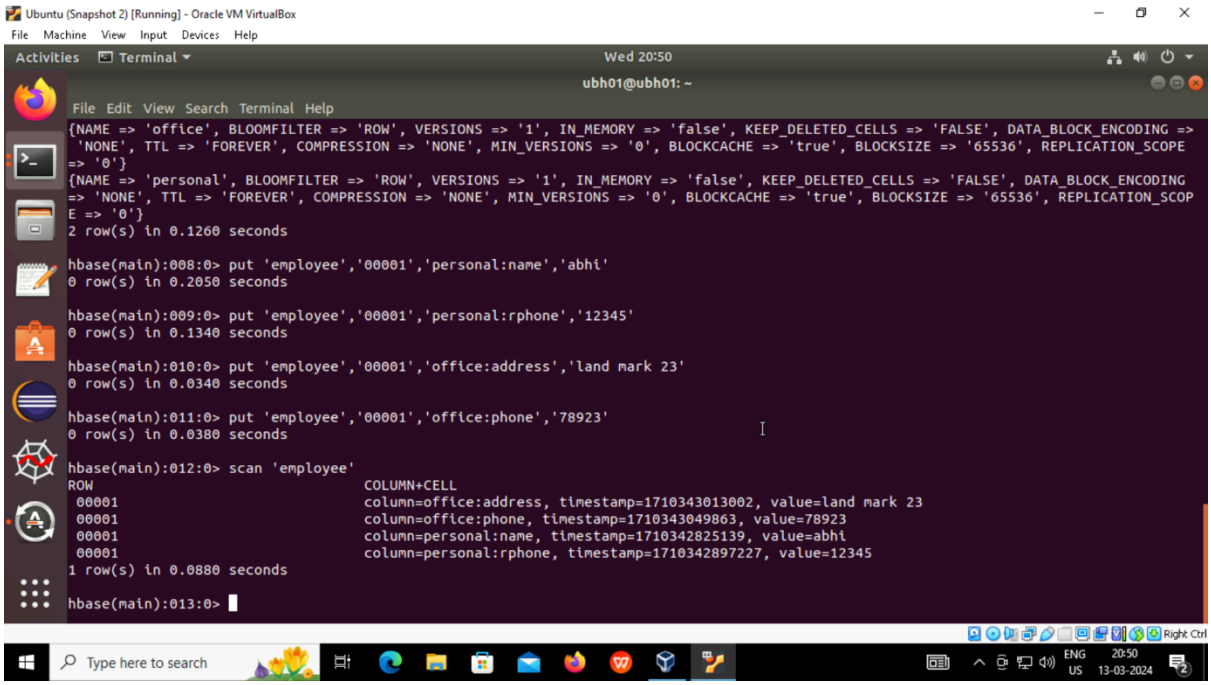
- Now start hbase script by running command – **hbase shell**
- To verify list of tables in hbase, the command is – **list**
- To check status and version of hbase, the commands are – **status** and – **version**
- To verify current user, the command is –whoami



- To create columnar table in HBase,
  Syntax:
  create ‘tablename’,’columnarfamily_name1’,’columnarfamily_name2’
- To get details of a table, the command is –**describe ‘tablename’**
- To insert a value into table,
  Syntax:
  put ‘tablename’,’rowkey’,’columnarfamily_name:column_name’,’value’



- After inserting all values into table, to show all values in the table,
  The command is – **scan ‘tablename’**



- To retrieve a particular row of a columnar table,
  The command is –**get ‘tablename’,’rowkey’**