VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

CHAITANYA GADGIL (1BM19CS223)

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
May-2022 to July-2022

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 "**BIG DATA ANALYTICS**" carried out by **CHAITANYA GADGIL (1BM19CS223),** 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 2022. 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.

ANTARA ROY CHOUDURY	**Dr. Jyothi S Nayak**
Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	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

MongoDB – LAB 1

```
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.update({_id:1},{$set:{hobbies:"cricket"}},{upsert:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find()
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing",
"hobbies" : "cricket" }
> db.Student.insert({id:1,name:"xyz",grade:"VIII",hobbies:"chess"})
WriteResult({ "nInserted" : 1 })
> db.Student.find({name:/xyz/}).pretty()
{
"_id" : ObjectId("6256987834dadfe4d50f9d70"),
"id" : 1,
"name" : "xyz",
"grade" : "VIII",
"hobbies" : "chess"
}
> db.Student.find().sort({name:1}).pretty()
{
"_id" : 1,
"StudName" : "MichelleJacintha",
"Grade" : "VII",
"Hobbies" : "InternetSurfing",
"hobbies" : "cricket"
}
{
"_id" : ObjectId("6256987834dadfe4d50f9d70"),
"id" : 1,
"name" : "xyz",
"grade" : "VIII",
"hobbies" : "chess"
}
> db.Student.find().skip(1).pretty()
{
"_id" : ObjectId("6256987834dadfe4d50f9d70"),
"id" : 1,
"name" : "xyz",
"grade" : "VIII",
"hobbies" : "chess"
}
> db.createCollection("food")
{ "ok" : 1 }
```

```
> db.food.insert({_id:1,fruits:['grapes','mango']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['grapes','mango','cherry']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:3,fruits:['banana','cherry']})
WriteResult({ "nInserted" : 1 })
> db.food.find({fruits:['grapes','mango']})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
> db.food.find({'fruits':{$size:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
{ "_id" : 3, "fruits" : [ "banana", "cherry" ] }
> db.food.find({_id:2},{'fruits':{$slice:2}})
{ "_id" : 2, "fruits" : [ "grapes", "mango" ] }
> db.food.find({fruits:{$all:['grapes','mango']}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
> db.food.update({_id:3},{$set:{'fruits.1':'apple'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.food.find()
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
{ "_id" : 3, "fruits" : [ "banana", "apple" ] }
> db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.createCollection("Customers")
{ "ok" : 1 }
> db.Customers.insert({custId:1,acctBal:1000,acctType:"current"})
WriteResult({ "nInserted" : 1 })
> db.Customers.insert({custId:2,acctBal:2000,acctType:"current"})
WriteResult({ "nInserted" : 1 })
> db.Customers.insert({custId:3,acctBal:3000,acctType:"savings"})
WriteResult({ "nInserted" : 1 })
> db.Customers.aggregate({$group:{_id:"$custId",toAcctBal:{$sum:"$acctBal"}}})
{ "_id" : 3, "toAcctBal" : 3000 }
{ "_id" : 1, "toAcctBal" : 1000 }
{ "_id" : 2, "toAcctBal" : 2000 }
>
db.Customers.aggregate({$match:{acctType:"current"}},{$group:{_id:"$custId",toAcctBal:{$sum:"
$acctBal"}}})
{ "_id" : 2, "toAcctBal" : 2000 }
{ "_id" : 1, "toAcctBal" : 1000 }
>
db.Customers.aggregate({$match:{acctType:"current"}},{$group:{_id:"$custId",toAcctBal:{$sum:"
$acctBal"}}},
{$match:{toAcctBal:{$gt:500}}})
{ "_id" : 2, "toAcctBal" : 2000 }
{ "_id" : 1, "toAcctBal" : 1000}
```

MongoDB – LAB 2

```
> db.createCollection("Bank");
> {"ok": 1 }
> db.insert({CustID:1, Name:"Trivikran Hegde, Type:"Savings", Contact:["9945678231",
"080-22364587"}});
> db.Bank.Insert({CustID:1, Nane: "Trtvtkran Hegde, Type:"Savings",
Contact:["9945678231", "060-22364587"]}); writeResult([ 'nInserted": 1})
> db.Bank. Insert({CustID:2, Name: "Vishvesh Bhat", Type:"Savings,
Contact:["6325985615", "000-23651452"]}); WriteResult("ninserted": 1 })
> db.Bank. Insert({CustID:3, Name: "Vaishak that", Type:"Savings",
Contact:["8971456321", "000-13529458"]}); WriteResult((nInserted": 1})
> db.Bank Insert({CustID:4, Name: "Pramod P Parande", Type:"Current".
Contact:["9745236589", "080-56324587"]});
> writeResult({"nInserted": 1}}
> db.Bank.insert({CustID:4, Name: "Shreyas R 5, Type:"Current",
Contact:["9445678321","044-65611729", "080-25639856"]});
> WriteResult({nInserted": 1})
> db.Bank.find({});
{_id: ObjectId("625877809329139694718882"), "CustID" : 1, "Name": "Trivikran Hegde",
"Type": "Savings", "Contact" : [ "9945078231", "080-223645871"]}
{_id: ObjectId("625d77bd9329139694f188a3"), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings", "Contact" : [ "6325985615", "080-23651452"]}
{_id: ObjectId("625d77e693291396941884"), "CustID" : 3, "Name": "Vatshak Bhat",
"Type": "Savings", "Contact" : ["8971456321", "080-335294581]
{_id: ObjectId("62547822932913989 4f188a5"), "CustID" : 4, "Name": "Praned P Parande",
"Type" : "Current", "Contact" : [ "9745236589", "080-56324587"]}
{_id: ObjectId("625d78659329139894f188a6"), "CustID" : 4, "Name": "shreyas R 5",
"Type"
: "Current", "Contact" : [ "9445678321", "044-65011729"]}
> db.Bank.updateMany({CustID:1},{$pop: {Contact:1}});
{acknowledged": true, "natchedCount": 1, "nodifiedCount": 1}
> db.Bank.find();
{_id: ObjectId("625d7709329139694f188a2"), "CustIo": 1, "Name": "Trivikran Hegde",
"Type": "Savings", "Contact":"9945678231" }
{_id: ObjectId>("625d77bd9329139694f18a3"), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings", "Contact" : [ "Savings", "Contact" : [ "6325985615", "010-23651452"]}
{_id: ObjectId("625d77e6932913989471884"), "CustID" : 3, "Name": "Vaishak Bhat",
"Type": "Savings", "Contact":["8971456321", "080-3529458"]}
{_id ObjectId("625d782293291396947188a5"), "CustID" : 4, "Name": "Pramod P Parande",
"Type" : "Current", "Contact" : [ "9745236589", "080-56324587"]}
```

```
{_id: ObjectId("625d786593291396947188a6"), "CustID": 4, "Name": "Shreyas R S",
"Type" : "Current", "Contact" : [ "9445678321", "044-65611729"]}
>db.Bank.updateMany({CustID: 1}), {$pull: {Contact:"000-25639856"}} };
acknowledged": true, "natchedCount": 5, "modifiedCount": 1 }
>db.Bank.find({});
{_id: ObjectId("625d77809329139694f18882"), "CustID" : 1, "Name": "Trivikram Hegde",
"Type": "Savings", "Contact" ["9945678231" ]},
{_id: ObjectId("625877bd9329139694f1983), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings", "Contact": ["6325985615", "080-23651452"]}
{_id: ObjectId("625677e69329139694718804"), "CustID": 3, "Name": "Vaishak Bhat",
"Type": "Savings", "Contact":["8971456321", "080-33529458"]}
{_id ObjectId("625d7822932913969471885), "CustID": 4, "Name": "Pranod Parande",
"Type" : "Current", "Contact" : ["9745236589", "080-563245871"]}
{_id: ObjectId("625678659329139694188a6"), "CustID": 4, "Name": "Shreyas RS", "Type"
:
"Current", "Contact" : [ 9445678321", "044-65011729"]}
>db.Bank.createIndex({Name:1, Type:1}, {name:});
uncaught exception: SyntaxError: expected expression, got '}'
(shell)11:43
db.Bank.createIndex({Name:1, Type:1}, {name:"Find current account holders"});
{
"createdCollectionAutomatically":false,
"nunIndexesBefore": 1,
TounIndexesAfter": 2,
"ok": 1
}
>db.Bank.find({});
{_id: ObjectId("62567708932913969410882"), "CustID": 1, "Name": "Trivikram Hegde"
"Type": "Savings", "Contact":"9945678231"]}
{_id: ObjectId("625477bd932913969410883"), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings", "Contact" : ["6325985615", "080-23651452"]
{_id: ObjectId("625d77e59329139694718884"), "CustID": 3, "Name": "Vatshak Bhat",
"Type": "Savings", "Contact: [ "8971456321", "080-33529458"]}
{_id: ObjectId("6254782293291396694f188a5″), "CustID" : 4, "Name": "Pramod P Parande",
"Type" : "Current", "Contact" : [ "9745236589", "080-56324587"]}
{_id: ObjectId("625878659329139694718806"), "CustID": 4, "Name": "Shreyas RS",
"Type"
: "Current", "Contact" : [ "9445678321", "044-65611729"]}
>db.Bank.getIndexes()
> db.Bank.update({id:625078659329139694F188a6), ($set: {CustID:53}, {upsert:true});
uncaught exception: SyntaxError: identifier starts innediately after > numeric literal:
(shell):1:20
> db.Bank.update({id:"62585932913941"}, {$set: {CustID:5}}, {upsert:true});
writeResult({
nhatched":0,
"nUpserted" 11,
"Modified": 0,
"id":"625d78659329139694f18826"
})
> db.Bank.find({});
{_id: ObjectId("625d7700932913909418882"), "CustID" : 1, "Name": "Trivikran Hegde",
"Type": "Savings", "Contact":"9945678231" ] }
{_id: ObjectId("625477bd9329139094F188a3"), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings. "Contact" : ["0325981615", "O80-36514521"]}
{_id: ObjectId("625d77e693291396947188a4"), "CustID": 3, "Name": "Valshak Bhat",
"Type": "Savings", "Contact": [ "8971456321", "080-33529458"]}
{_id: ObjectId("825878229329139694188a5"), "CustID: 4, "Name": "Pramod P Parande",
"Type" : "Current", "Contact" : [ "9745236589", "080-56324587"]}
{_id: ObjectId("625d766593291390941886"), "CustID": 4, "Name": "Shreyas R S", "Type" :
"Current", "Contact" : [ "9445678323", "044-65611729"]}
> db.Bank.update({_id:"62507865932913969471880a6", CustID:4}, {$set:
{Name:"Sumantha
K 5, Type:"Savings", Contact:["9856325478","11-65897458"]},{upsert:true});
WriteResult("Matched" : 1, "nupserted": 6, "Modified":1})
> db.Bank.find({});
{_id: ObjectId("625d7780932913909418882"), "CustID": 1, "Name": "Trivikran Hegde",
"Type": "Savings", "Contact" : [ "9945678231"] }
{_id: ObjectId("625d77bd9329139694f188a3"), "CustID": 2, "Name": "Vishvesh Bhat",
"Type": "Savings, "Contact" : ["6325985615", "080-36514529"]}
{_id: ObjectId("825d77e6932913969418844"), "CustID" : 3, "Name": "Vaishak Bhat",
"Type": "Savings", "Contact" : [ "8971456321", "080-34529458"]}
{_id: ObjectId("625d78229329139094F188a5"), "CustID" : 4, "Name": "Pranod P Parande",
"Type" : "Current", "Contact" : [ "9745236589", "080-56324587"]}
{(id: ObjectId("625d78659329139694f188a6"), "CustID: 4, "Name": "Sumantha x 5",
"Type": "Savings", "Contact" : ["9445678321", "044-05611729"]}
```

Lab 3 - Cassandra
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
Employee

```
cqlsh> create keyspace mployee_space WITH REPLICATION = {'class' :
'SimpleStrategy','replication_factor':2};
CREATE TABLE employee_space.employee_info (emp_id int PRIMARY KEY,emp_name
text,designation
text,date_of_joining timestamp,salary float,dept_name text);
cqlsh> begin batch INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
VALUES(1,'abc','Manager','2022-01-24',100000,'Marketing');
... apply batch;
cqlsh> begin batch INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
```

```
VALUES(2,'pqr','Accountant','2021-01-24',200000,'Accounts');
... INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
VALUES(3,'xyz','Manager','2021-03-24',500000,'Marketing');
... INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
VALUES(4,'ijk','Administrator','2021-05-24',500000,'Administration');
... INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
VALUES(5,'lmn','Administrator','2009-05-24',2000000,'Administration');
... apply batch;
cqlsh> use employee_space;
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | salary
--------+---------------------------------+----------------+---------------+-----------+--------
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | 2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Marketing | Manager | abc | 1e+05
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk | 5e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz | 5e+05
(5 rows)
cqlsh:employee_space> update employee_info set emp_name='efg' where emp_id=1;
cqlsh:employee_space> update employee_info set dept_name='Development' where
emp_id=1;
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | salary
--------+---------------------------------+----------------+---------------+-----------+--------
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | 2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | efg | 1e+05
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk | 5e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz | 5e+05
(5 rows)
cqlsh:employee_space> alter table employee_info add projects set<text>;
cqlsh:employee_space> update employee_info set projects=projects+{'Web
development','machine learning'} where
emp_id=2;
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | projects | salary
--------+---------------------------------+----------------+---------------+-----------+--------------------
---------------------
+--------
```

```
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | null |
2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | efg | null |
1e+05
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | {'Web development',
'machine
learning'} | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk | null |
5e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz | null | 5e+05
(5 rows)
cqlsh:employee_space> update employee_info set projects=projects+{'Web
development','machine
learning','cybersecurity'} where emp_id=5;
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | projects
| salary
--------+---------------------------------+----------------+---------------+-----------
+-------------------------------------------------------+--------
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | {'Web
development',
'cybersecurity', 'machine learning'} | 2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | efg |
null | 1e+05
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | {'Web development',
'machine learning'} | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk |
null | 5e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz |
null | 5e+05
(5 rows)
cqlsh:employee_space> INSERT INTO
employee_space.employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_
name)
VALUES(6,'mno','Manager','2022-01-24',100000,'Marketing') using ttl 15;
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | projects
| salary
--------+---------------------------------+----------------+---------------+-----------
+-------------------------------------------------------+--------
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | {'Web
development',
'cybersecurity', 'machine learning'} | 2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | efg |
null | 1e+05
```

```
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | {'Web development',
'machine learning'} | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk |
null | 5e+05
6 | 2022-01-23 18:30:00.000000+0000 | Marketing | Manager | mno |
null | 1e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz |
null | 5e+05
(6 rows)
cqlsh:employee_space> select * from employee_info;
emp_id | date_of_joining | dept_name | designation | emp_name | projects
| salary
--------+---------------------------------+----------------+---------------+-----------
+---------------------------------------------------------+--------
5 | 2009-05-23 18:30:00.000000+0000 | Administration | Administrator | lmn | {'Web
development',
'cybersecurity', 'machine learning'} | 2e+06
1 | 2022-01-23 18:30:00.000000+0000 | Development | Manager | efg |
null | 1e+05
2 | 2021-01-23 18:30:00.000000+0000 | Accounts | Accountant | pqr | {'Web development',
'machine learning'} | 2e+05
4 | 2021-05-23 18:30:00.000000+0000 | Administration | Administrator | ijk |
null | 5e+05
3 | 2021-03-23 18:30:00.000000+0000 | Marketing | Manager | xyz |
null | 5e+05
(5 rows)
```

Lab 4 - Cassandra

Perform the following DB operations using Cassandra.

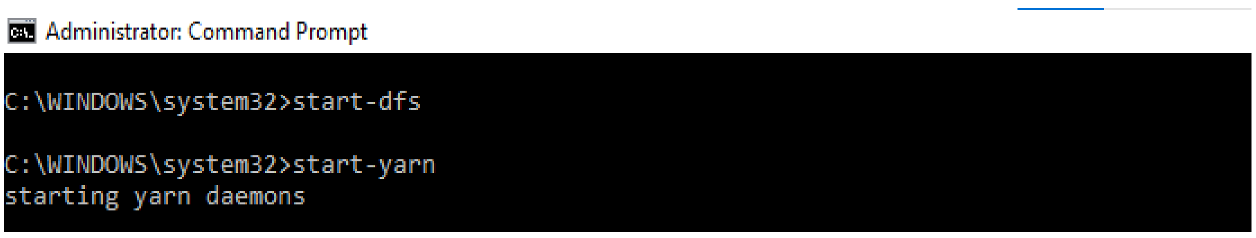
1. Create a keyspace by name Library
2. Create a column family by name Library-Info with attributes
Stud_Id Primary Key,
Counter_value of type Counter,
Stud_Name, Book-Name, Book-Id,
Date_of_issue
3. Insert the values into the table in batch
4. Display the details of the table created and increase the 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

```
cqlsh> create keyspace library_space WITH
REPLICATION={'class':'SimpleStrategy','replication_factor':2};
cqlsh> use library_space;
cqlsh:library_space> create table library_info(stud_id int,counter_value counter,stud_name
text,book_name
text,book_id int,date_of_issue timestamp,PRIMARY
KEY(stud_id,stud_name,book_name,book_id,date_of_issue));
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=1 and stud_name='abc'
and book_name='book1' and book_id=11 and date_of_issue='2022-01-30';
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=2 and stud_name='def'
and book_name='book2' and book_id=12 and date_of_issue='2022-03-30';
```

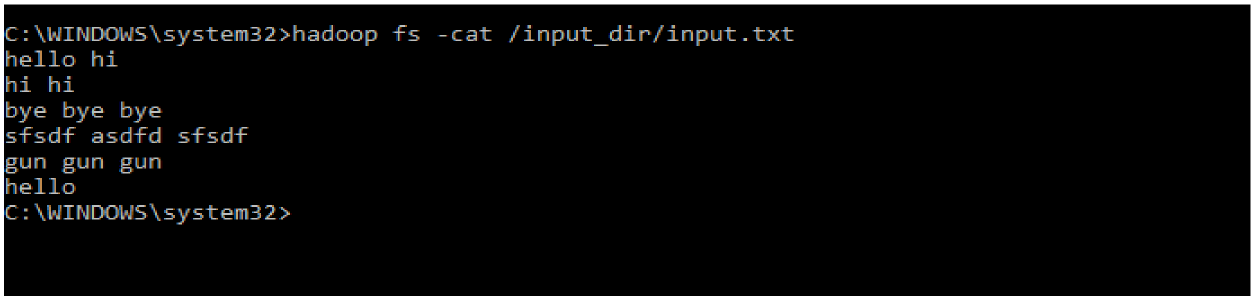
```
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=3 and stud_name='ghi'
and book_name='book3' and book_id=13 and date_of_issue='2022-05-30';
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=4 and stud_name='jkl'
and book_name='book4' and book_id=14 and date_of_issue='2022-07-30';
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=5 and stud_name='mno'
and book_name='book5' and book_id=15 and date_of_issue='2022-09-30';
cqlsh:library_space> select * from library_info;
stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+--------------------------------+---------------
5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 | 1
1 | abc | book1 | 11 | 2022-01-29 18:30:00.000000+0000 | 1
2 | def | book2 | 12 | 2022-03-29 18:30:00.000000+0000 | 1
4 | jkl | book4 | 14 | 2022-07-29 18:30:00.000000+0000 | 1
3 | ghi | book3 | 13 | 2022-05-29 18:30:00.000000+0000 | 1
(5 rows)
cqlsh:library_space> update library_info set counter_value=counter_value+1 where
stud_id=5 and stud_name='mno'
and book_name='book5' and book_id=15 and date_of_issue='2022-09-30';
cqlsh:library_space> select * from library_info;
stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+--------------------------------+---------------
5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 | 2
1 | abc | book1 | 11 | 2022-01-29 18:30:00.000000+0000 | 1
2 | def | book2 | 12 | 2022-03-29 18:30:00.000000+0000 | 1
4 | jkl | book4 | 14 | 2022-07-29 18:30:00.000000+0000 | 1
3 | ghi | book3 | 13 | 2022-05-29 18:30:00.000000+0000 | 1
(5 rows)
cqlsh:library_space> copy
library_info(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) to
'/home/bmscecse/Desktop/bda.csv';
Using 11 child processes

Starting copy of library_space.library_info with columns [stud_id, stud_name, book_name,

book_id, date_of_issue,

counter_value].

Processed: 5 rows; Rate: 45 rows/s; Avg. rate: 45 rows/s

5 rows exported to 1 files in 0.121 seconds.

cqlsh:library_space> create table library_info_copy(stud_id int,counter_value

counter,stud_name text,book_name
text,book_id int,date_of_issue timestamp,PRIMARY
KEY(stud_id,stud_name,book_name,book_id,date_of_issue));
cqlsh:library_space> copy
library_info_copy(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value)
from '/home/bmscecse/Desktop/new.csv';
Using 11 child processes
Starting copy of library_space.library_info_copy with columns [stud_id, stud_name,
book_name, book_id,
date_of_issue, counter_value].
Processed: 5 rows; Rate: 8 rows/s; Avg. rate: 12 rows/s
5 rows imported from 1 files in 0.406 seconds (0 skipped).
cqlsh:library_space> select * from library_info where counter_value=2 allow filtering;
stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
5 | mno | book5 | 15 | 2022-09-29 18:30:00.000000+0000 | 2
```

LAB 5

SCREENSHOT OF HADOOP INSTALLATION


Administrator: Command Prompt

```
C:\WINDOWS\system32>start-dfs

C:\WINDOWS\system32>start-yarn
starting yarn daemons
```

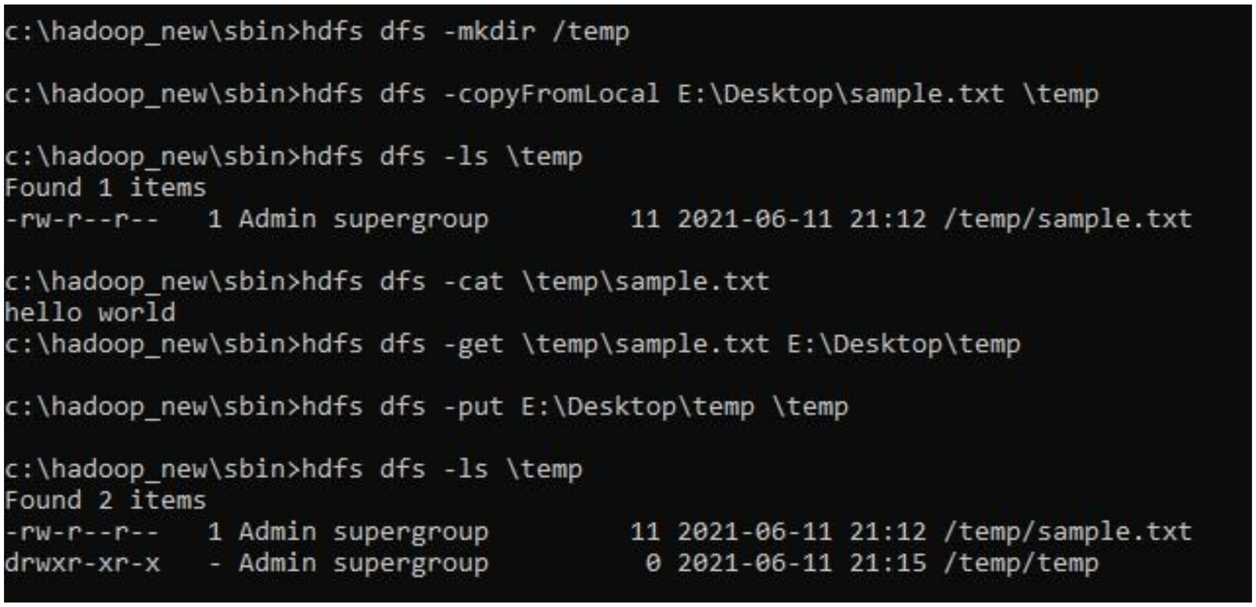


```
C:\WINDOWS\system32>hadoop fs -cat /input_dir/input.txt
hello hi
hi hi
bye bye bye
sfsdf asdfd sfsdf
gun gun gun
hello
C:\WINDOWS\system32>
```

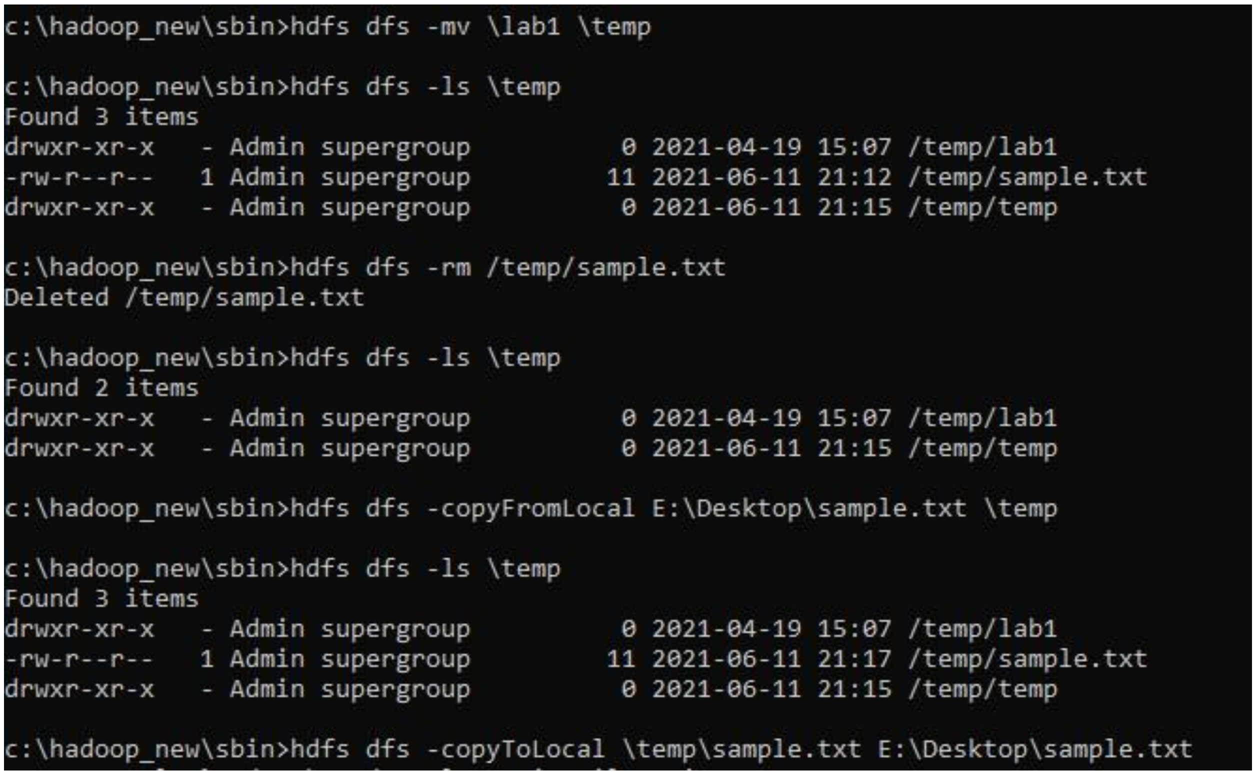
LAB 6

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

```
c:\hadoop_new\sbin>hdfs dfs -mkdir /temp


c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp


c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 1 items
-rw-r--r--   1 Admin supergroup        11 2021-06-11 21:12 /temp/sample.txt

c:\hadoop_new\sbin>hdfs dfs -cat \temp\sample.txt hello
world

c:\hadoop_new\sbin>hdfs dfs -get \temp\sample.txt E:\Desktop\temp


c:\hadoop_new\sbin>hdfs dfs -put E:\Desktop\temp \temp


c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 2 items
-rw-r--r--   1 Admin supergroup        11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x   -
Admin supergroup         0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -mv \lab1 \temp


c:\hadoop_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x   - Admin
supergroup         0 2021-04-19 15:07 /temp/lab1 -rw-r--r--   1 Admin
supergroup        11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x   -
Admin supergroup         0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -rm /temp/sample.txt
Deleted /temp/sample.txt

c:\hadoop_new\sbin>hdfs dfs -ls \temp Found 2 items drwxr-xr-x   - Admin
supergroup         0 2021-04-19 15:07 /temp/lab1 drwxr-xr-x   - Admin
supergroup         0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x   - Admin
supergroup         0 2021-04-19 15:07 /temp/lab1 -rw-r--r--   1 Admin supergroup
11 2021-06-11 21:17 /temp/sample.txt drwxr-xr-x   - Admin supergroup         0
2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyToLocal \temp\sample.txt E:\Desktop\sample.txt
```



```
c:\hadoop_new\sbin>hdfs dfs -mkdir /temp

c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 1 items
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:12 /temp/sample.txt

c:\hadoop_new\sbin>hdfs dfs -cat \temp\sample.txt
hello world
c:\hadoop_new\sbin>hdfs dfs -get \temp\sample.txt E:\Desktop\temp

c:\hadoop_new\sbin>hdfs dfs -put E:\Desktop\temp \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 2 items
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:12 /temp/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-06-11 21:15 /temp/temp
```



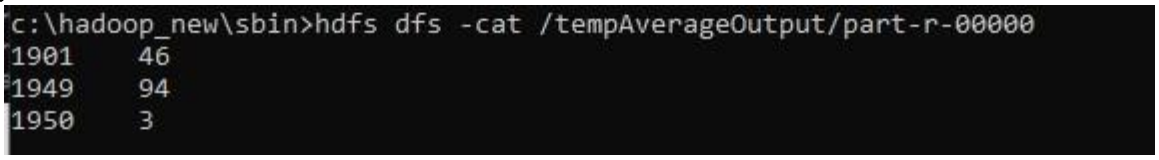
```
c:\hadoop_new\sbin>hdfs dfs -mv \lab1 \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 3 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 15:07 /temp/lab1
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:12 /temp/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -rm /temp/sample.txt
Deleted /temp/sample.txt

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 2 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 15:07 /temp/lab1
drwxr-xr-x   - Admin supergroup          0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 3 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 15:07 /temp/lab1
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:17 /temp/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyToLocal \temp\sample.txt E:\Desktop\sample.txt
```

LAB 7

For the given file, Create a Map Reduce program to
a) Find the average temperature for each year from the NCDC data set.

```
// AverageDriver.java package temperature;

import org.apache.hadoop.io.*; import org.apache.hadoop.fs.*; import
org.apache.hadoop.mapreduce.*; 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.java package temperature;

import org.apache.hadoop.io.*; import org.apache.hadoop.mapreduce.*; import java.io.IOException;

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

public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
{
        String line = value.toString();     String year = line.substring(15,19);       int temperature;
        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 != MISSING && quality.matches("[01459]"))
        context.write(new Text(year),new IntWritable(temperature)); }
}
```

```
//AverageReducer.java package temperature;

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

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable>
{
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException,InterruptedException
        {
                int max_temp = 0;               int count = 0;
                for (IntWritable value : values)
                {
                        max_temp += value.get();
                        count+=1;
                }
                context.write(key, new IntWritable(max_temp/count));
        }
}
```



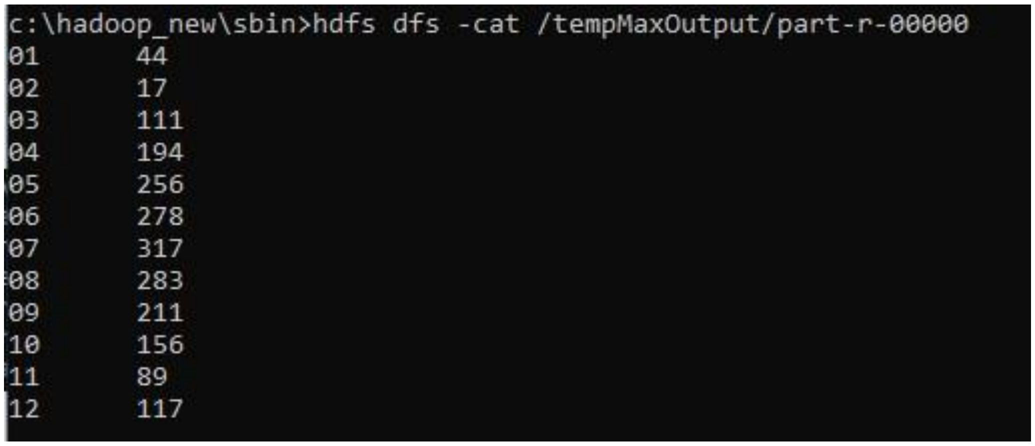
```
c:\hadoop_new\sbin>hdfs dfs -cat /tempAverageOutput/part-r-00000
1901    46
1949    94
1950    3
```

```
//TempDriver.java package
temperatureMax;

import org.apache.hadoop.io.*; import org.apache.hadoop.fs.*; import
org.apache.hadoop.mapreduce.*; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TempDriver
{       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(TempDriver.class);                  job.setJobName("Max
temperature");
                         FileInputFormat.addInputPath(job,new Path(args[0]));
                         FileOutputFormat.setOutputPath(job,new Path (args[1]));

                       job.setMapperClass(TempMapper.class);
job.setReducerClass(TempReducer.class);
                       job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true)?0:1);
            }
}

//TempMapper.java package
temperatureMax;

import org.apache.hadoop.io.*; import
org.apache.hadoop.mapreduce.*; import
java.io.IOException;

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

public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
{
```

```
            String line = value.toString();      String month = line.substring(19,21);
int temperature;            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 != MISSING &&
quality.matches("[01459]"))                  context.write(new Text(month),new
IntWritable(temperature)); }

}

//TempReducer.java package
temperatureMax;

import org.apache.hadoop.io.*; import
org.apache.hadoop.mapreduce.*; import
java.io.IOException;

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

public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
{
            String line = value.toString();      String month = line.substring(19,21);
int temperature;            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 != MISSING &&
quality.matches("[01459]"))                  context.write(new Text(month),new
IntWritable(temperature));
```

```
        }
}
```



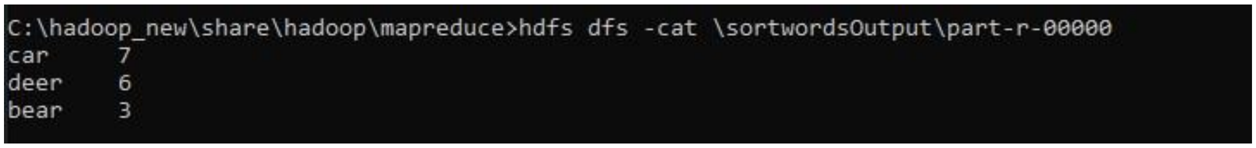
```
c:\hadoop_new\sbin>hdfs dfs -cat /tempMaxOutput/part-r-00000
01      44
02      17
03      111
04      194
05      256
06      278
07      317
08      283
09      211
10      156
11      89
12      117
```

LAB 8

For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top ‘n’ maximum occurrence of words.

```
// TopN.java package sortWords;

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.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser; import utils.MiscUtils;

import java.io.IOException; import java.util.*;

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.setCombinerClass(TopNReducer.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);
  }

  /**
   * The mapper reads one line at the time, splits it into an array of single words and emits every      *
word to the reducers with the value of 1.
   */
  public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);        private Text word = new Text();
    private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";

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

   /**
    * The reducer retrieves every word and puts it into a Map: if the word already exists in the      * map,
increments its value, otherwise sets it to 1.
    */
   public static class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

       private Map<Text, IntWritable> countMap = new HashMap<>();

       @Override
       public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {

          // computes the number of occurrences of a single word           int sum = 0;           for
(IntWritable val : values) {              sum += val.get();
          }
          // puts the number of occurrences of this word into the map.
          // We need to create another Text object because the Text instance
          // we receive is the same for all the words           countMap.put(new Text(key), new
IntWritable(sum));
       }
@Override
       protected void cleanup(Context context) throws IOException, InterruptedException {

          Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(countMap);

          int counter = 0;           for (Text key : sortedMap.keySet()) {              if (counter++ == 3) {
break;
             }
             context.write(key, sortedMap.get(key));
          }
       }
   }

   /**
    * The combiner retrieves every word and puts it into a Map: if the word already exists in the      *
map, increments its value, otherwise sets it to 1.
    */
   public static class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {

       @Override
```

```
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {

            // computes the number of occurrences of a single word            int sum = 0;            for
(IntWritable val : values) {                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
}
    }
}

// MiscUtils.java package utils;

import java.util.*;

public class MiscUtils {

    /**
sorts the map by values. Taken from:
http://javarevisited.blogspot.it/2012/12/how-to-sort-hashmap-java-by-key-and-value.html
     */
    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());
            }
        });

        //LinkedHashMap will keep the keys in the order they are inserted
        //which is currently sorted on natural ordering
        Map<K, V> sortedMap = new LinkedHashMap<K, V>();
for (Map.Entry<K, V> entry : entries) {
            sortedMap.put(entry.getKey(), entry.getValue());
        }

        return sortedMap;
    }
}
```



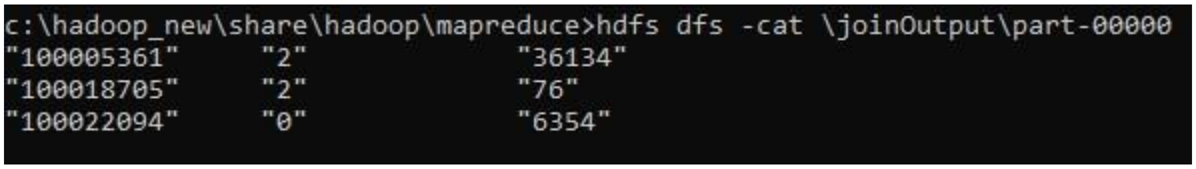
```
C:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \sortwordsOutput\part-r-00000
car     7
deer    6
bear    3
```

LAB 9

Create a Hadoop Map Reduce program to combine information from the users file along with Information from the posts file by using the concept of join and display user_id, Reputation and Score.

```
// JoinDriver.java 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,
Posts.class);
                MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
User.class);

                FileOutputFormat.setOutputPath(conf, outputPath);

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

                conf.setMapOutputKeyClass(TextPair.class);
                conf.setReducerClass(JoinReducer.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);
        }
}

// JoinReducer.java import java.io.IOException; import java.util.Iterator;

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

public class JoinReducer 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);
                }
        }
}

// User.java 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]));
            }
}

//Posts.java 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]));
            }
}

// TextPair.java 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());
  }
  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);
    }
  }
}
```



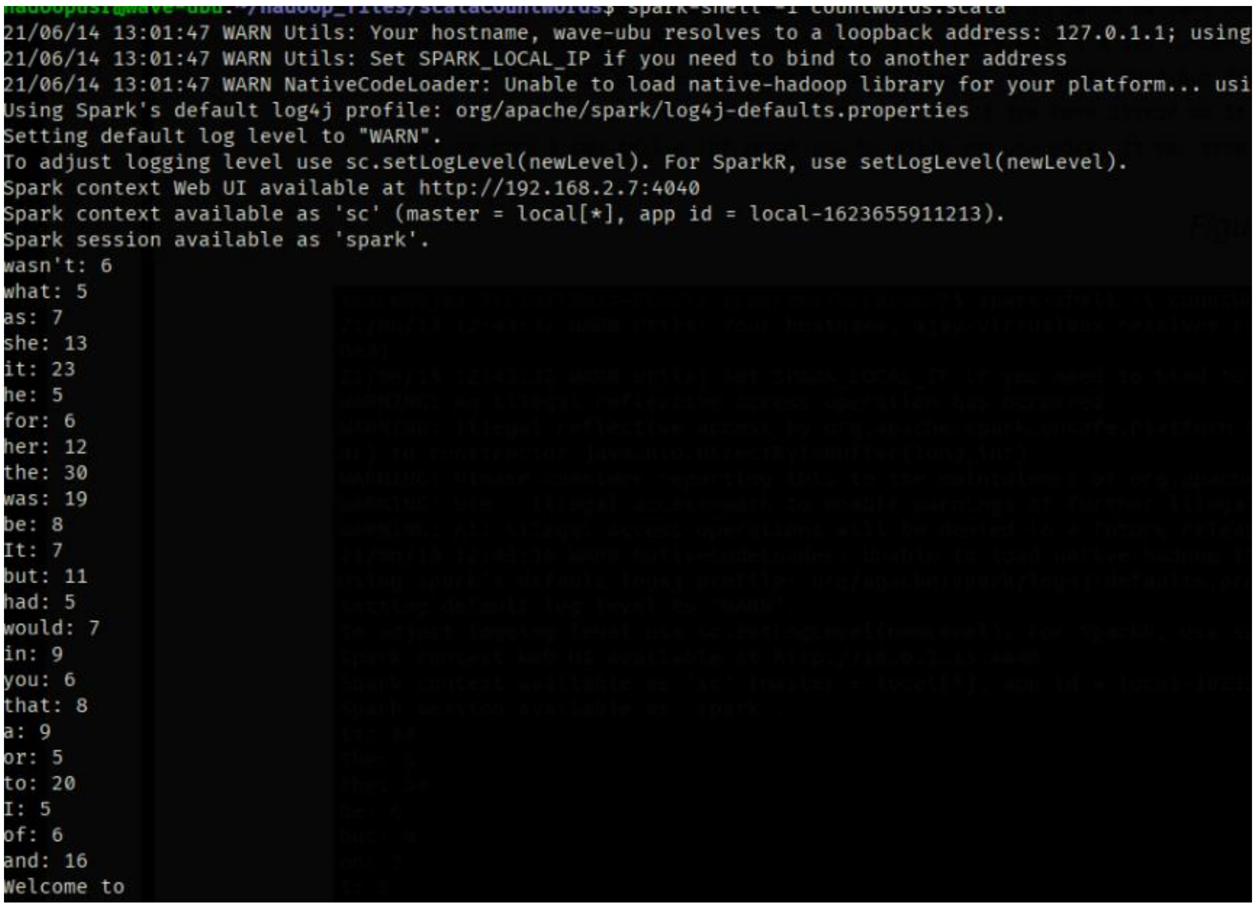
```
c:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \joinOutput\part-00000
"100005361"     "2"             "36134"
"100018705"     "2"             "76"
"100022094"     "0"             "6354"
```

LAB 10

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

```
scala> println("Hello World!");
Hello World!
```

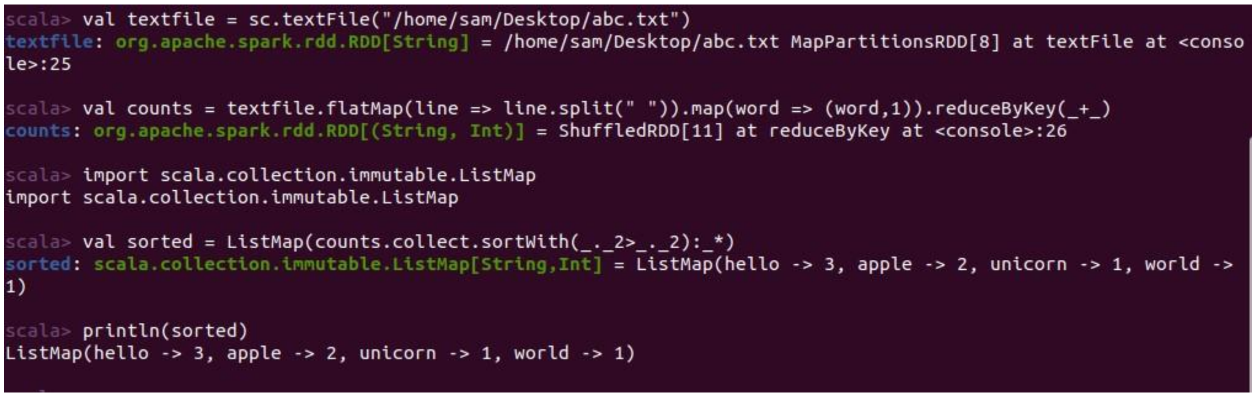
```
val data=sc.textFile("sparkdata.txt")
data.collect;
val splitdata = data.flatMap(line => line.split(" "));
splitdata.collect;
val mapdata = splitdata.map(word => (word,1));
mapdata.collect;
val reducedata = mapdata.reduceByKey(_+_);
reducedata.collect;
```



```
hadoopusr@wave-ubu:~/hadoop_files/scalaCountWords$ spark-shell -i countwords.scala
21/06/14 13:01:47 WARN Utils: Your hostname, wave-ubu resolves to a loopback address: 127.0.1.1; using
21/06/14 13:01:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
21/06/14 13:01:47 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.2.7:4040
Spark context available as 'sc' (master = local[*], app id = local-1623655911213).
Spark session available as 'spark'.
wasn't: 6
what: 5
as: 7
she: 13
it: 23
he: 5
for: 6
her: 12
the: 30
was: 19
be: 8
It: 7
but: 11
had: 5
would: 7
in: 9
you: 6
that: 8
a: 9
or: 5
to: 20
I: 5
of: 6
and: 16
Welcome to
```

LAB 11

**Using RDD and Flat Map 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**



```
scala> val textfile = sc.textFile("/home/sam/Desktop/abc.txt")
textfile: org.apache.spark.rdd.RDD[String] = /home/sam/Desktop/abc.txt MapPartitionsRDD[8] at textFile at <conso
le>:25

scala> val counts = textfile.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_+_)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:26

scala> import scala.collection.immutable.ListMap
import scala.collection.immutable.ListMap

scala> val sorted = ListMap(counts.collect.sortWith(_._2>_._2):_*)
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(hello -> 3, apple -> 2, unicorn -> 1, world ->
1)

scala> println(sorted)
ListMap(hello -> 3, apple -> 2, unicorn -> 1, world -> 1)
```