VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS

Submitted by

VARUN URS M S (1BM20CS182)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
April-2023 to July-2023

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "LAB COURSE **BIG DATA ANALYTICS**" carried out by **VARUN URS M S (1BM20CS182),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (20CS6PEBDA)** work prescribed for the said degree.

Dr. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

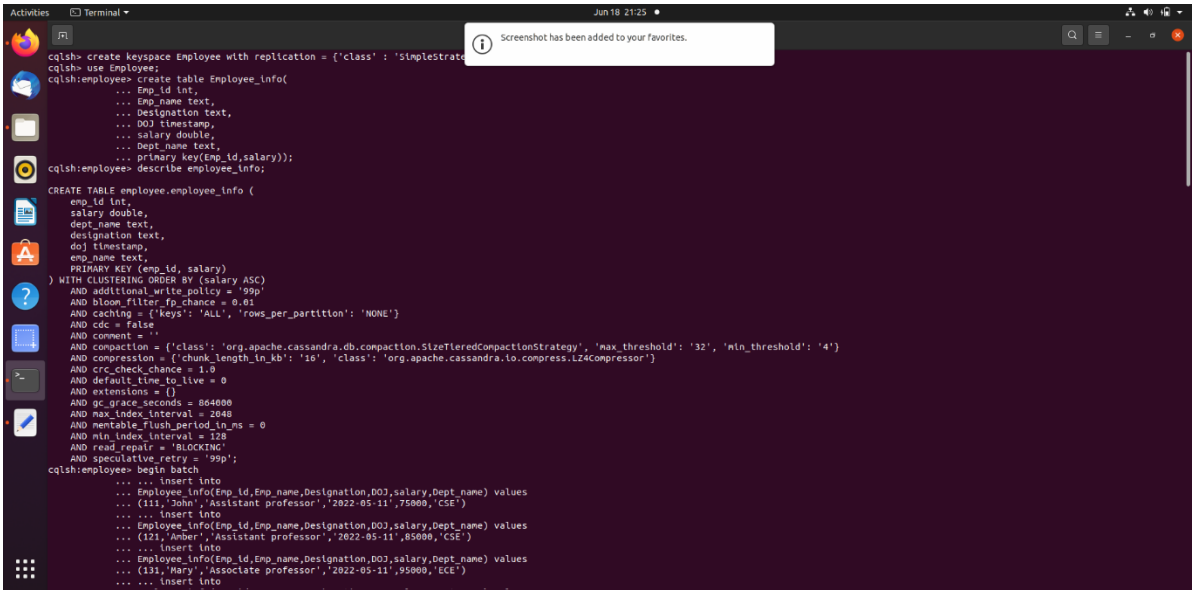
Index Sheet

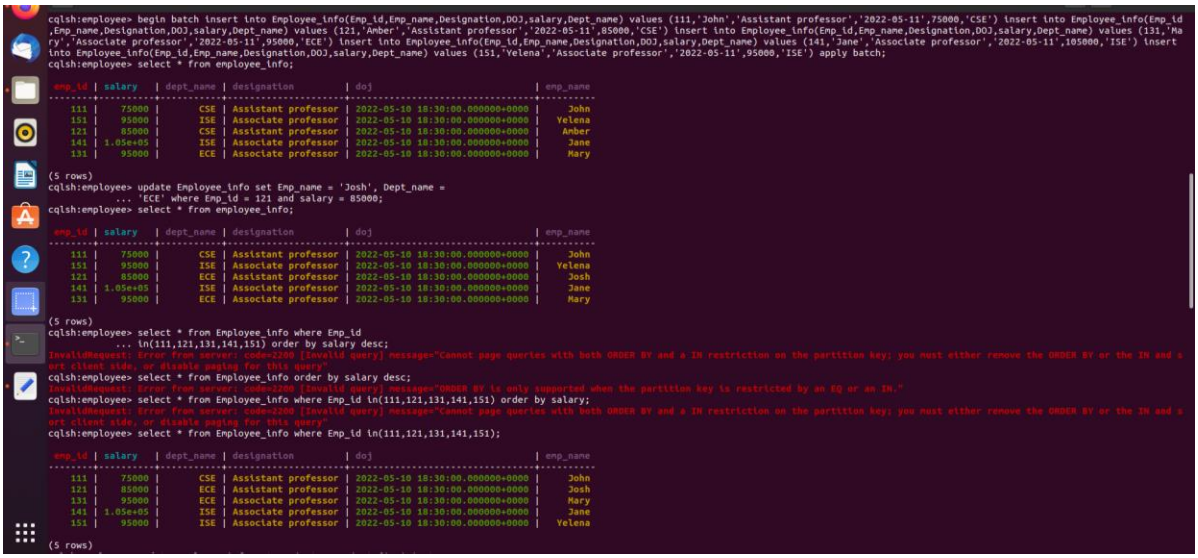
Course Outcome

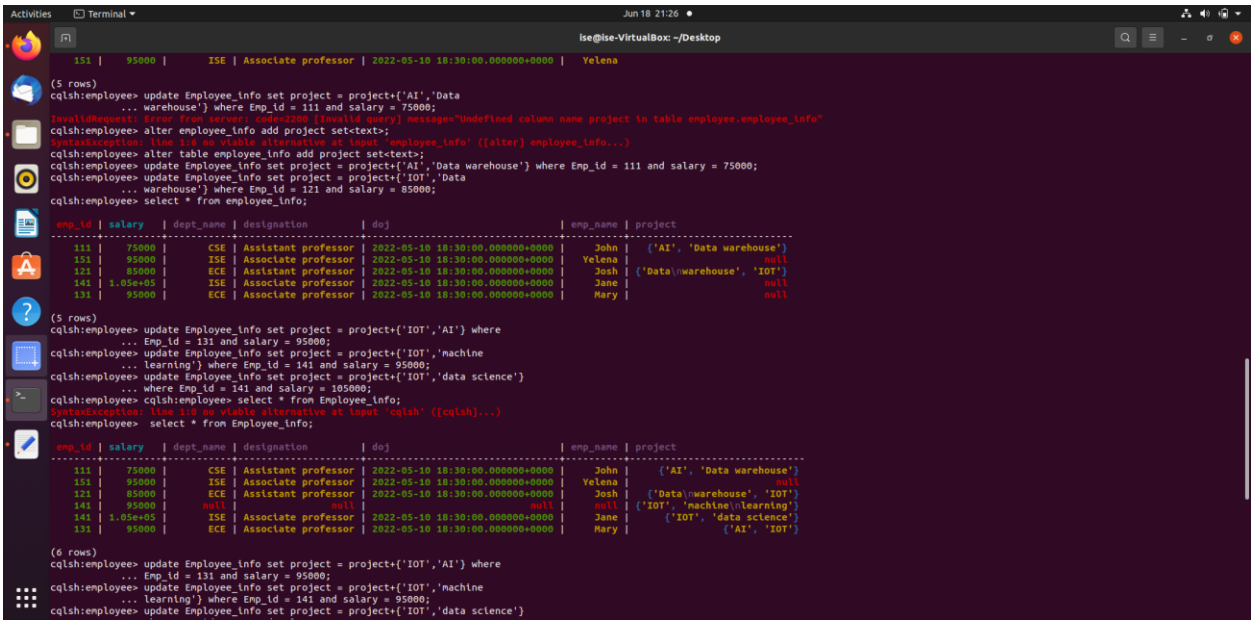
CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO3	Design and implement big data applications by applying NoSQL, Hadoop or Spark

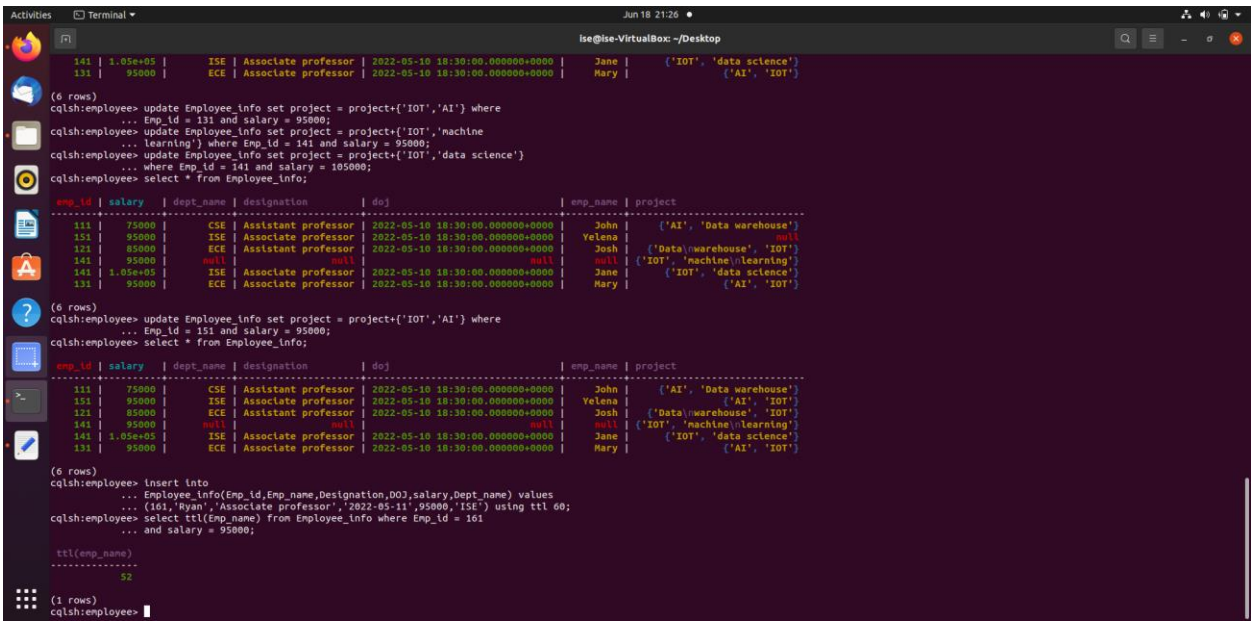
LAB 1 Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee
2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name
3. Insert the values into the table in batch
4. Update Employee name and Department of Emp-Id 121
5. Sort the details of Employee records based on salary
6. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.
7. Update the altered table to add project names.
8. Create a TTL of 15 seconds to display the values of Employees.



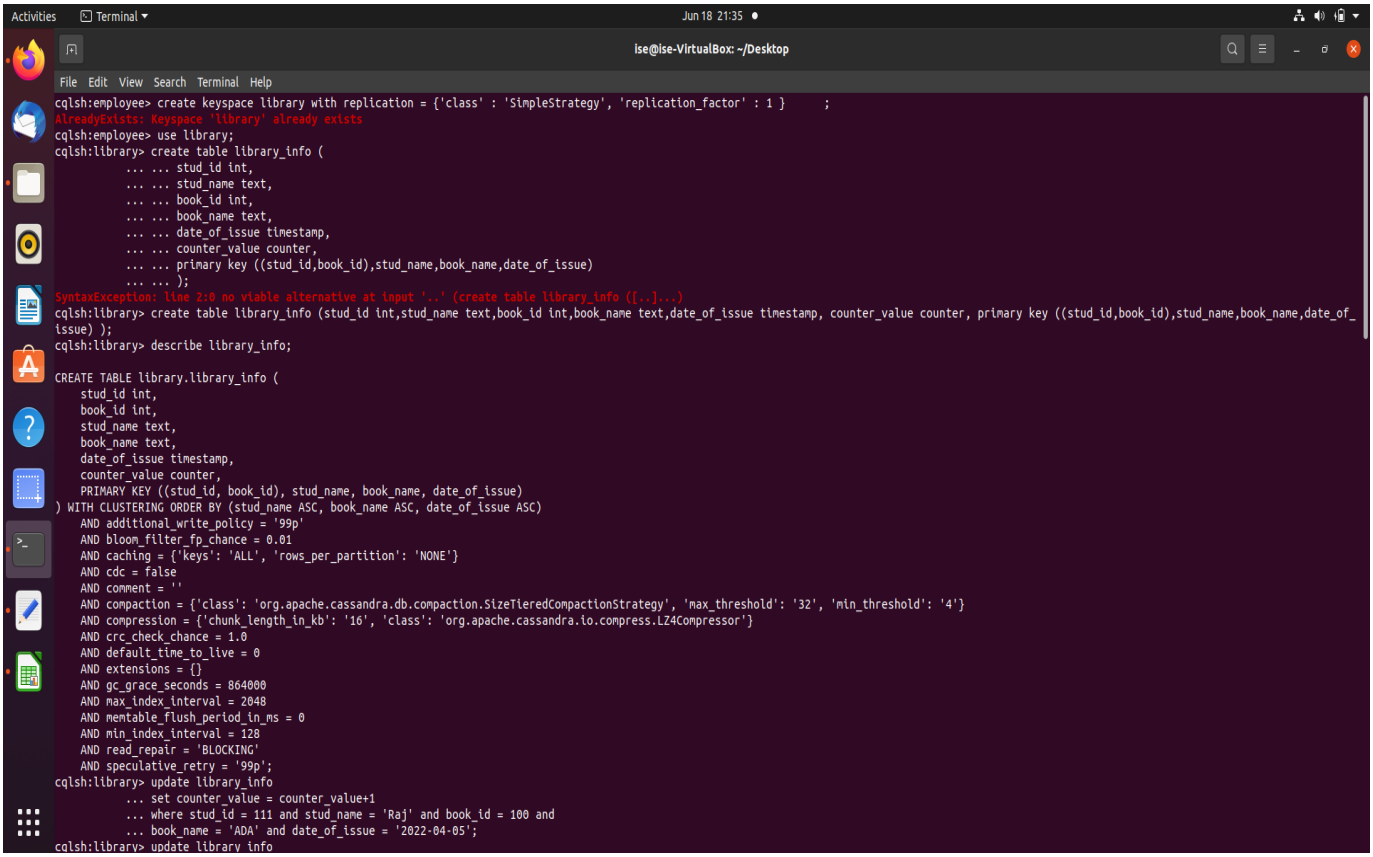


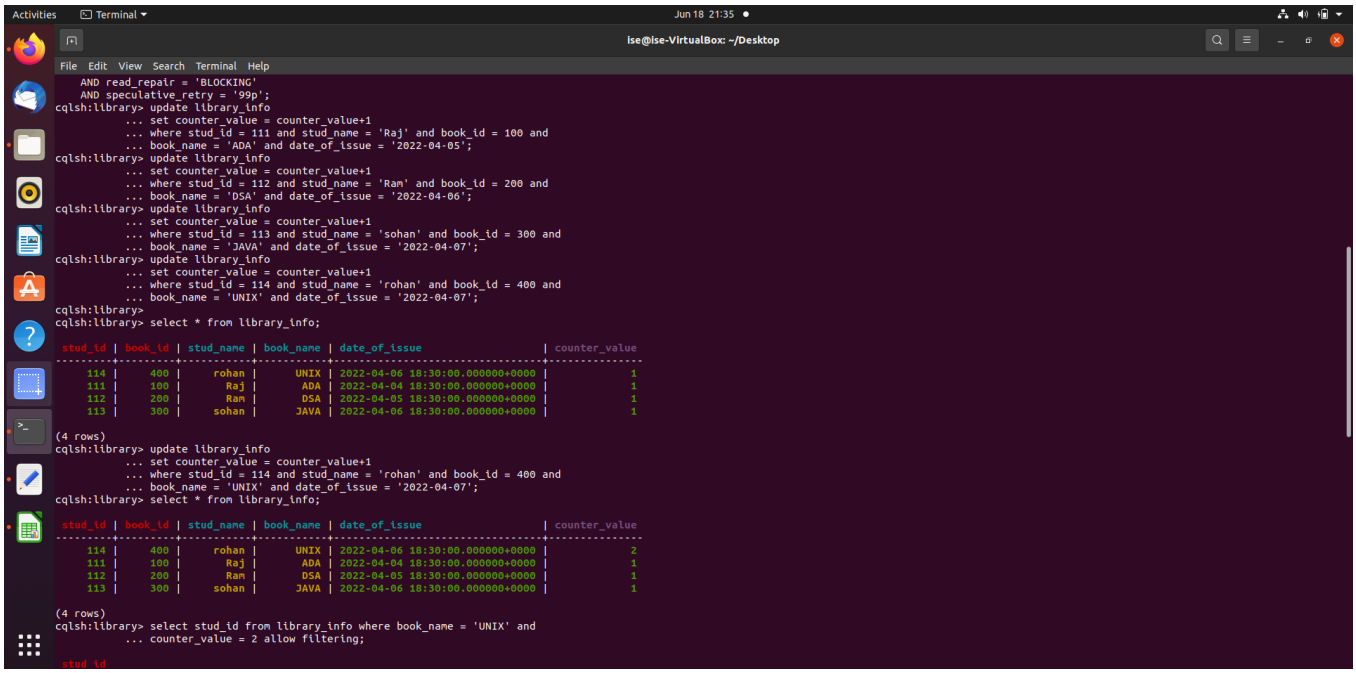

Activities
Terminal
Jun 18 21:26
ise@ise-VirtualBox: ~/Desktop
151 | 95000 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Yelena
(5 rows)
cqlsh:employee> update Employee_info set project = project+{'AI','Data
... warehouse'} where Emp_id = 111 and salary = 75000;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name project in table employee.employee_info"
cqlsh:employee> alter employee_info add project set<text>;
SyntaxException: line 1:6 no viable alternative at input 'employee_info' ([alter] employee_info...)
cqlsh:employee> alter table employee_info add project set<text>;
cqlsh:employee> update Employee_info set project = project+{'AI','Data warehouse'} where Emp_id = 111 and salary = 75000;
cqlsh:employee> update Employee_info set project = project+{'IOT','Data
... warehouse'} where Emp_id = 121 and salary = 85000;
cqlsh:employee> select * from employee_info;
emp_id | salary | dept_name | designation | doj | emp_name | project
111 | 75000 | CSE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | John | {'AI', 'Data warehouse'}
151 | 95000 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Yelena | null
121 | 85000 | ECE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | Josh | {'Data\nwarehouse', 'IOT'}
141 | 1.05e+05 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Jane | null
131 | 95000 | ECE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Mary | null
(5 rows)
cqlsh:employee> update Employee_info set project = project+{'IOT','AI'} where
... Emp_id = 131 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','machine
... learning'} where Emp_id = 141 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','data science'}
... where Emp_id = 141 and salary = 105000;
cqlsh:employee> cqlsh:employee> select * from Employee_info;
SyntaxException: line 1:0 no viable alternative at input 'cqlsh' ([cqlsh]...)
cqlsh:employee> select * from Employee_info;
emp_id | salary | dept_name | designation | doj | emp_name | project
111 | 75000 | CSE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | John | {'AI', 'Data warehouse'}
151 | 95000 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Yelena | null
121 | 85000 | ECE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | Josh | {'Data\nwarehouse', 'IOT'}
141 | 95000 | null | null | null | null | {'IOT', 'machine\nlearning'}
141 | 1.05e+05 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Jane | {'IOT', 'data science'}
131 | 95000 | ECE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Mary | {'AI', 'IOT'}
(6 rows)
cqlsh:employee> update Employee_info set project = project+{'IOT','AI'} where
... Emp_id = 131 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','machine
... learning'} where Emp_id = 141 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','data science'}

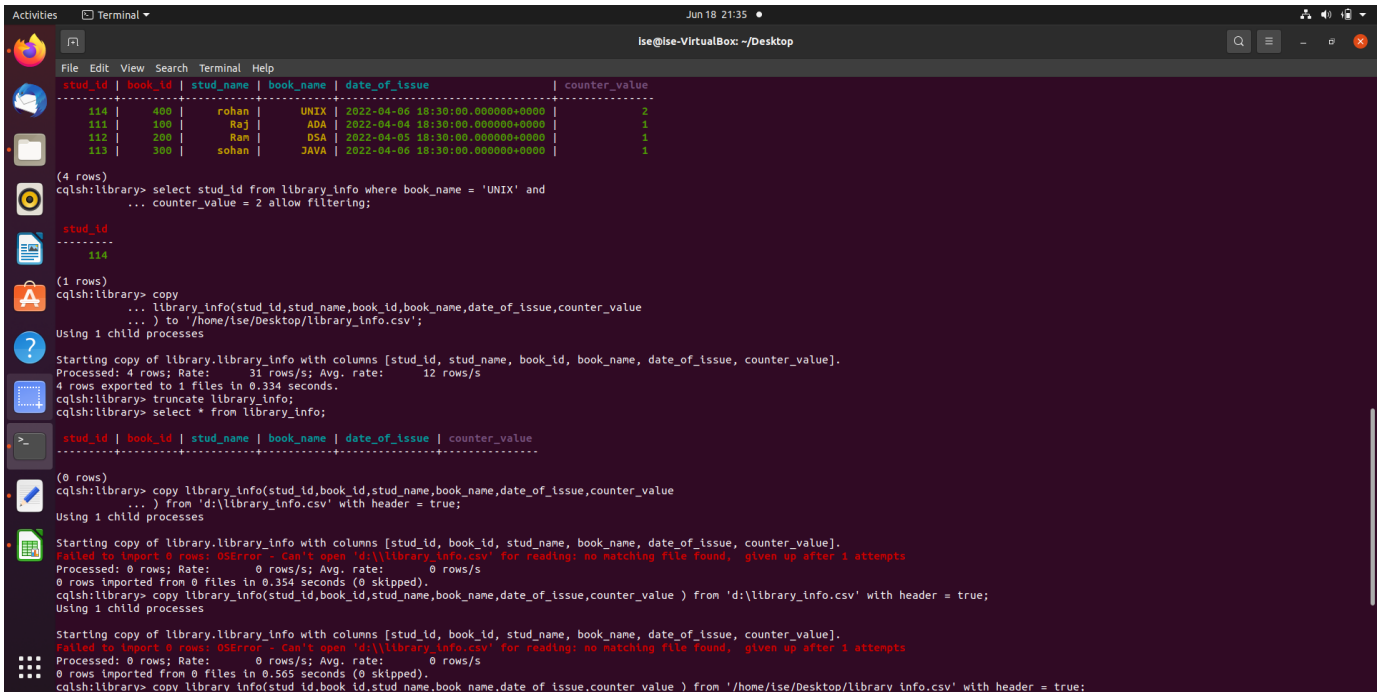

Activities
Terminal
Jun 18 21:26
ise@ise-VirtualBox: ~/Desktop
141 | 1.05e+05 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Jane | {'IOT', 'data science'}
131 | 95000 | ECE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Mary | {'AI', 'IOT'}
(6 rows)
cqlsh:employee> update Employee_info set project = project+{'IOT','AI'} where
... Emp_id = 131 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','machine
... learning'} where Emp_id = 141 and salary = 95000;
cqlsh:employee> update Employee_info set project = project+{'IOT','data science'}
... where Emp_id = 141 and salary = 105000;
cqlsh:employee> select * from Employee_info;
emp_id | salary | dept_name | designation | doj | emp_name | project
111 | 75000 | CSE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | John | {'AI', 'Data warehouse'}
151 | 95000 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Yelena | null
121 | 85000 | ECE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | Josh | {'Data\nwarehouse', 'IOT'}
141 | 95000 | null | null | null | null | {'IOT', 'machine\nlearning'}
141 | 1.05e+05 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Jane | {'IOT', 'data science'}
131 | 95000 | ECE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Mary | {'AI', 'IOT'}
(6 rows)
cqlsh:employee> update Employee_info set project = project+{'IOT','AI'} where
... Emp_id = 151 and salary = 95000;
cqlsh:employee> select * from Employee_info;
emp_id | salary | dept_name | designation | doj | emp_name | project
111 | 75000 | CSE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | John | {'AI', 'Data warehouse'}
151 | 95000 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Yelena | {'AI', 'IOT'}
121 | 85000 | ECE | Assistant professor | 2022-05-10 18:30:00.000000+0000 | Josh | {'Data\nwarehouse', 'IOT'}
141 | 95000 | null | null | null | null | {'IOT', 'machine\nlearning'}
141 | 1.05e+05 | ISE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Jane | {'IOT', 'data science'}
131 | 95000 | ECE | Associate professor | 2022-05-10 18:30:00.000000+0000 | Mary | {'AI', 'IOT'}
(6 rows)
cqlsh:employee> insert into
... Employee_info(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) values
... (161,'Ryan','Associate professor','2022-05-11',95000,'ISE') using ttl 60;
cqlsh:employee> select ttl(Emp_name) from Employee_info where Emp_id = 161
... and salary = 95000;
ttl(emp_name)
52
(1 rows)
cqlsh:employee>

LAB 2 Perform the following DB operations using Cassandra.

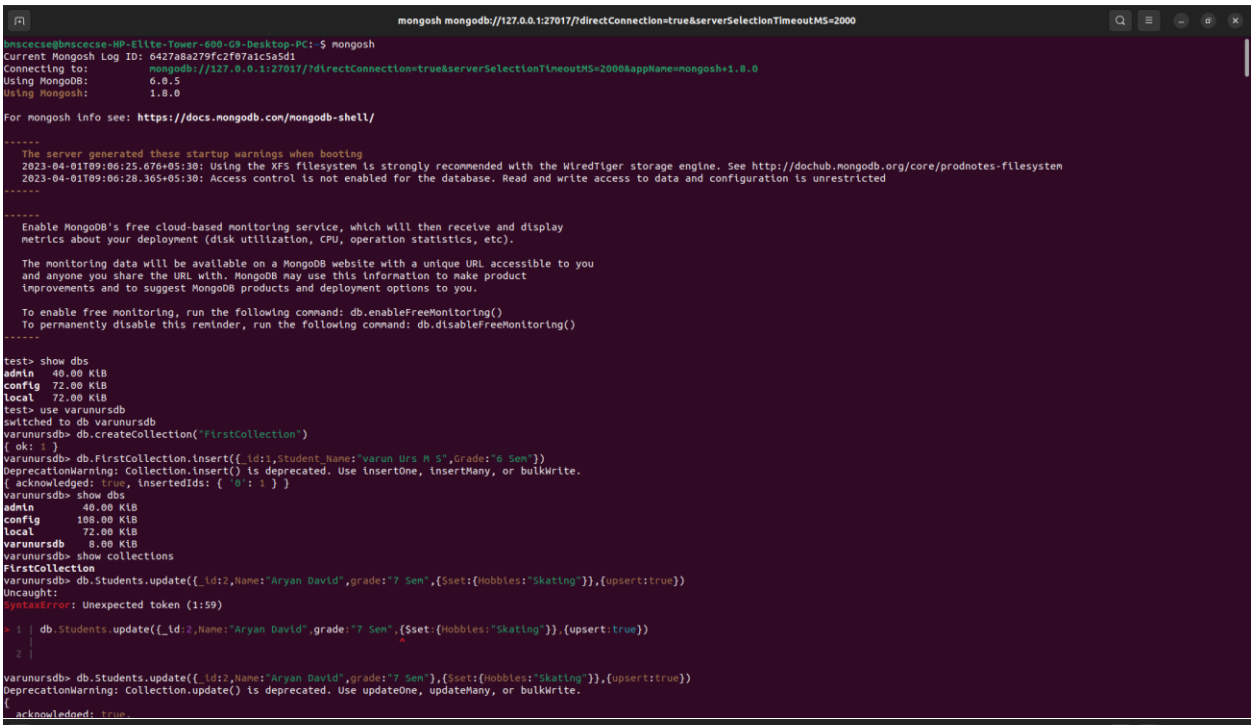
1. Create a keyspace by name Library
2. Create a column family by name Library-Info with attributes Stud_Id Primary Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue
3. Insert the values into the table in batch
4. Display the details of the table created and increase the value of the counter
5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.
6. Export the created column to a csv file
7. Import a given csv dataset from local file system into Cassandra column family

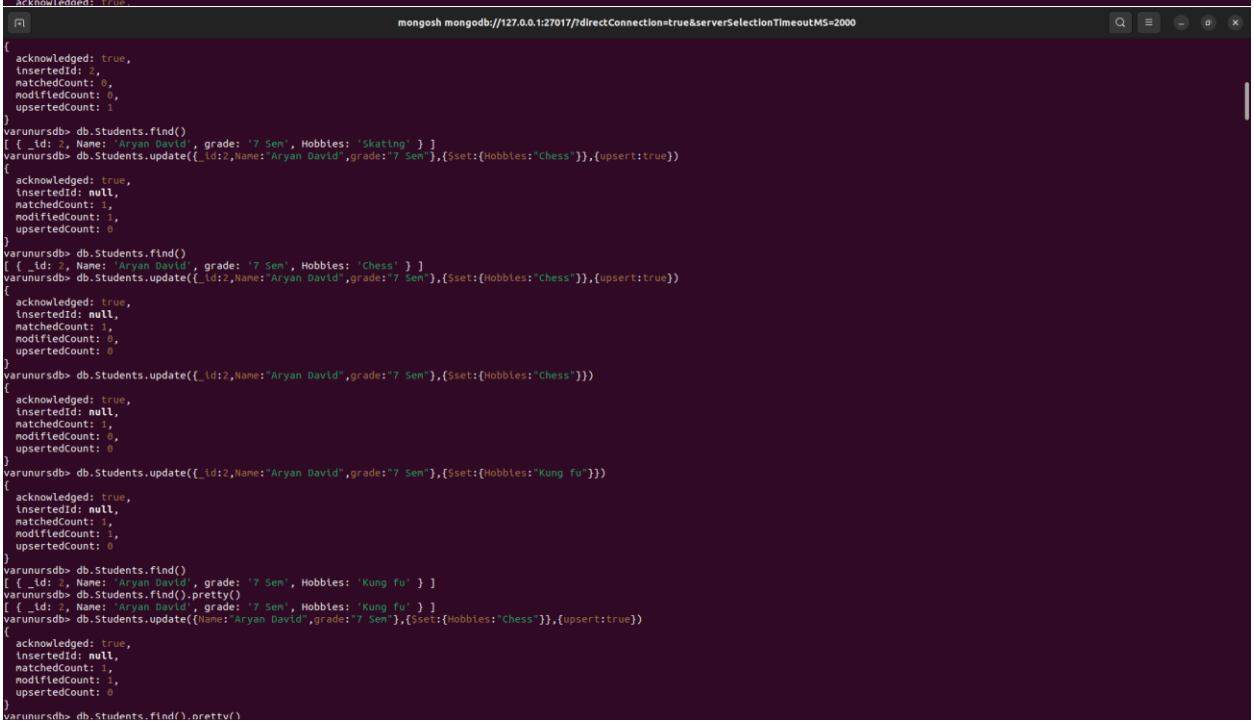


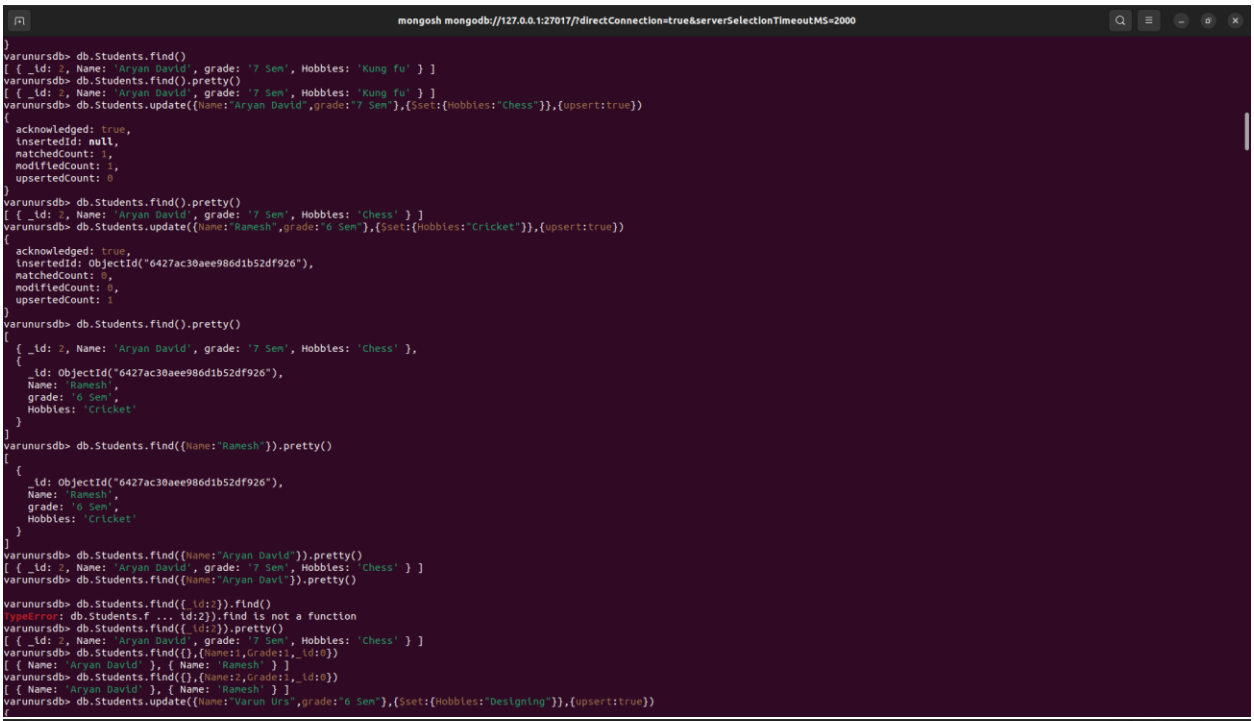

Activities
Terminal
Jun 18 21:35
ise@ise-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
AND read_repair = 'BLOCKING'
AND speculative_retry = '99p';
cqlsh:library> update library_info
... set counter_value = counter_value+1
... where stud_id = 111 and stud_name = 'Raj' and book_id = 100 and
... book_name = 'ADA' and date_of_issue = '2022-04-05';
cqlsh:library> update library_info
... set counter_value = counter_value+1
... where stud_id = 112 and stud_name = 'Ram' and book_id = 200 and
... book_name = 'DSA' and date_of_issue = '2022-04-06';
cqlsh:library> update library_info
... set counter_value = counter_value+1
... where stud_id = 113 and stud_name = 'sohan' and book_id = 300 and
... book_name = 'JAVA' and date_of_issue = '2022-04-07';
cqlsh:library> update library_info
... set counter_value = counter_value+1
... where stud_id = 114 and stud_name = 'rohan' and book_id = 400 and
... book_name = 'UNIX' and date_of_issue = '2022-04-07';
cqlsh:library>
cqlsh:library> select * from library_info;
stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
114 | 400 | rohan | UNIX | 2022-04-06 18:30:00.000000+0000 | 1
111 | 100 | Raj | ADA | 2022-04-04 18:30:00.000000+0000 | 1
112 | 200 | Ram | DSA | 2022-04-05 18:30:00.000000+0000 | 1
113 | 300 | sohan | JAVA | 2022-04-06 18:30:00.000000+0000 | 1
(4 rows)
cqlsh:library> update library_info
... set counter_value = counter_value+1
... where stud_id = 114 and stud_name = 'rohan' and book_id = 400 and
... book_name = 'UNIX' and date_of_issue = '2022-04-07';
cqlsh:library> select * from library_info;
stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
114 | 400 | rohan | UNIX | 2022-04-06 18:30:00.000000+0000 | 2
111 | 100 | Raj | ADA | 2022-04-04 18:30:00.000000+0000 | 1
112 | 200 | Ram | DSA | 2022-04-05 18:30:00.000000+0000 | 1
113 | 300 | sohan | JAVA | 2022-04-06 18:30:00.000000+0000 | 1
(4 rows)
cqlsh:library> select stud_id from library_info where book_name = 'UNIX' and
... counter_value = 2 allow filtering;
stud_id

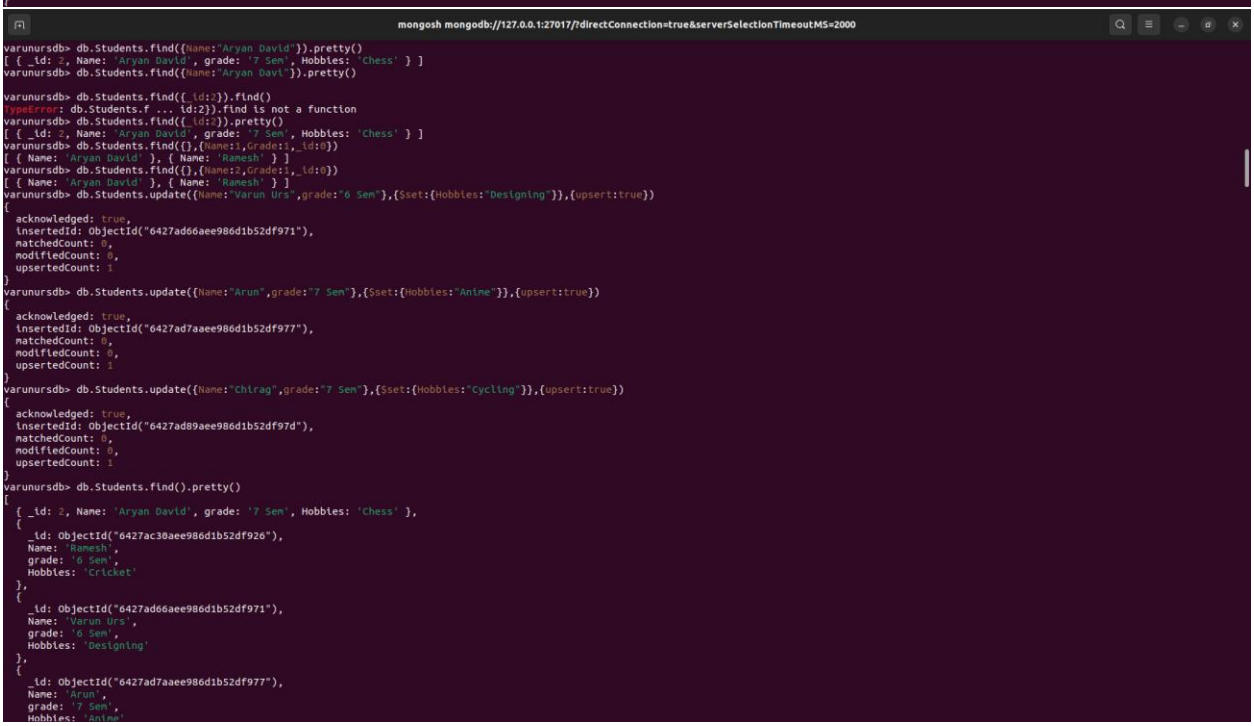

Activities
Terminal
Jun 18 21:35
ise@ise-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
114 | 400 | rohan | UNIX | 2022-04-06 18:30:00.000000+0000 | 2
111 | 100 | Raj | ADA | 2022-04-04 18:30:00.000000+0000 | 1
112 | 200 | Ram | DSA | 2022-04-05 18:30:00.000000+0000 | 1
113 | 300 | sohan | JAVA | 2022-04-06 18:30:00.000000+0000 | 1
(4 rows)
cqlsh:library> select stud_id from library_info where book_name = 'UNIX' and
... counter_value = 2 allow filtering;
stud_id
114
(1 rows)
cqlsh:library> copy
... library_info(stud_id,stud_name,book_id,book_name,date_of_issue,counter_value
...) to '/home/ise/Desktop/library_info.csv';
Using 1 child processes
Starting copy of library.library_info with columns [stud_id, stud_name, book_id, book_name, date_of_issue, counter_value].
Processed: 4 rows; Rate: 31 rows/s; Avg. rate: 12 rows/s
4 rows exported to 1 files in 0.334 seconds.
cqlsh:library> truncate library_info;
cqlsh:library> select * from library_info;
stud_id | book_id | stud_name | book_name | date_of_issue | counter_value
(0 rows)
cqlsh:library> copy library_info(stud_id,book_id,stud_name,book_name,date_of_issue,counter_value
...) from 'd:\library_info.csv' with header = true;
Using 1 child processes
Starting copy of library.library_info with columns [stud_id, book_id, stud_name, book_name, date_of_issue, counter_value].
Failed to import 0 rows: OSError - Can't open 'd:\\library_info.csv' for reading: no matching file found, given up after 1 attempts
Processed: 0 rows; Rate: 0 rows/s; Avg. rate: 0 rows/s
0 rows imported from 0 files in 0.354 seconds (0 skipped).
cqlsh:library> copy library_info(stud_id,book_id,stud_name,book_name,date_of_issue,counter_value) from 'd:\library_info.csv' with header = true;
Using 1 child processes
Starting copy of library.library_info with columns [stud_id, book_id, stud_name, book_name, date_of_issue, counter_value].
Failed to import 0 rows: OSError - Can't open 'd:\\library_info.csv' for reading: no matching file found, given up after 1 attempts
Processed: 0 rows; Rate: 0 rows/s; Avg. rate: 0 rows/s
0 rows imported from 0 files in 0.565 seconds (0 skipped).
cqlsh:library> copy library_info(stud_id,book_id,stud_name,book_name,date_of_issue,counter_value) from '/home/ise/Desktop/library_info.csv' with header = true;

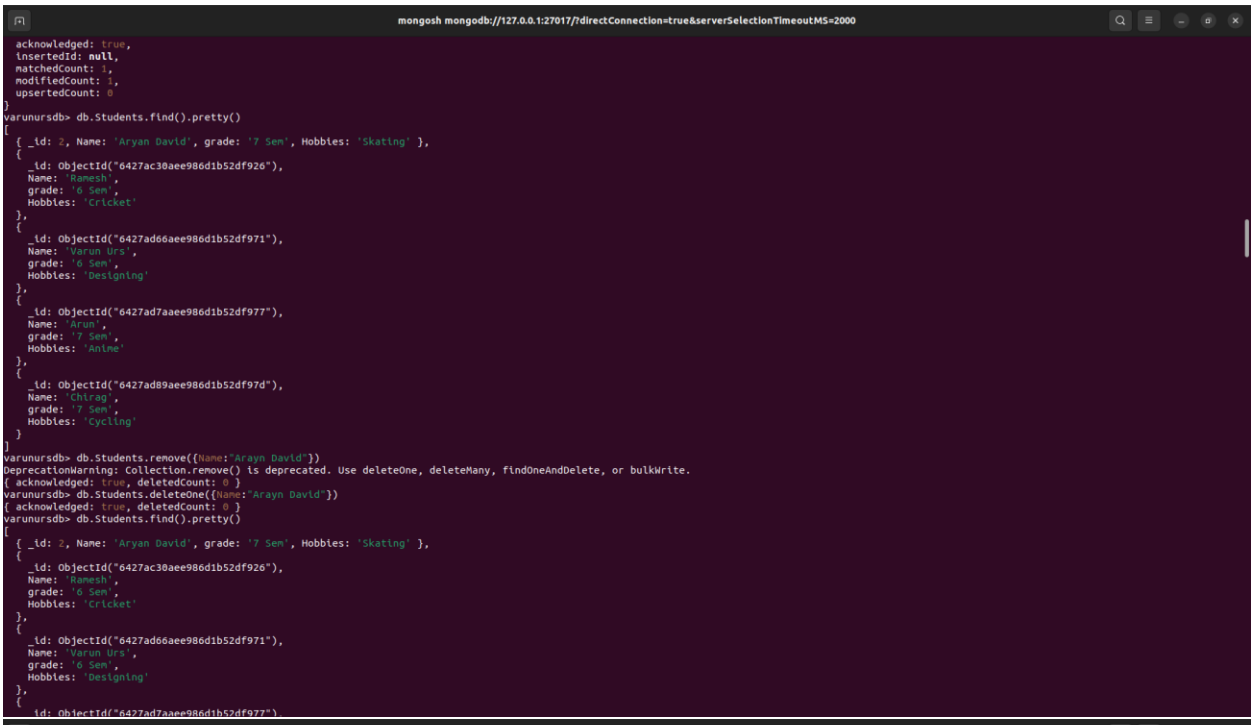
LAB 3 MongoDB- CRUD Demonstration

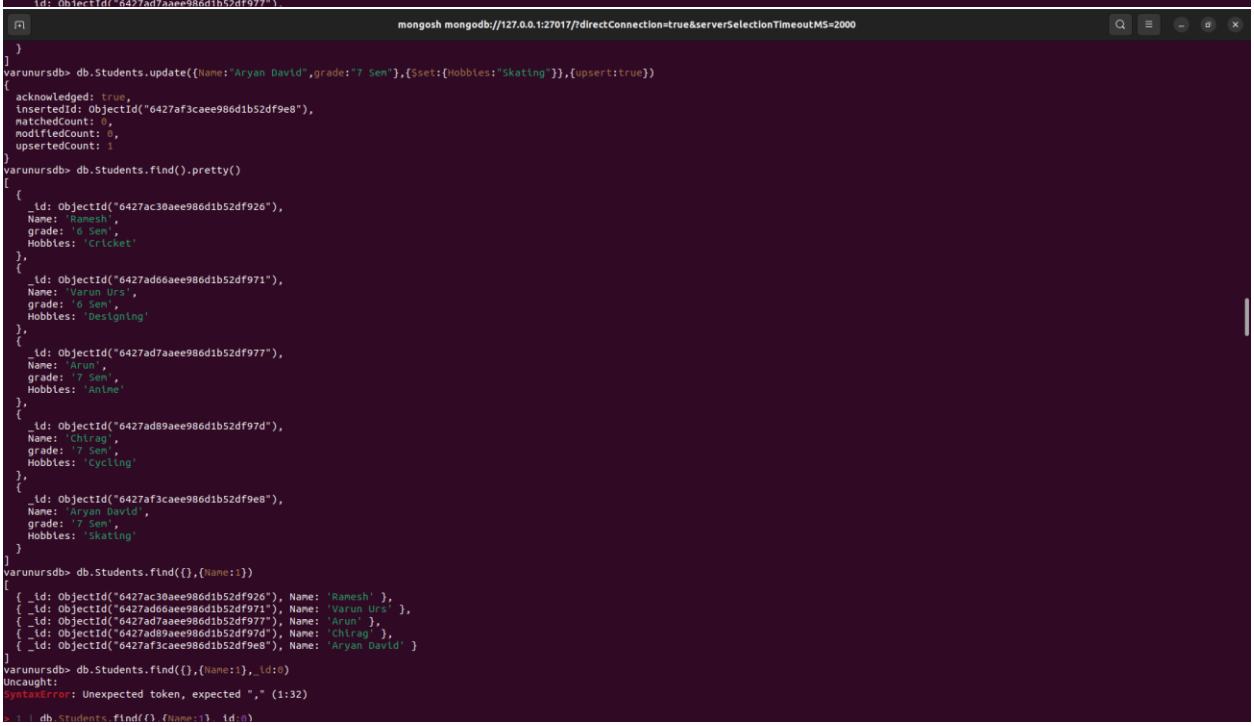


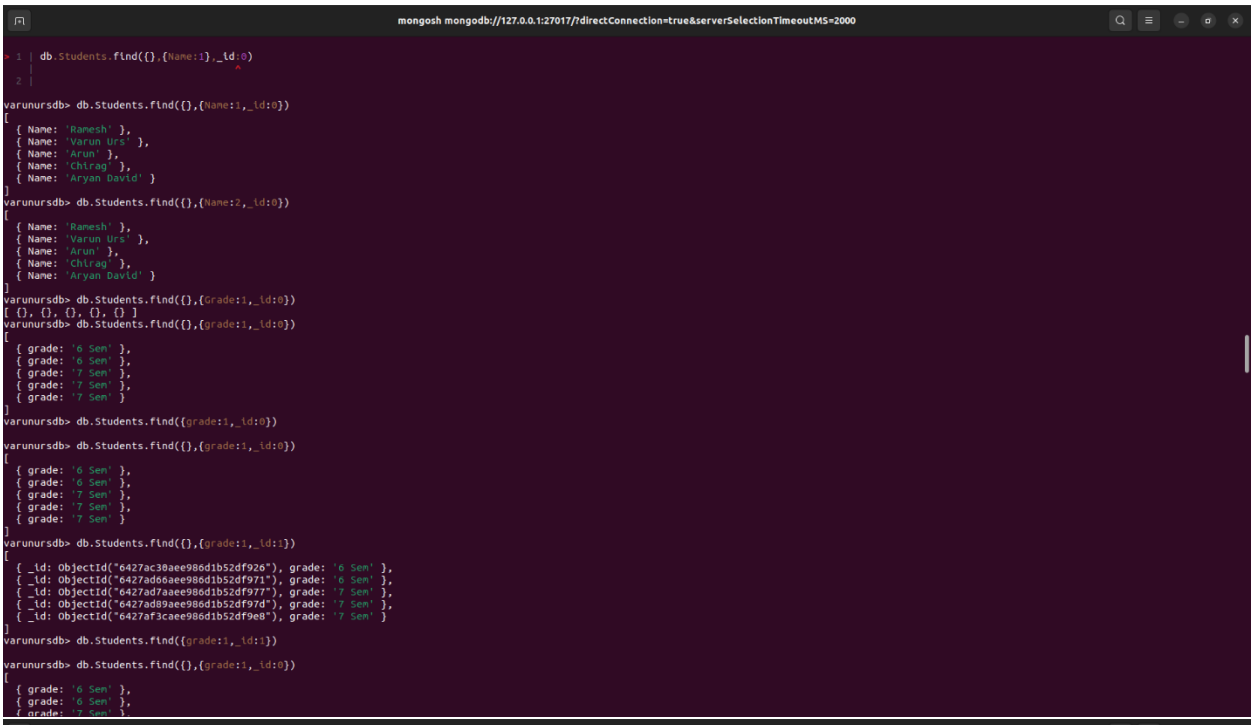


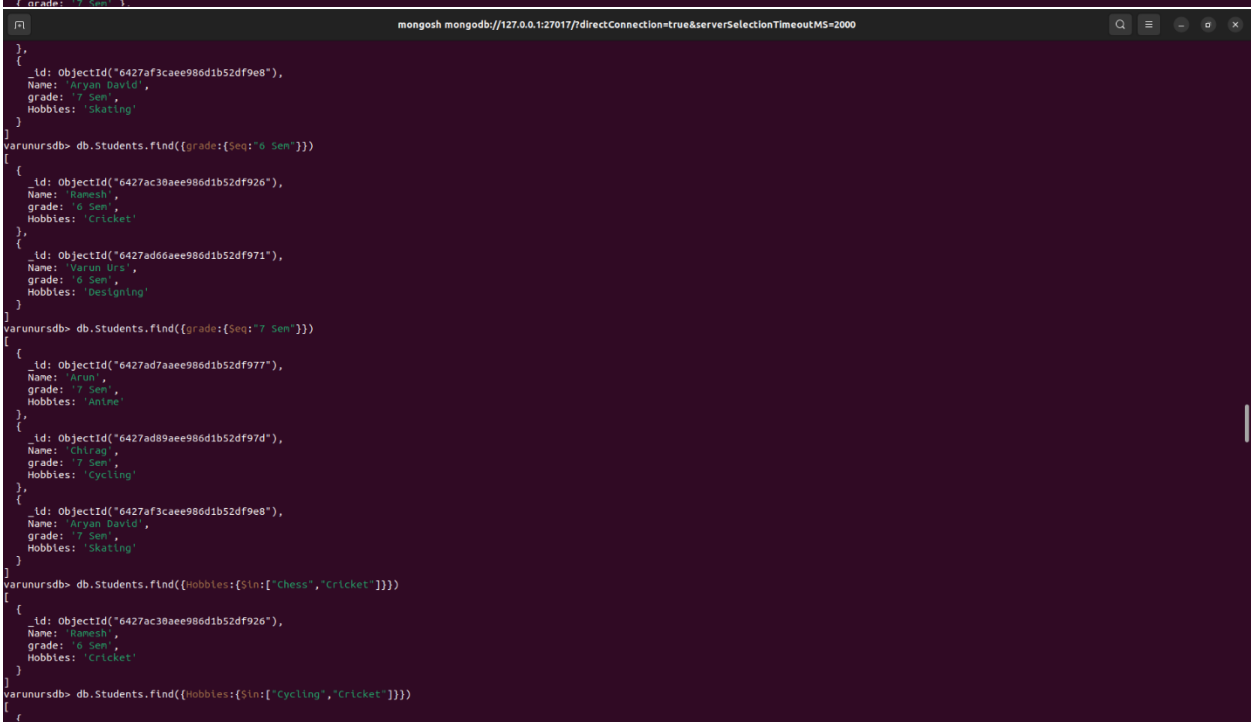

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
}
varunursdb> db.Students.find()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Kung fu' }]
varunursdb> db.Students.find().pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Kung fu' }]
varunursdb> db.Students.update({Name:"Aryan David",grade:"7 Sem"},{$set:{Hobbies:"Chess"}},{upsert:true})
{
acknowledged: true,
insertedId: null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
}
varunursdb> db.Students.find().pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' }]
varunursdb> db.Students.update({Name:"Ramesh",grade:"6 Sem"},{$set:{Hobbies:"Cricket"}},{upsert:true})
{
acknowledged: true,
insertedId: ObjectId("6427ac30aee986d1b52df926"),
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
}
varunursdb> db.Students.find().pretty()
[
{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' },
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
}
]
varunursdb> db.Students.find({Name:"Ramesh"}).pretty()
[
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
}
]
varunursdb> db.Students.find({Name:"Aryan David"}).pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' }]
varunursdb> db.Students.find({Name:"Aryan Davi"}).pretty()
varunursdb> db.Students.find({_id:2}).find()
TypeError: db.Students.f ... id:2}).find is not a function
varunursdb> db.Students.find({_id:2}).pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' }]
varunursdb> db.Students.find({},{Name:1,Grade:1,_id:0})
[{ Name: 'Aryan David' }, { Name: 'Ramesh' }]
varunursdb> db.Students.find({},{Name:2,Grade:1,_id:0})
[{ Name: 'Aryan David' }, { Name: 'Ramesh' }]
varunursdb> db.Students.update({Name:"Varun Urs",grade:"6 Sem"},{$set:{Hobbies:"Designing"}},{upsert:true})

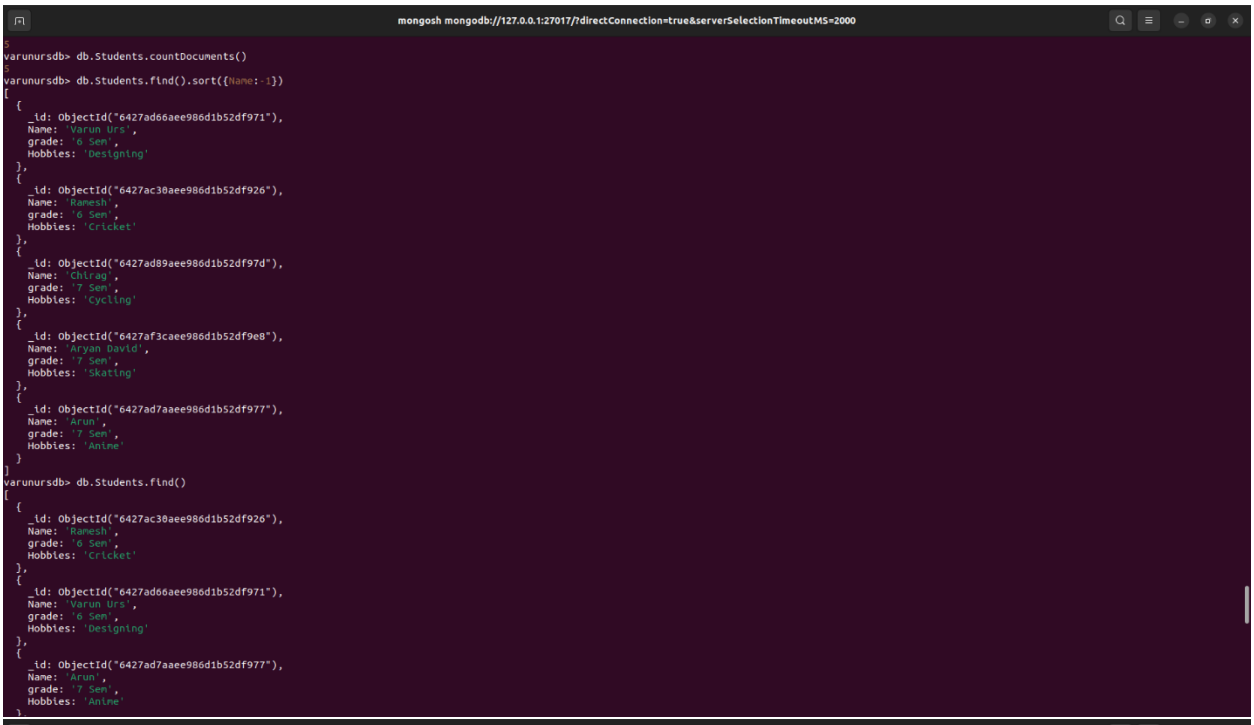

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
varunursdb> db.Students.find({Name:"Aryan David"}).pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' }]
varunursdb> db.Students.find({Name:"Aryan Davi"}).pretty()
varunursdb> db.Students.find({_id:2}).find()
TypeError: db.Students.f ... id:2}).find is not a function
varunursdb> db.Students.find({_id:2}).pretty()
[{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' }]
varunursdb> db.Students.find({},{Name:1,Grade:1,_id:0})
[{ Name: 'Aryan David' }, { Name: 'Ramesh' }]
varunursdb> db.Students.find({},{Name:2,Grade:1,_id:0})
[{ Name: 'Aryan David' }, { Name: 'Ramesh' }]
varunursdb> db.Students.update({Name:"Varun Urs",grade:"6 Sem"},{$set:{Hobbies:"Designing"}},{upsert:true})
{
acknowledged: true,
insertedId: ObjectId("6427ad66aee986d1b52df971"),
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
}
varunursdb> db.Students.update({Name:"Arun",grade:"7 Sem"},{$set:{Hobbies:"Anime"}},{upsert:true})
{
acknowledged: true,
insertedId: ObjectId("6427ad7aaee986d1b52df977"),
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
}
varunursdb> db.Students.update({Name:"Chirag",grade:"7 Sem"},{$set:{Hobbies:"Cycling"}},{upsert:true})
{
acknowledged: true,
insertedId: ObjectId("6427ad89aee986d1b52df97d"),
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
}
varunursdb> db.Students.find().pretty()
[
{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Chess' },
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
},
{
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
},
{
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'

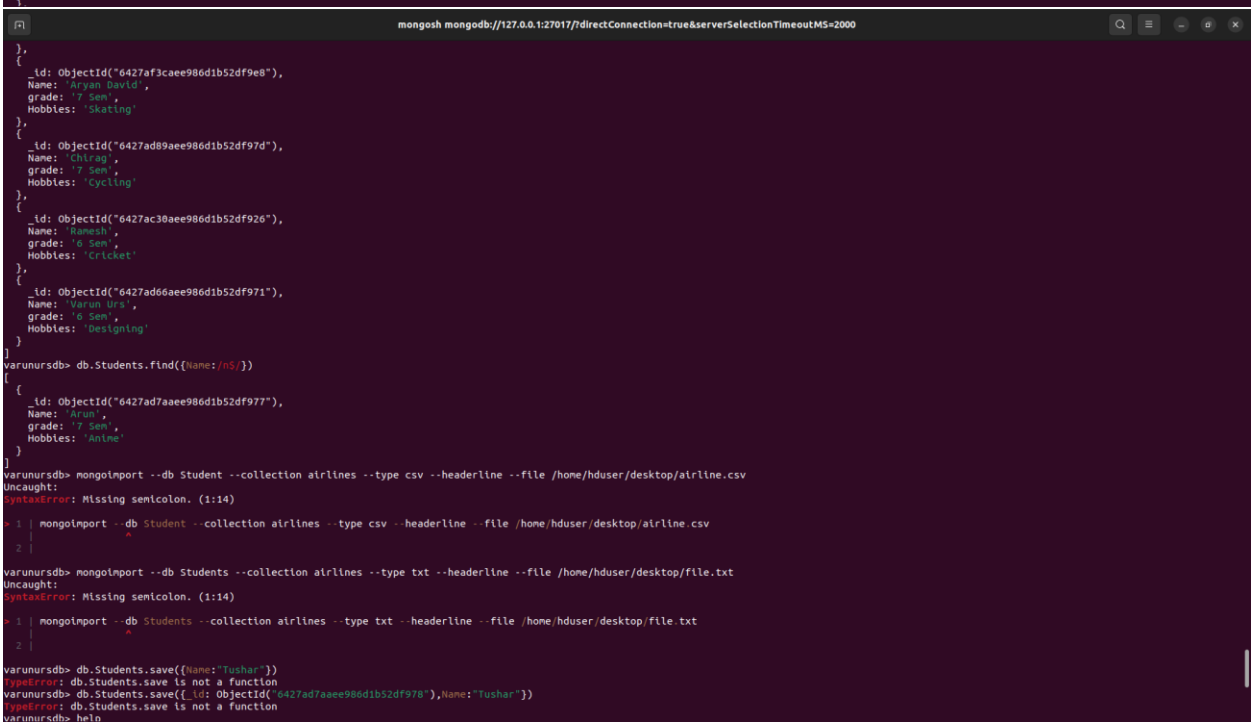

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
acknowledged: true,
insertedId: null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
varunursdb> db.Students.find().pretty()
{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Skating' },
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
_id: ObjectId("6427ad89aee986d1b52df97d"),
Name: 'Chirag',
grade: '7 Sem',
Hobbies: 'Cycling'
varunursdb> db.Students.remove({Name:"Arayn David"})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 0 }
varunursdb> db.Students.deleteOne({Name:"Arayn David"})
{ acknowledged: true, deletedCount: 0 }
varunursdb> db.Students.find().pretty()
{ _id: 2, Name: 'Aryan David', grade: '7 Sem', Hobbies: 'Skating' },
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'


mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
varunursdb> db.Students.update({Name:"Aryan David",grade:"7 Sem"},{$set:{Hobbies:"Skating"}},{upsert:true})
acknowledged: true,
insertedId: ObjectId("6427af3caee986d1b52df9e8"),
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
varunursdb> db.Students.find().pretty()
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
_id: ObjectId("6427ad89aee986d1b52df97d"),
Name: 'Chirag',
grade: '7 Sem',
Hobbies: 'Cycling'
_id: ObjectId("6427af3caee986d1b52df9e8"),
Name: 'Aryan David',
grade: '7 Sem',
Hobbies: 'Skating'
varunursdb> db.Students.find({},{Name:1})
{ _id: ObjectId("6427ac30aee986d1b52df926"), Name: 'Ramesh' },
{ _id: ObjectId("6427ad66aee986d1b52df971"), Name: 'Varun Urs' },
{ _id: ObjectId("6427ad7aaee986d1b52df977"), Name: 'Arun' },
{ _id: ObjectId("6427ad89aee986d1b52df97d"), Name: 'Chirag' },
{ _id: ObjectId("6427af3caee986d1b52df9e8"), Name: 'Aryan David' }
varunursdb> db.Students.find({},{Name:1},_id:0)
Uncaught:
SyntaxError: Unexpected token, expected "," (1:32)
1 | db.Students.find({},{Name:1},_id:0)

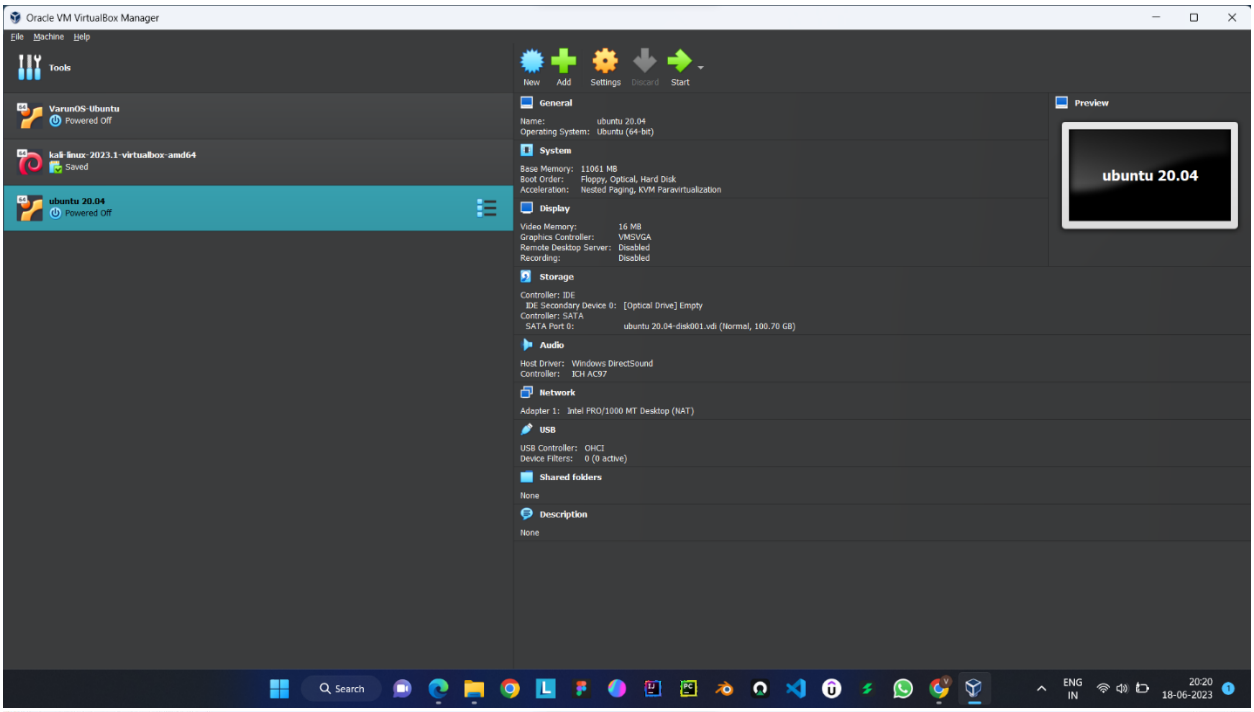

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
1 | db.Students.find({},{Name:1},_id:0)
2 |
varunursdb> db.Students.find({},{Name:1,_id:0})
{ Name: 'Ramesh' },
{ Name: 'Varun Urs' },
{ Name: 'Arun' },
{ Name: 'Chirag' },
{ Name: 'Aryan David' }
varunursdb> db.Students.find({},{Name:2,_id:0})
{ Name: 'Ramesh' },
{ Name: 'Varun Urs' },
{ Name: 'Arun' },
{ Name: 'Chirag' },
{ Name: 'Aryan David' }
varunursdb> db.Students.find({},{Grade:1,_id:0})
[{}, {}, {}, {}, {}]
varunursdb> db.Students.find({},{grade:1,_id:0})
{ grade: '6 Sem' },
{ grade: '6 Sem' },
{ grade: '7 Sem' },
{ grade: '7 Sem' },
{ grade: '7 Sem' }
varunursdb> db.Students.find({grade:1,_id:0})
varunursdb> db.Students.find({},{grade:1,_id:0})
{ grade: '6 Sem' },
{ grade: '6 Sem' },
{ grade: '7 Sem' },
{ grade: '7 Sem' },
{ grade: '7 Sem' }
varunursdb> db.Students.find({},{grade:1,_id:1})
{ _id: ObjectId("6427ac30aee986d1b52df926"), grade: '6 Sem' },
{ _id: ObjectId("6427ad66aee986d1b52df971"), grade: '6 Sem' },
{ _id: ObjectId("6427ad7aaee986d1b52df977"), grade: '7 Sem' },
{ _id: ObjectId("6427ad89aee986d1b52df97d"), grade: '7 Sem' },
{ _id: ObjectId("6427af3caee986d1b52df9e8"), grade: '7 Sem' }
varunursdb> db.Students.find({grade:1,_id:1})
varunursdb> db.Students.find({},{grade:1,_id:0})
{ grade: '6 Sem' },
{ grade: '6 Sem' },

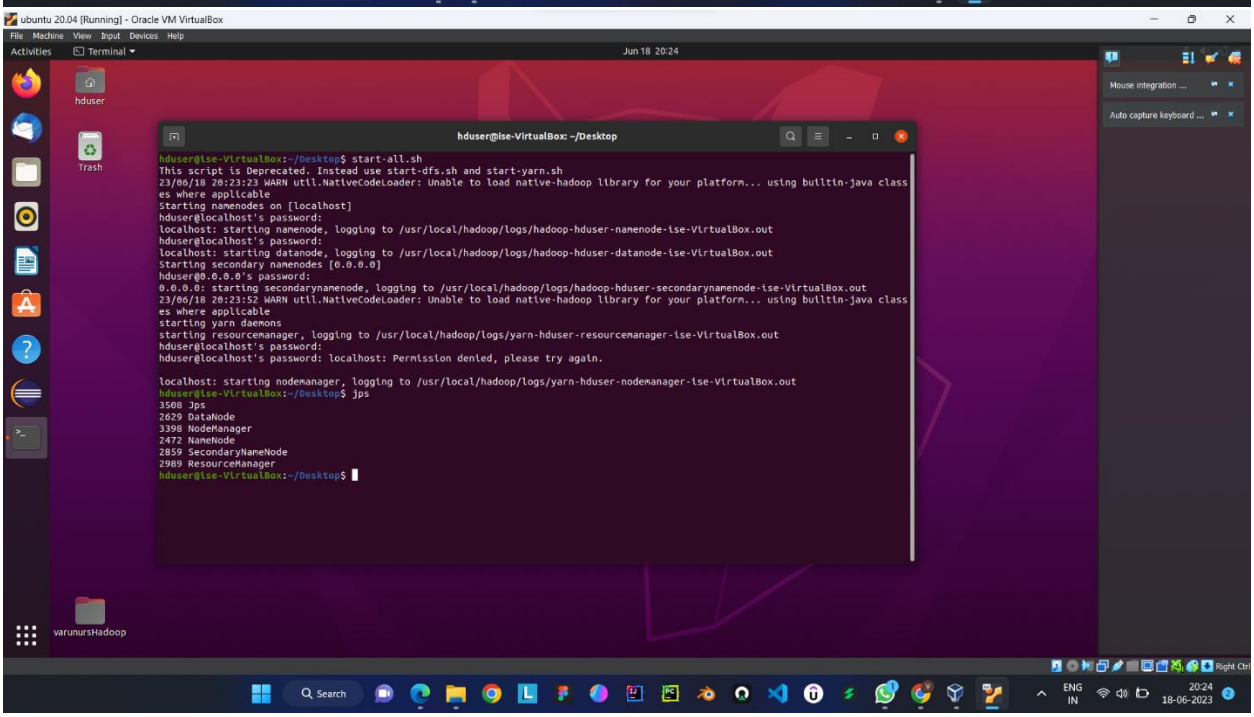

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
},
{
_id: ObjectId("6427af3caee986d1b52df9e8"),
Name: 'Aryan David',
grade: '7 Sem',
Hobbies: 'Skating'
}
]
varunursdb> db.Students.find({grade:{$eq:"6 Sem"}})
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
},
{
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
}
]
varunursdb> db.Students.find({grade:{$eq:"7 Sem"}})
{
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
},
{
_id: ObjectId("6427ad89aee986d1b52df97d"),
Name: 'Chirag',
grade: '7 Sem',
Hobbies: 'Cycling'
},
{
_id: ObjectId("6427af3caee986d1b52df9e8"),
Name: 'Aryan David',
grade: '7 Sem',
Hobbies: 'Skating'
}
]
varunursdb> db.Students.find({Hobbies:{$in:["Chess","Cricket"]}})
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
}
]
varunursdb> db.Students.find({Hobbies:{$in:["Cycling","Cricket"]}})
[
{


mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
5
varunursdb> db.Students.countDocuments()
5
varunursdb> db.Students.find().sort({Name:-1})
[
{
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
},
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
},
{
_id: ObjectId("6427ad89aee986d1b52df97d"),
Name: 'Chirag',
grade: '7 Sem',
Hobbies: 'Cycling'
},
{
_id: ObjectId("6427af3caee986d1b52df9e8"),
Name: 'Aryan David',
grade: '7 Sem',
Hobbies: 'Skating'
},
{
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
}
]
varunursdb> db.Students.find()
[
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
},
{
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
},
{
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
}

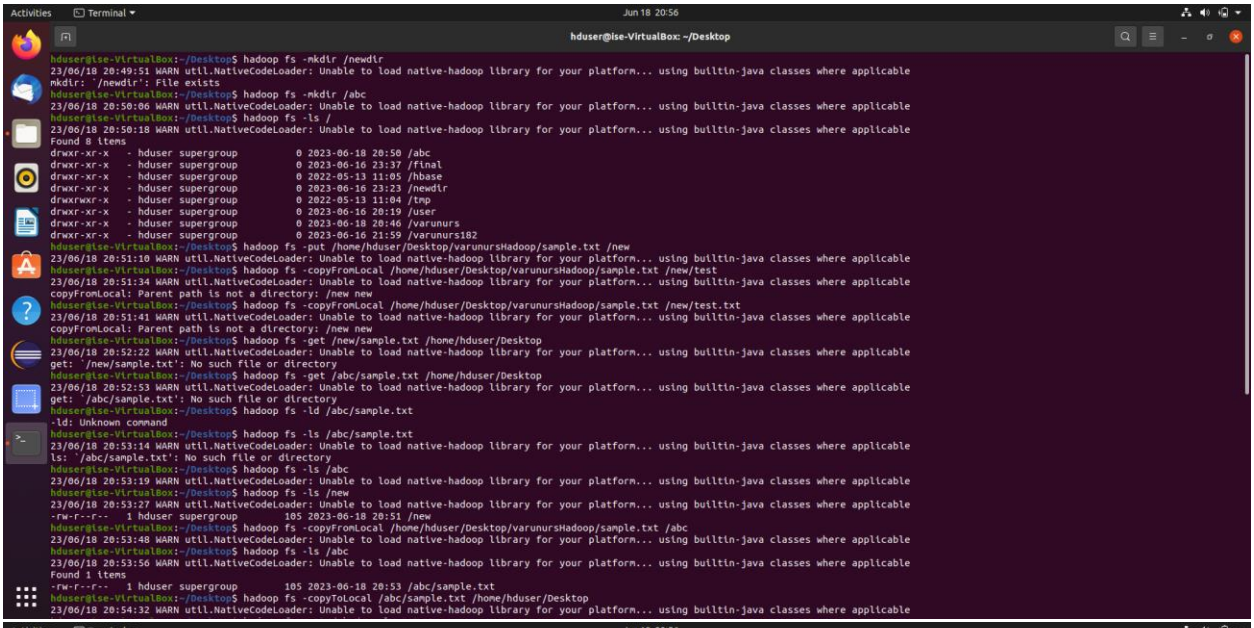

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
},
{
_id: ObjectId("6427af3caee986d1b52df9e8"),
Name: 'Aryan David',
grade: '7 Sem',
Hobbies: 'Skating'
},
{
_id: ObjectId("6427ad89aee986d1b52df97d"),
Name: 'Chirag',
grade: '7 Sem',
Hobbies: 'Cycling'
},
{
_id: ObjectId("6427ac30aee986d1b52df926"),
Name: 'Ramesh',
grade: '6 Sem',
Hobbies: 'Cricket'
},
{
_id: ObjectId("6427ad66aee986d1b52df971"),
Name: 'Varun Urs',
grade: '6 Sem',
Hobbies: 'Designing'
}
]
varunursdb> db.Students.find({Name:/nS/})
[
{
_id: ObjectId("6427ad7aaee986d1b52df977"),
Name: 'Arun',
grade: '7 Sem',
Hobbies: 'Anime'
}
]
varunursdb> mongoimport --db Student --collection airlines --type csv --headerline --file /home/hduser/desktop/airline.csv
Uncaught:
SyntaxError: Missing semicolon. (1:14)
> 1 | mongoimport --db Student --collection airlines --type csv --headerline --file /home/hduser/desktop/airline.csv
| ^
2 |
varunursdb> mongoimport --db Students --collection airlines --type txt --headerline --file /home/hduser/desktop/file.txt
Uncaught:
SyntaxError: Missing semicolon. (1:14)
> 1 | mongoimport --db Students --collection airlines --type txt --headerline --file /home/hduser/desktop/file.txt
| ^
2 |
varunursdb> db.Students.save({Name:"Tushar"})
TypeError: db.Students.save is not a function
varunursdb> db.Students.save({_id: ObjectId("6427ad7aaee986d1b52df978"),Name:"Tushar"})
TypeError: db.Students.save is not a function
varunursdb> help

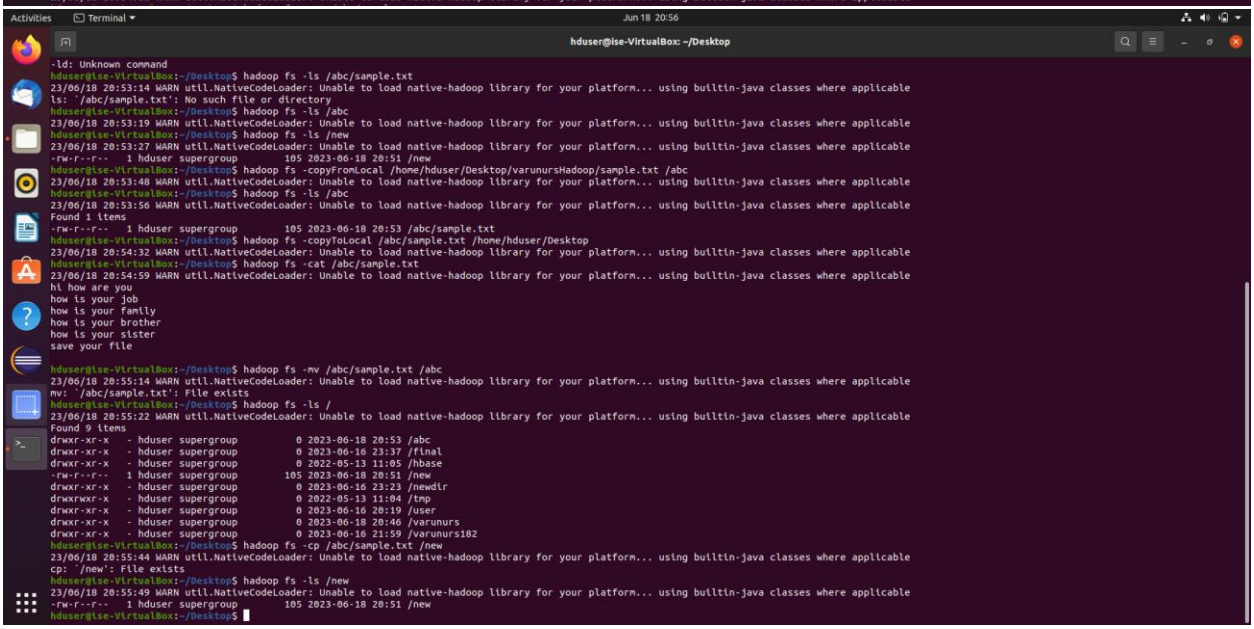
LAB 4. Screenshot of Hadoop installed





LAB 5 Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed).





LAB 6. From the following link extract the weather data
https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all

Create a Map Reduce program to

a) **find average temperature for each year from NCDC data set.**
b) **find the mean max temperature for every month**

Average Temperature

AverageDriver

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver {
   public static void main(String[] args) throws Exception {
      if (args.length != 2) {
        System.err.println("Please Enter the input and output parameters");
        System.exit(-1);
      }
      Job job = new Job();
      job.setJarByClass(AverageDriver.class);
      job.setJobName("Max temperature");
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      job.setMapperClass(AverageMapper.class);
      job.setReducerClass(AverageReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
   }
```

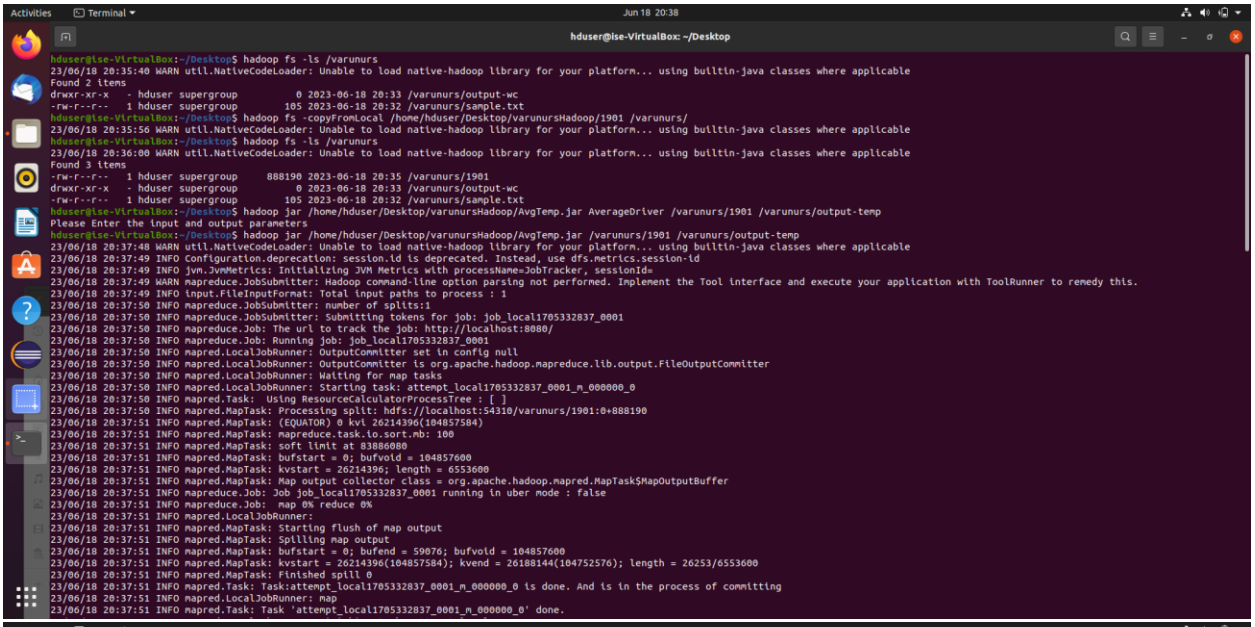
AverageMapper

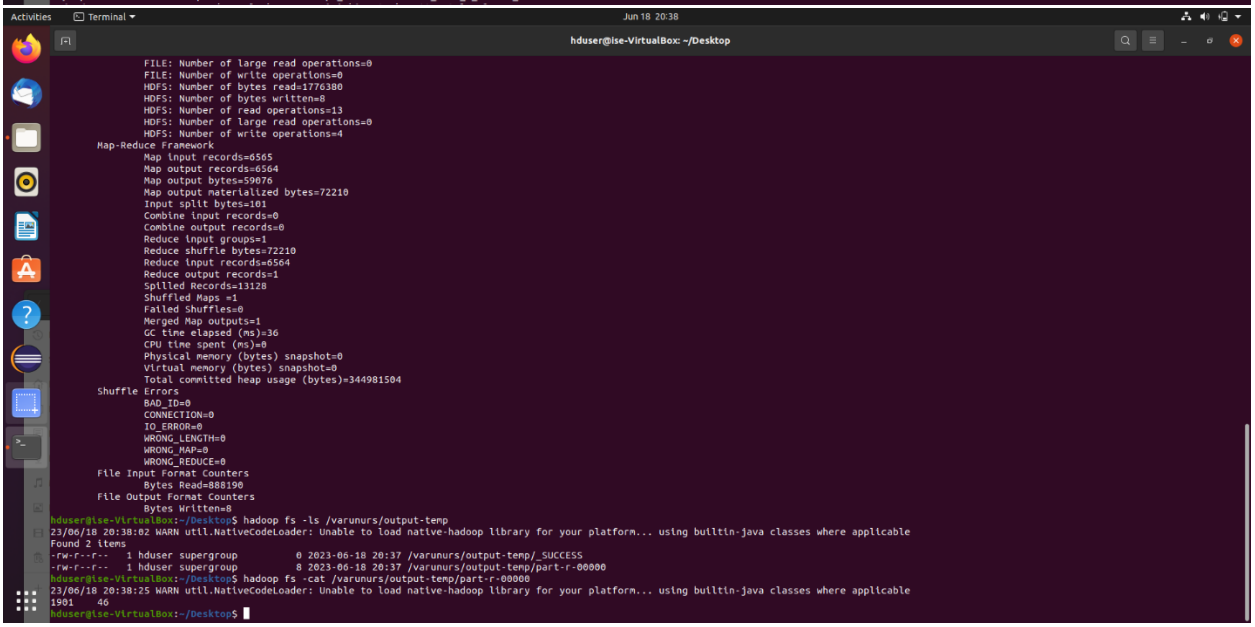
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
  public static final int MISSING = 9999;

  public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int temperature;
    String line = value.toString();
    String year = line.substring(15, 19);
    if (line.charAt(87) == '+') {
      temperature = Integer.parseInt(line.substring(88, 92));
    } else {
      temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
      context.write(new Text(year), new IntWritable(temperature));
  }
}
```

AverageReducer

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class AverageReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {

    int max_temp = 0;
```

```
    int count = 0;
    for (IntWritable value : values) {
      max_temp += value.get();
      count++;
     }
    context.write(key, new IntWritable(max_temp / count));
   }
 }
```





MeanMax

MeanMaxDriver

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MeanMaxDriver {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Please Enter the input and output parameters");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(MeanMaxDriver.class);
    job.setJobName("Max temperature");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(MeanMaxMapper.class);
    job.setReducerClass(MeanMaxReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

MeanMaxMapper

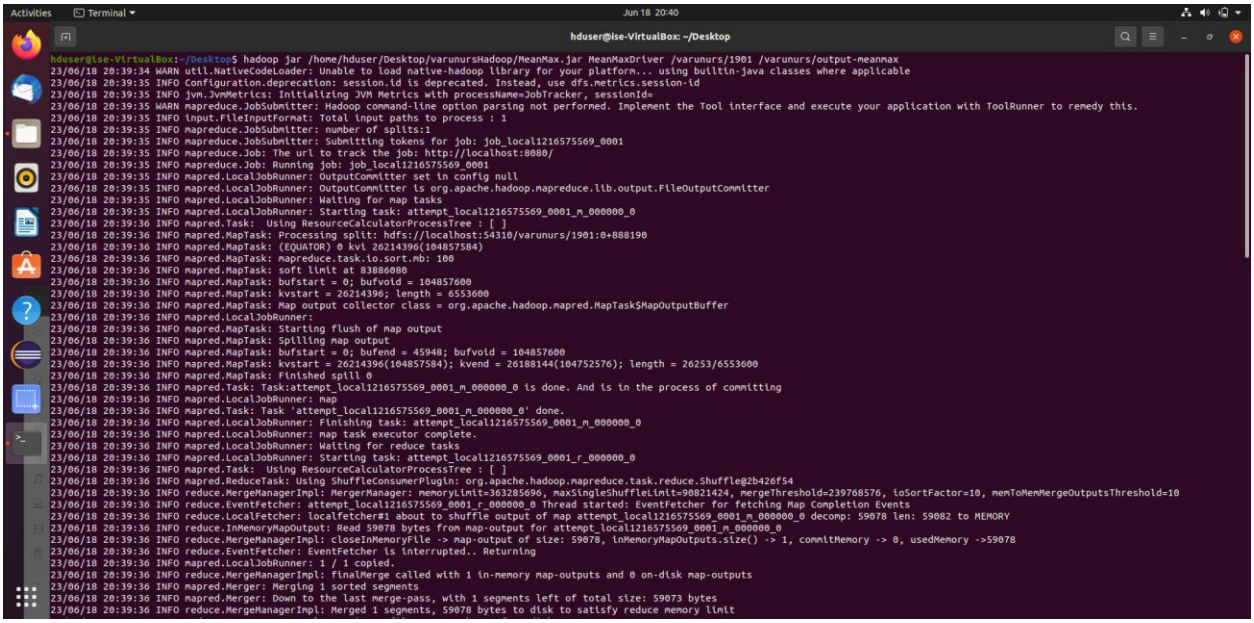
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
```

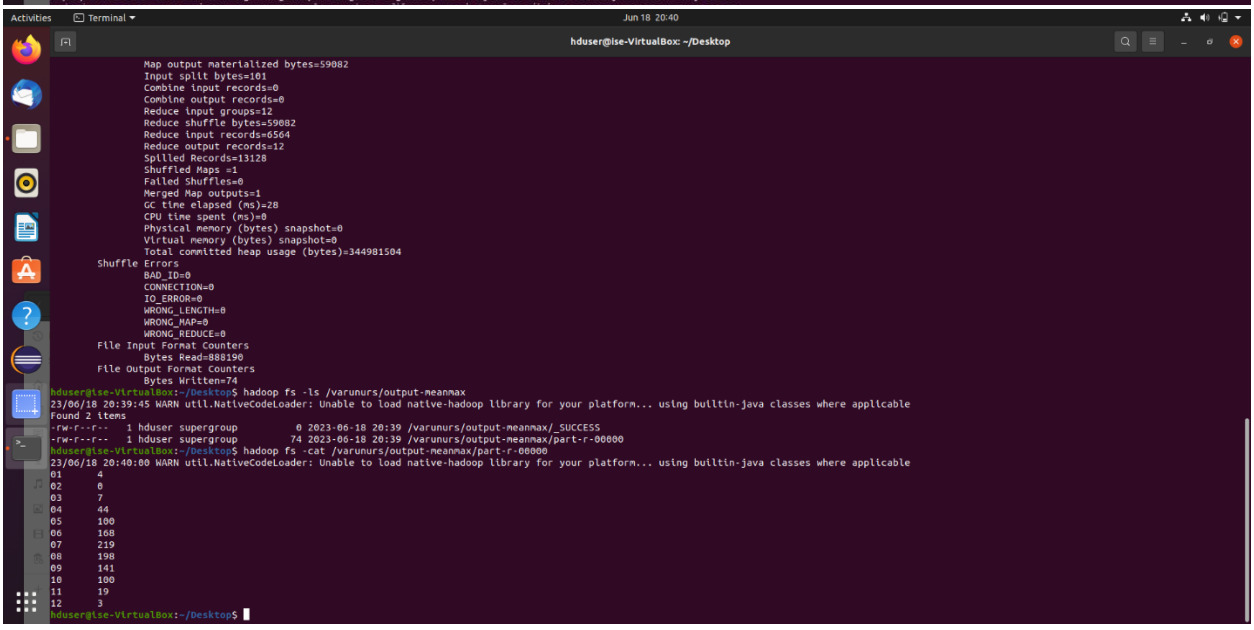
```
public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
  public static final int MISSING = 9999;

  public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int temperature;
    String line = value.toString();
    String month = line.substring(19, 21);
    if (line.charAt(87) == '+') {
      temperature = Integer.parseInt(line.substring(88, 92));
    } else {
      temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
      context.write(new Text(month), new IntWritable(temperature));
  }
}
```

MeanMaxReducer

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int max_temp = 0;
    int total_temp = 0;
    int count = 0;
    int days = 0;
    for (IntWritable value : values) {
      int temp = value.get();
      if (temp > max_temp)
        max_temp = temp;
```

```
    count++;
    if (count == 3) {
      total_temp += max_temp;
      max_temp = 0;
      count = 0;
      days++;
    }
  } context.write(key, new IntWritable(total_temp / days));
}}
```





LAB 7. For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

TopNDriver

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class TopN {
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();
    if (otherArgs.length != 2) {
      System.err.println("Usage: TopN <in> <out>");
      System.exit(2);
    }
    Job job = Job.getInstance(conf);
    job.setJobName("Top N");
    job.setJarByClass(TopN.class);
    job.setMapperClass(TopNMapper.class);
    job.setReducerClass(TopNReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

  public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
    private static final IntWritable one = new IntWritable(1);
```

```
    private Text word = new Text();

    private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";

    public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
      String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
      StringTokenizer itr = new StringTokenizer(cleanLine);
      while (itr.hasMoreTokens()) {
        this.word.set(itr.nextToken().trim());
        context.write(this.word, one);
      }
    }
  }
}
```

TopNCombiner

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values)
      sum += val.get();
    context.write(key, new IntWritable(sum));
  }
}
```

TopNMapper

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
  private static final IntWritable one = new IntWritable(1);

  private Text word = new Text();

  private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";

  public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
    String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
    StringTokenizer itr = new StringTokenizer(cleanLine);
    while (itr.hasMoreTokens()) {
      this.word.set(itr.nextToken().trim());
      context.write(this.word, one);
    }
  }
}
```

TopNReducer

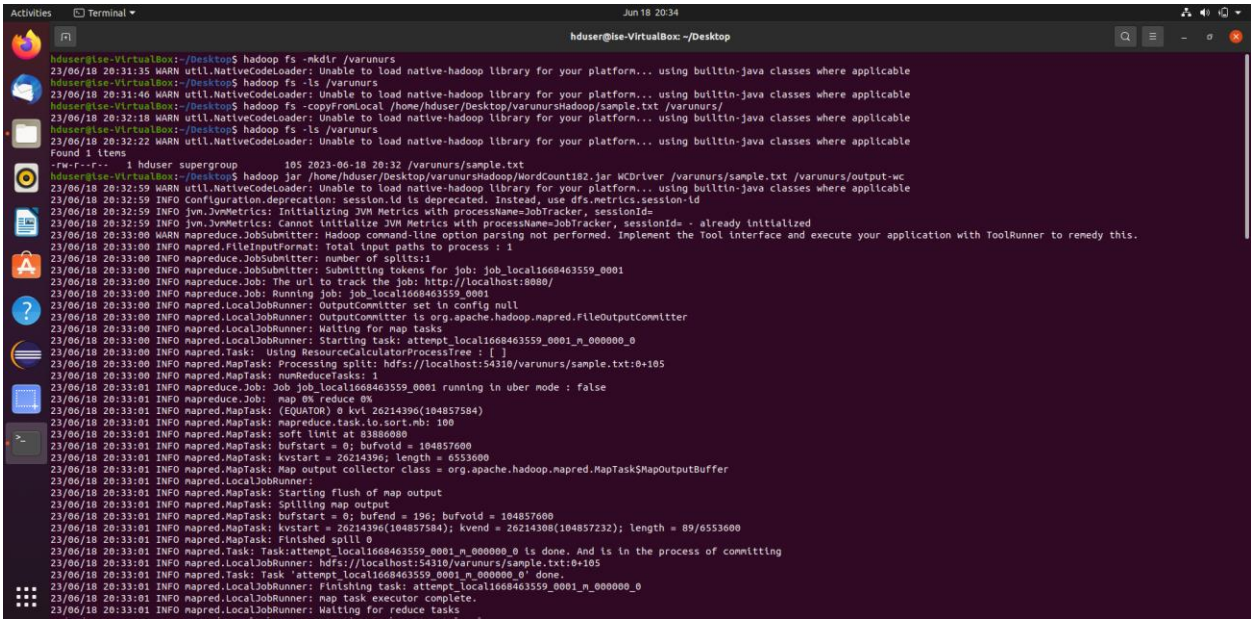
```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import utils.MiscUtils;
```

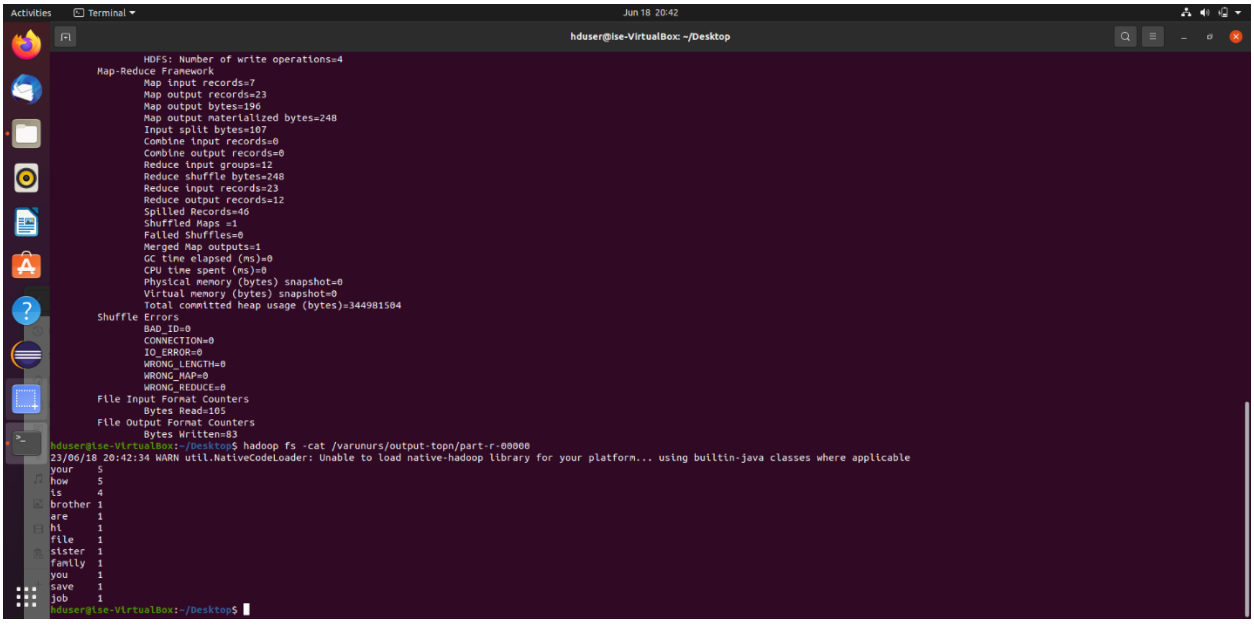
```
public class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  private Map<Text, IntWritable> countMap = new HashMap<>();

  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values)
      sum += val.get();
    this.countMap.put(new Text(key), new IntWritable(sum));
  }

  protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
IOException, InterruptedException {
    Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);
    int counter = 0;
    for (Text key : sortedMap.keySet()) {
      if (counter++ == 20)
        break;
      context.write(key, sortedMap.get(key));
    }
  }
}
```

MiscUtils

```
package utils;
import java.util.*;
public class MiscUtils {
public static <K extends Comparable, V extends Comparable> Map<K, V>
sortByValues(Map<K, V> map) {
List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K, V>>(map.entrySet());
Collections.sort(entries, new Comparator<Map.Entry<K, V>>() {
@Override
public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) {
return o2.getValue().compareTo(o1.getValue());
```

```
}
});
Map<K, V> sortedMap = new LinkedHashMap<K, V>();
for (Map.Entry<K, V> entry : entries) {
sortedMap.put(entry.getKey(), entry.getValue());
}
return sortedMap;
}
}
```





LAB 8. Create a Map Reduce program to demonstrating join operation

JoinDriver

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.MultipleInputs;
import org.apache.hadoop.util.*;


public class JoinDriver extends Configured implements Tool {

   public static class KeyPartitioner implements Partitioner<TextPair, Text> {
          @Override
          public void configure(JobConf job) {}

          @Override
          public int getPartition(TextPair key, Text value, int numPartitions) {
             return (key.getFirst().hashCode() & Integer.MAX_VALUE) % numPartitions;
          }
   }

   @Override
   public int run(String[] args) throws Exception {

      if (args.length != 3) {
         System.out.println("Usage: <Department Emp Strength input> <Department Name
input> <output>");
         return -1;
      }

      JobConf conf = new JobConf(getConf(), getClass());
         conf.setJobName("Join 'Department Emp Strength input' with 'Department Name
input'");

      Path AInputPath = new Path(args[0]);
      Path BInputPath = new Path(args[1]);
      Path outputPath = new Path(args[2]);
```

```
        MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,
DeptNameMapper.class);
        MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
DeptEmpStrengthMapper.class);

        FileOutputFormat.setOutputPath(conf, outputPath);

        conf.setPartitionerClass(KeyPartitioner.class);
           conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

           conf.setMapOutputKeyClass(TextPair.class);

        conf.setReducerClass(JoinReduce.class);

           conf.setOutputKeyClass(Text.class);

           JobClient.runJob(conf);

        return 0;
    }

    public static void main(String[] args) throws Exception {

        int exitCode = ToolRunner.run(new JoinDriver(), args);
        System.exit(exitCode);
    }
}
```

JoinReduce

```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
```

```
public class JoinReduce extends MapReduceBase implements Reducer<TextPair, Text, Text,
Text> {

   @Override
   public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text> output,
Reporter reporter)
            throws IOException
   {

      Text nodeId = new Text(values.next());
      while (values.hasNext()) {
         Text node = values.next();
         Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());
         output.collect(key.getFirst(), outValue);
      }
   }
}
```

TextPair

```
import java.io.*;

import org.apache.hadoop.io.*;

public class TextPair implements WritableComparable<TextPair> {

  private Text first;
  private Text second;

  public TextPair() {
    set(new Text(), new Text());
  }

  public TextPair(String first, String second) {
    set(new Text(first), new Text(second));
  }
```

```
public TextPair(Text first, Text second) {
  set(first, second);
}

public void set(Text first, Text second) {
  this.first = first;
  this.second = second;
}

public Text getFirst() {
  return first;
}

public Text getSecond() {
  return second;
}

@Override
public void write(DataOutput out) throws IOException {
  first.write(out);
  second.write(out);
}

@Override
public void readFields(DataInput in) throws IOException {
  first.readFields(in);
  second.readFields(in);
}

@Override
public int hashCode() {
  return first.hashCode() * 163 + second.hashCode();
}

@Override
public boolean equals(Object o) {
  if (o instanceof TextPair) {
    TextPair tp = (TextPair) o;
    return first.equals(tp.first) && second.equals(tp.second);
```

```
  }
  return false;
}

@Override
public String toString() {
  return first + "\t" + second;
}

@Override
public int compareTo(TextPair tp) {
  int cmp = first.compareTo(tp.first);
  if (cmp != 0) {
    return cmp;
  }
  return second.compareTo(tp.second);
}
// ^^ TextPair

// vv TextPairComparator
public static class Comparator extends WritableComparator {

  private static final Text.Comparator TEXT_COMPARATOR = new Text.Comparator();

  public Comparator() {
    super(TextPair.class);
  }

  @Override
  public int compare(byte[] b1, int s1, int l1,
                     byte[] b2, int s2, int l2) {

    try {
      int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
      int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
      int cmp = TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
      if (cmp != 0) {
        return cmp;
      }
```

```
      return TEXT_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,
                                     b2, s2 + firstL2, l2 - firstL2);
    } catch (IOException e) {
      throw new IllegalArgumentException(e);
    }
  }
}

static {
  WritableComparator.define(TextPair.class, new Comparator());
}
// ^^ TextPairComparator

// vv TextPairFirstComparator
public static class FirstComparator extends WritableComparator {

  private static final Text.Comparator TEXT_COMPARATOR = new Text.Comparator();

  public FirstComparator() {
    super(TextPair.class);
  }

  @Override
  public int compare(byte[] b1, int s1, int l1,
                     byte[] b2, int s2, int l2) {

    try {
      int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
      int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
      return TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
    } catch (IOException e) {
      throw new IllegalArgumentException(e);
    }
  }

  @Override
  public int compare(WritableComparable a, WritableComparable b) {
    if (a instanceof TextPair && b instanceof TextPair) {
      return ((TextPair) a).first.compareTo(((TextPair) b).first);
```

```
    }
    return super.compare(a, b);
   }
  }
  // ^^ TextPairFirstComparator

// vv TextPair
}
```

Posts

```
import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class Posts extends MapReduceBase implements Mapper<LongWritable, Text, TextPair,
Text> {

@Override
public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)
throws IOException
{
String valueString = value.toString();
String[] SingleNodeData = valueString.split("\t");
output.collect(new TextPair(SingleNodeData[3], "0"), new

Text(SingleNodeData[9]));
}
}
```

User

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.IntWritable;

public class User extends MapReduceBase implements Mapper<LongWritable, Text, TextPair,
Text> {

@Override
public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)

throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t");
output.collect(new TextPair(SingleNodeData[0], "1"), new Text(SingleNodeData[1]));
}
}
```

DeptEmpStrengthMapper

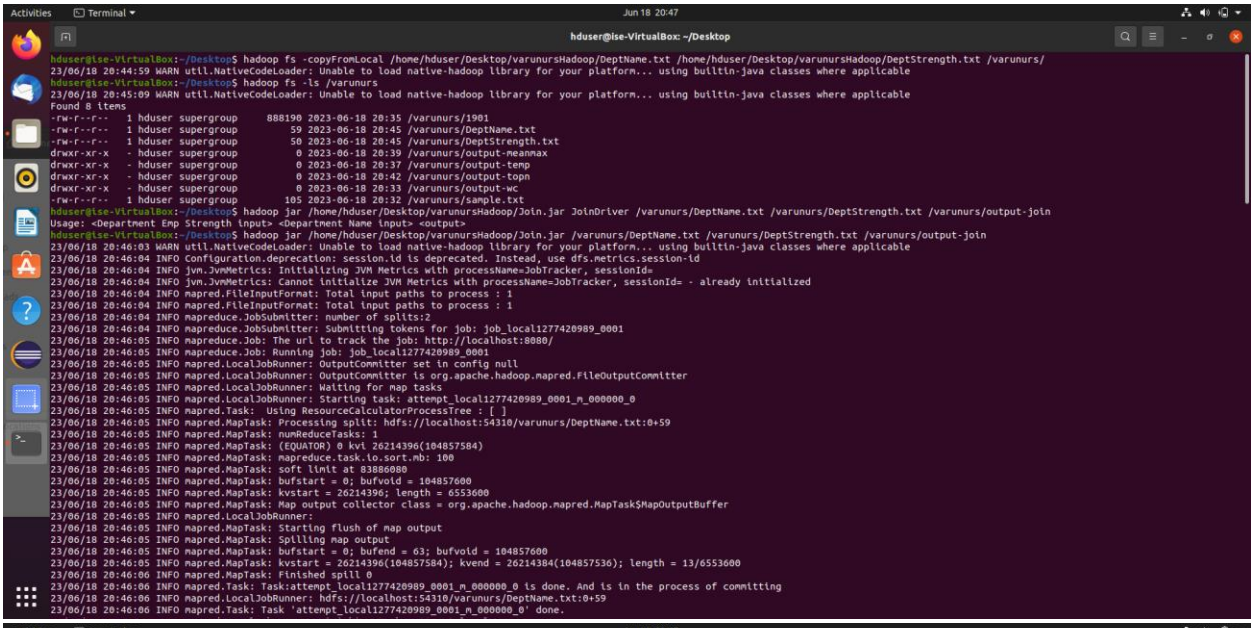
```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
```

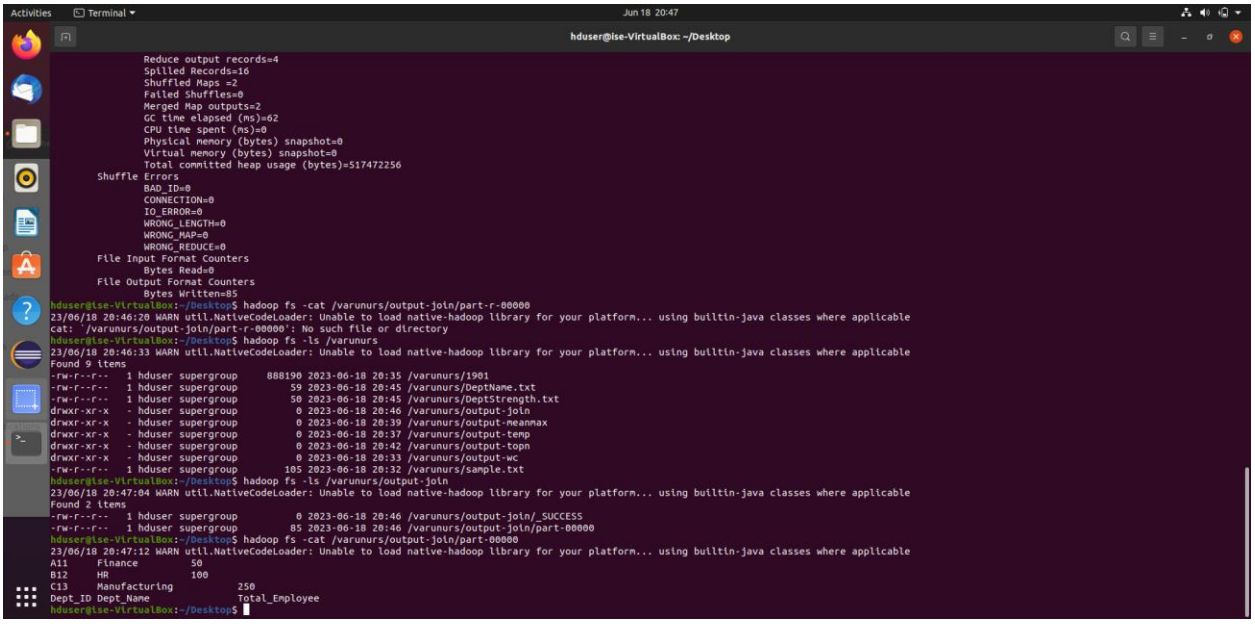
```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.IntWritable;

public class DeptEmpStrengthMapper extends MapReduceBase implements
Mapper<LongWritable, Text, TextPair, Text> {

   @Override
   public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)
           throws IOException
   {

      String valueString = value.toString();
      String[] SingleNodeData = valueString.split("\t");
      output.collect(new TextPair(SingleNodeData[0], "1"), new Text(SingleNodeData[1]));
   }
}
```

DeptNameMapper

```
import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class DeptNameMapper extends MapReduceBase implements Mapper<LongWritable,
Text, TextPair, Text>  {

   @Override
   public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)
           throws IOException
   {
```

```
        String valueString = value.toString();
        String[] SingleNodeData = valueString.split("\t");
        output.collect(new TextPair(SingleNodeData[0], "0"), new Text(SingleNodeData[1]));
    }
}
```



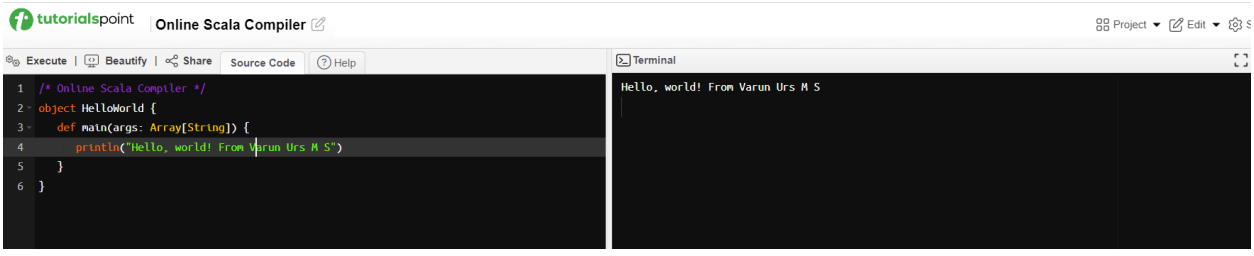


LAB 9. Program to print word count on scala shell and print “Hello world” on scala IDE

Word Count

```
scala> val data = sc.textFile("sparkdata.txt")
data: org.apache.spark.rdd.RDD[String] = sparkdata.txt MapPartitionsRDD[6] at textFile at <console>:2
4

scala> data.collect;
res5: Array[String] = Array(welcome to spark by varunurs and also welcome to the user by akash;)

scala> val splitdata1 = data.flatMap(line=>line.split(" "));
splitdata1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[7] at flatMap at <console>:25

scala> splitdata.collect;
res6: Array[String] = Array(welcome, to, spark, by, varunurs, and, also, welcome, to, the, user, by,
akash;)

scala> val mapdata1 = splitdata.map(word=>(word,1));
mapdata1: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[8] at map at <console>:25

scala> mapdata1.collect;
res7: Array[(String, Int)] = Array((welcome,1), (to,1), (spark,1), (by,1), (varunurs,1), (and,1), (al
so,1), (welcome,1), (to,1), (the,1), (user,1), (by,1), (akash;,1))

scala> val reddata = mapdata1.reduceByKey(_+_);
reddata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[9] at reduceByKey at <console>:25

scala> reddata.collect;
res8: Array[(String, Int)] = Array((welcome,2), (varunurs,1), (spark,1), (akash;,1), (to,2), (by,2),
(also,1), (user,1), (and,1), (the,1))

scala>
```

Hello World



LAB 10. Using RDD and FlaMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark