VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

Rohan Siwach (1BM19CS132)

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 **Rohan Siwach(1BM19CS132),** 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.

Dr. Rajeshwari B S
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

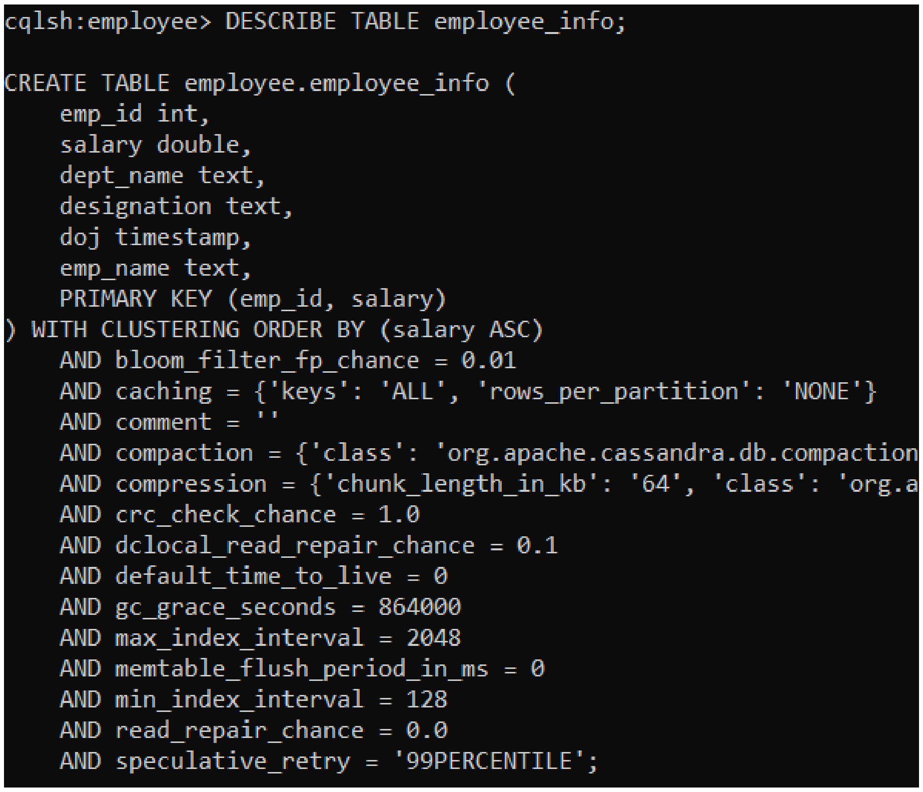
1. Perform the following DB operations using Cassandra. (Employee DB)

1.Create a keyspace by name Employee

create keyspace employee with replication = { 'class':'SimpleStrategy' , 'replication_factor' :1};

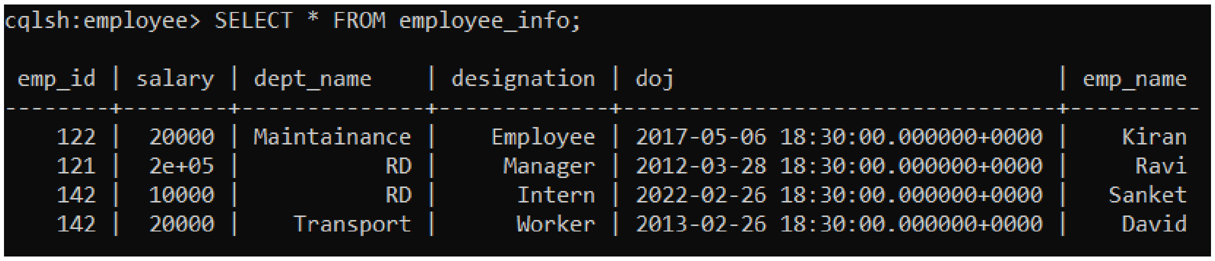
2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

create table employee_info(emp_id int,emp_name text, designation text, doj timestamp, salary double, dept_name text, primary key(emp_id,salary));

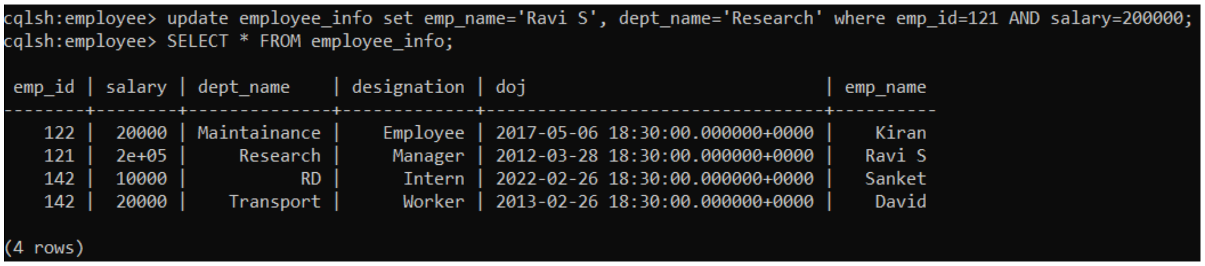

```
cqlsh:employee> DESCRIBE TABLE employee_info;

CREATE TABLE employee.employee_info (
    emp_id int,
    salary double,
    dept_name text,
    designation text,
    doj timestamp,
    emp_name text,
    PRIMARY KEY (emp_id, salary)
) WITH CLUSTERING ORDER BY (salary ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.a
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

3. Insert the values into the table in batch

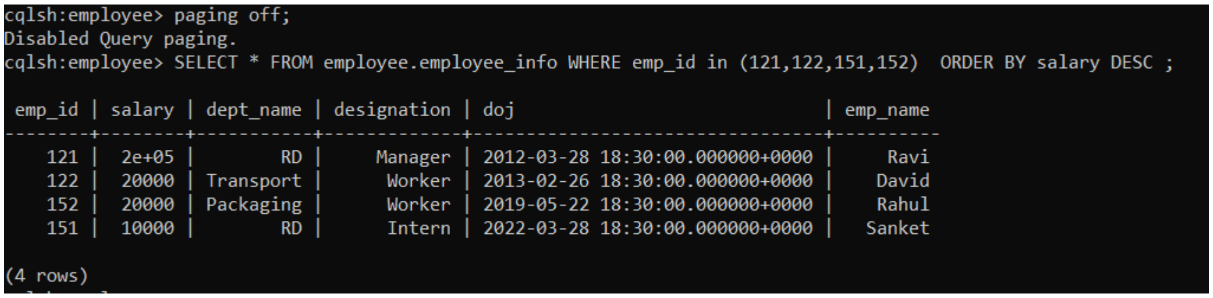
BEGIN BATCH
INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name) VALUES (121, 'Ravi', 'Manager', '2012-03-29', 200000, 'RD')
INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name) VALUES(122, 'David', 'Worker', '2013-02-27', 20000, 'Transport')
APPLY BATCH;


```
cqlsh:employee> SELECT * FROM employee_info;

 emp_id | salary | dept_name    | designation | doj                             | emp_name
--------+--------+--------------+-------------+---------------------------------+----------
    122 |  20000 | Maintainance |    Employee | 2017-05-06 18:30:00.000000+0000 |    Kiran
    121 |  2e+05 |           RD |     Manager | 2012-03-28 18:30:00.000000+0000 |     Ravi
    142 |  10000 |           RD |      Intern | 2022-02-26 18:30:00.000000+0000 |   Sanket
    142 |  20000 |    Transport |      Worker | 2013-02-26 18:30:00.000000+0000 |    David
```

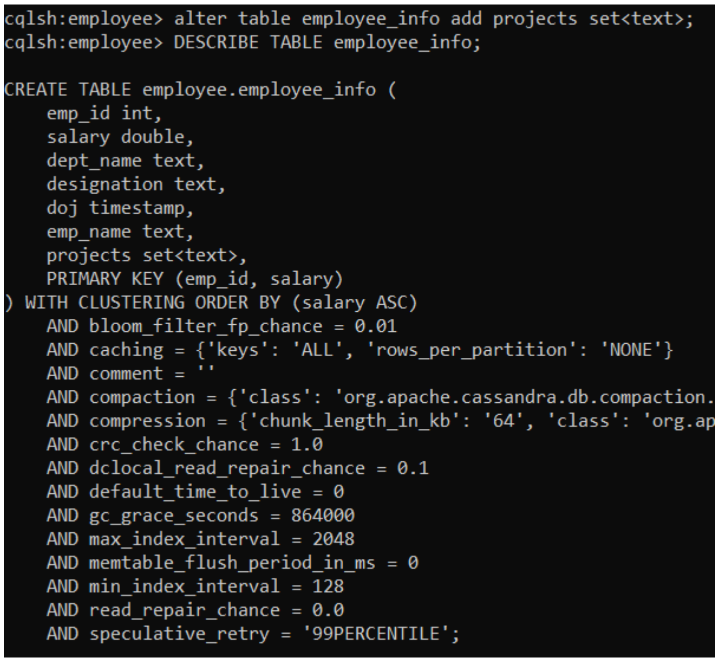
4. Update Employee name and Department of Emp-Id 121
update employee_info set emp_name='Ravi S', dept_name='Research' where emp_id=121 AND salary=200000;


```
cqlsh:employee> update employee_info set emp_name='Ravi S', dept_name='Research' where emp_id=121 AND salary=200000;
cqlsh:employee> SELECT * FROM employee_info;

 emp_id | salary | dept_name    | designation | doj                             | emp_name
--------+--------+--------------+-------------+---------------------------------+----------
    122 |  20000 | Maintainance |    Employee | 2017-05-06 18:30:00.000000+0000 |    Kiran
    121 |  2e+05 |     Research |     Manager | 2012-03-28 18:30:00.000000+0000 |   Ravi S
    142 |  10000 |           RD |      Intern | 2022-02-26 18:30:00.000000+0000 |   Sanket
    142 |  20000 |    Transport |      Worker | 2013-02-26 18:30:00.000000+0000 |    David

(4 rows)
```

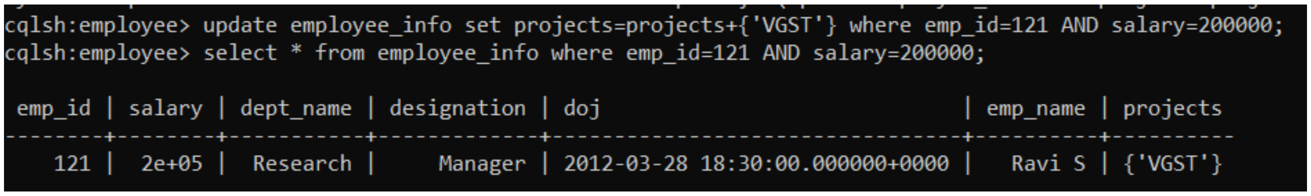
5. Sort the details of Employee records based on salary


```
cqlsh:employee> paging off;
Disabled Query paging.
cqlsh:employee> SELECT * FROM employee.employee_info WHERE emp_id in (121,122,151,152)  ORDER BY salary DESC ;

 emp_id | salary | dept_name | designation | doj                             | emp_name
--------+--------+-----------+-------------+---------------------------------+----------
    121 |  2e+05 |        RD |     Manager | 2012-03-28 18:30:00.000000+0000 |     Ravi
    122 |  20000 | Transport |      Worker | 2013-02-26 18:30:00.000000+0000 |    David
    152 |  20000 | Packaging |      Worker | 2019-05-22 18:30:00.000000+0000 |    Rahul
    151 |  10000 |        RD |      Intern | 2022-03-28 18:30:00.000000+0000 |   Sanket

(4 rows)
```

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.

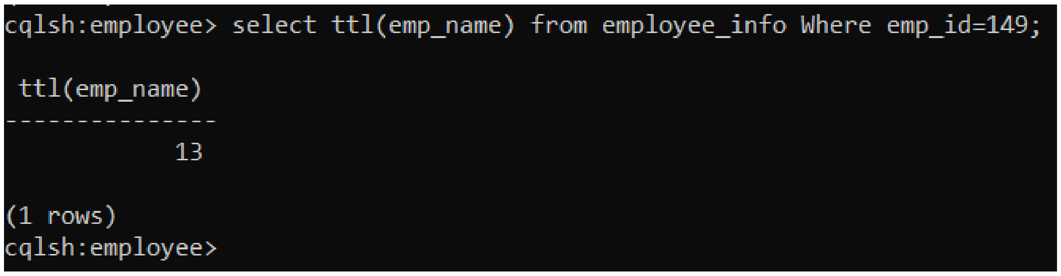

```
cqlsh:employee> alter table employee_info add projects set<text>;
cqlsh:employee> DESCRIBE TABLE employee_info;

CREATE TABLE employee.employee_info (
    emp_id int,
    salary double,
    dept_name text,
    designation text,
    doj timestamp,
    emp_name text,
    projects set<text>,
    PRIMARY KEY (emp_id, salary)
) WITH CLUSTERING ORDER BY (salary ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.ap
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

7. Update the altered table to add project names.
update employee_info set projects=projects+{'VGST'} where emp_id=121 AND salary=200000;



```
cqlsh:employee> update employee_info set projects=projects+{'VGST'} where emp_id=121 AND salary=200000;
cqlsh:employee> select * from employee_info where emp_id=121 AND salary=200000;

 emp_id | salary | dept_name | designation | doj                             | emp_name | projects
--------+--------+-----------+-------------+---------------------------------+----------+----------
    121 |  2e+05 |  Research |     Manager | 2012-03-28 18:30:00.000000+0000 |   Ravi S | {'VGST'}
```

8.Create a TTL of 15 seconds to display the values of Employee
cqlsh:employee> INSERT INTO employee_info(emp_id, emp_name, designation, doj, salary, dept_name) VALUES(149, 'Saket', 'Developer', '2021-02-20', 100000, 'RD') USING TTL 15;
cqlsh:employee> select ttl(emp_name) from employee_info Where emp_id=149;

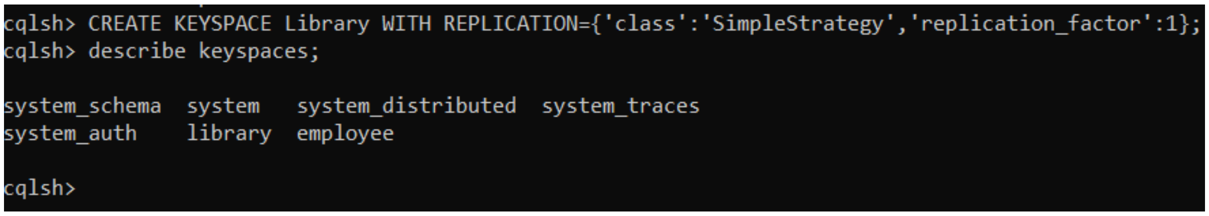


```
cqlsh:employee> select ttl(emp_name) from employee_info Where emp_id=149;

 ttl(emp_name)
---------------
            13

(1 rows)
cqlsh:employee>
```

2. Perform the following DB operations using Cassandra. (Library DB)

1. Create a keyspace by name Library

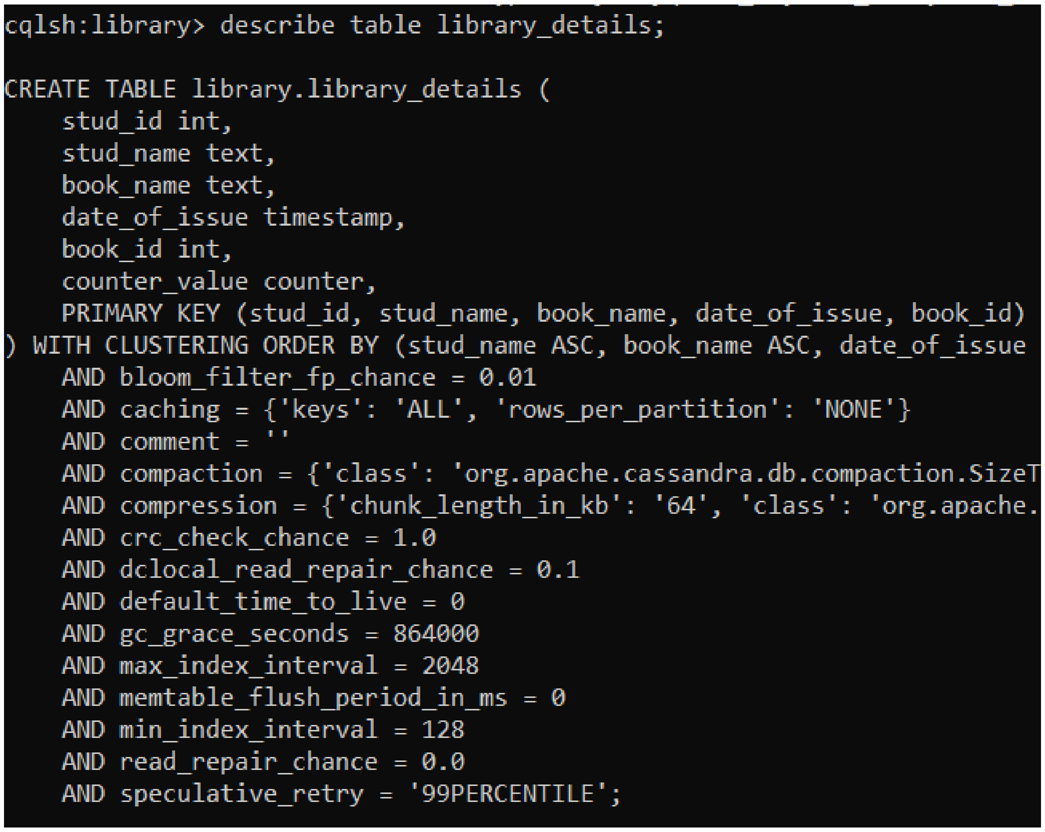
CREATE KEYSPACE Library WITH REPLICATION={'class':'SimpleStrategy','replication_factor':1};



```
cqlsh> CREATE KEYSPACE Library WITH REPLICATION={'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe keyspaces;

system_schema  system   system_distributed  system_traces
system_auth    library  employee

cqlsh>
```

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

create table library_details(stud_id int,counter_value counter,stud_name text,book_name text,date_of_issue timestamp,book_id int,primary key(stud_id,stud_name,book_name,date_of_issue,book_id));



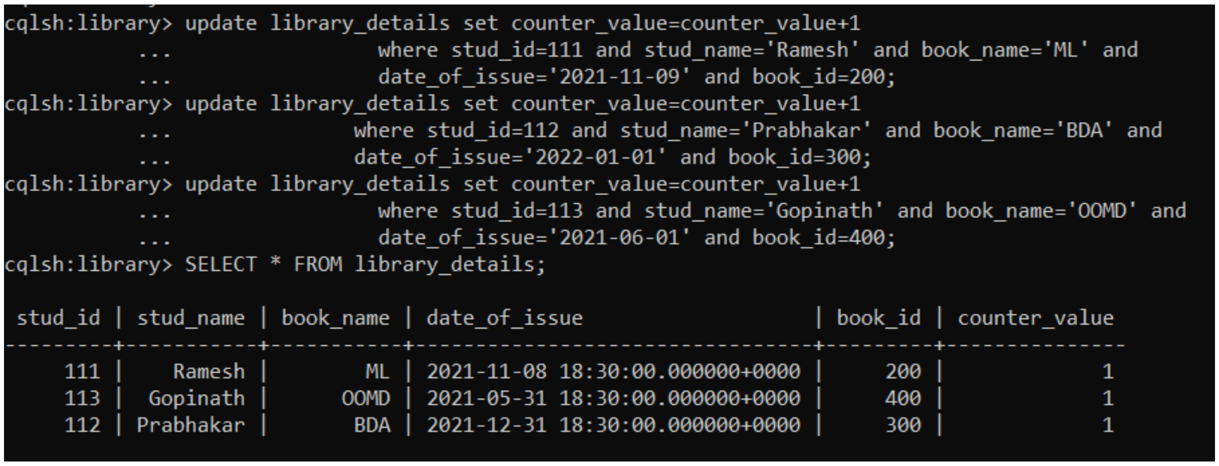
```
cqlsh:library> describe table library_details;

CREATE TABLE library.library_details (
    stud_id int,
    stud_name text,
    book_name text,
    date_of_issue timestamp,
    book_id int,
    counter_value counter,
    PRIMARY KEY (stud_id, stud_name, book_name, date_of_issue, book_id)
) WITH CLUSTERING ORDER BY (stud_name ASC, book_name ASC, date_of_issue
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeT
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

3. Insert the values into the table in batch

update library_details set counter_value=counter_value+1
where stud_id=111 and stud_name='Ramesh' and book_name='ML' and date_of_issue='2021-11-09' and

book_id=200;

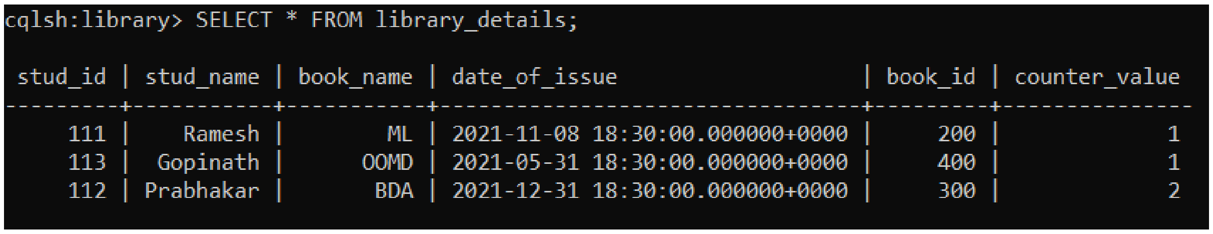
update library_details set counter_value=counter_value+1
where stud_id=112 and stud_name='Prabhakar' and book_name='BDA' and date_of_issue='2022-01-01' and book_id=300;
update library_details set counter_value=counter_value+1
where stud_id=113 and stud_name='Gopinath' and book_name='OOMD' and date_of_issue='2021-06-01' and book_id=400;



```
cqlsh:library> update library_details set counter_value=counter_value+1
           ...                 where stud_id=111 and stud_name='Ramesh' and book_name='ML' and
           ...                 date_of_issue='2021-11-09' and book_id=200;
cqlsh:library> update library_details set counter_value=counter_value+1
           ...               where stud_id=112 and stud_name='Prabhakar' and book_name='BDA' and
           ...               date_of_issue='2022-01-01' and book_id=300;
cqlsh:library> update library_details set counter_value=counter_value+1
           ...                 where stud_id=113 and stud_name='Gopinath' and book_name='OOMD' and
           ...                 date_of_issue='2021-06-01' and book_id=400;
cqlsh:library> SELECT * FROM library_details;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    Ramesh |        ML | 2021-11-08 18:30:00.000000+0000 |     200 |             1
     113 |  Gopinath |      OOMD | 2021-05-31 18:30:00.000000+0000 |     400 |             1
     112 | Prabhakar |       BDA | 2021-12-31 18:30:00.000000+0000 |     300 |             1
```

4. Display the details of the table created and increase the value of the counter

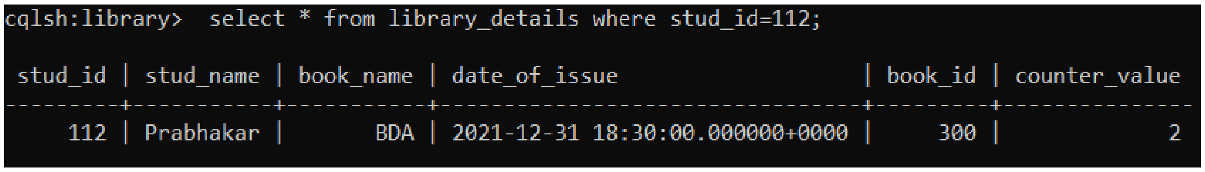
update library_details set counter_value=counter_value+1
where stud_id=112 and stud_name='Prabhakar' and book_name='BDA' and date_of_issue='2021-12-31' and book_id=300;



```
cqlsh:library> SELECT * FROM library_details;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    Ramesh |        ML | 2021-11-08 18:30:00.000000+0000 |     200 |             1
     113 |  Gopinath |      OOMD | 2021-05-31 18:30:00.000000+0000 |     400 |             1
     112 | Prabhakar |       BDA | 2021-12-31 18:30:00.000000+0000 |     300 |             2
```

5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

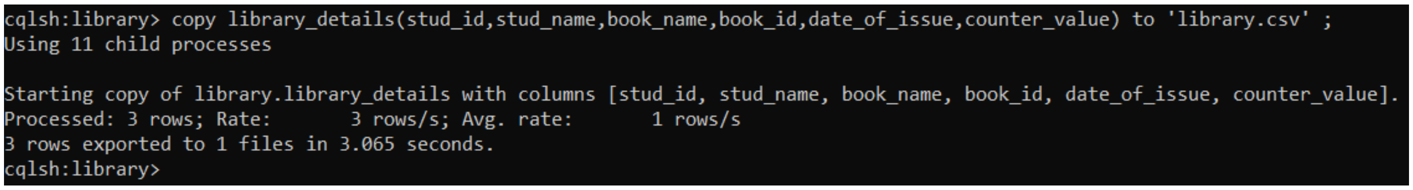
select * from library_details where stud_id=112;



```
cqlsh:library>  select * from library_details where stud_id=112;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     112 | Prabhakar |       BDA | 2021-12-31 18:30:00.000000+0000 |     300 |             2
```

6. Export the created column to a csv file

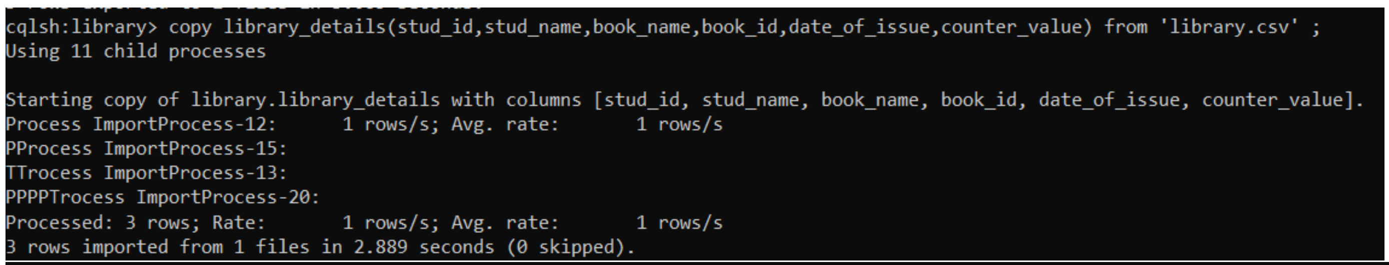
copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) to 'library.csv' ;



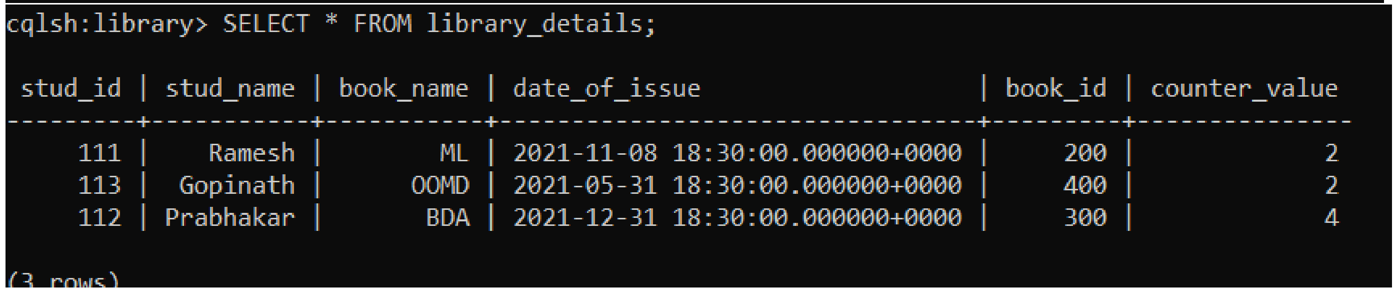
```
cqlsh:library> copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) to 'library.csv' ;
Using 11 child processes

Starting copy of library.library_details with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 3 rows; Rate:       3 rows/s; Avg. rate:       1 rows/s
3 rows exported to 1 files in 3.065 seconds.
cqlsh:library>
```

7. Import a given csv dataset from local file system into Cassandra column family

copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) from 'library.csv'
;



```
cqlsh:library> copy library_details(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) from 'library.csv' ;
Using 11 child processes

Starting copy of library.library_details with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Process ImportProcess-12:      1 rows/s; Avg. rate:       1 rows/s
PProcess ImportProcess-15:
TTrocess ImportProcess-13:
PPPPTrocess ImportProcess-20:
Processed: 3 rows; Rate:       1 rows/s; Avg. rate:       1 rows/s
3 rows imported from 1 files in 2.889 seconds (0 skipped).
```

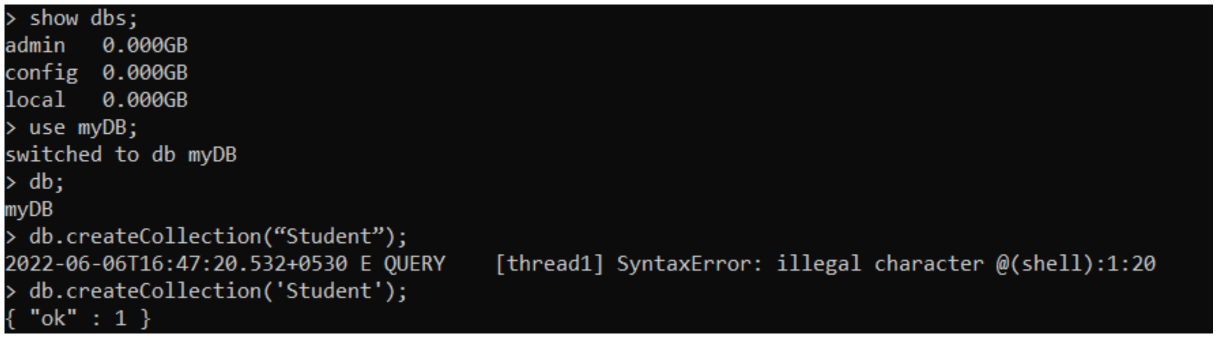


```
cqlsh:library> SELECT * FROM library_details;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    Ramesh |        ML | 2021-11-08 18:30:00.000000+0000 |     200 |             2
     113 |  Gopinath |      OOMD | 2021-05-31 18:30:00.000000+0000 |     400 |             2
     112 | Prabhakar |       BDA | 2021-12-31 18:30:00.000000+0000 |     300 |             4

(3 rows)
```

3.MongoDB- CRUD

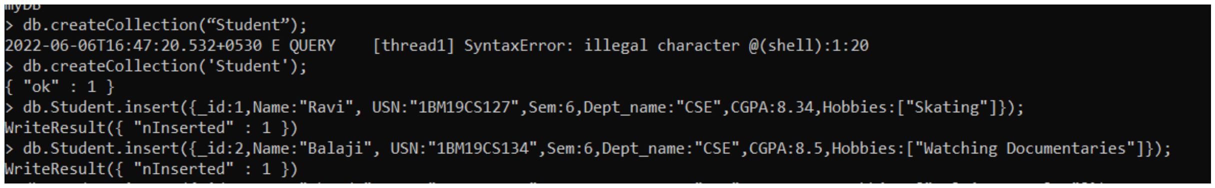
1) Using MongoDB

Demonstration



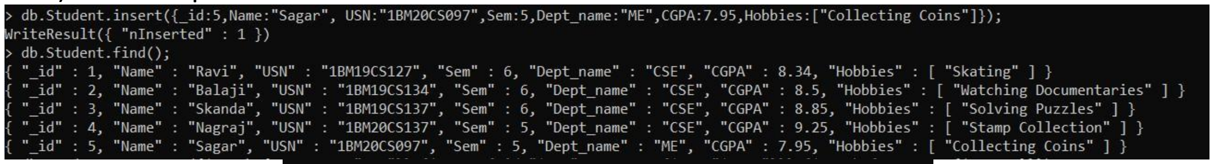
```
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
> use myDB;
switched to db myDB
> db;
myDB
> db.createCollection(“Student”);
2022-06-06T16:47:20.532+0530 E QUERY    [thread1] SyntaxError: illegal character @(shell):1:20
> db.createCollection('Student');
{ "ok" : 1 }
```

i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)).

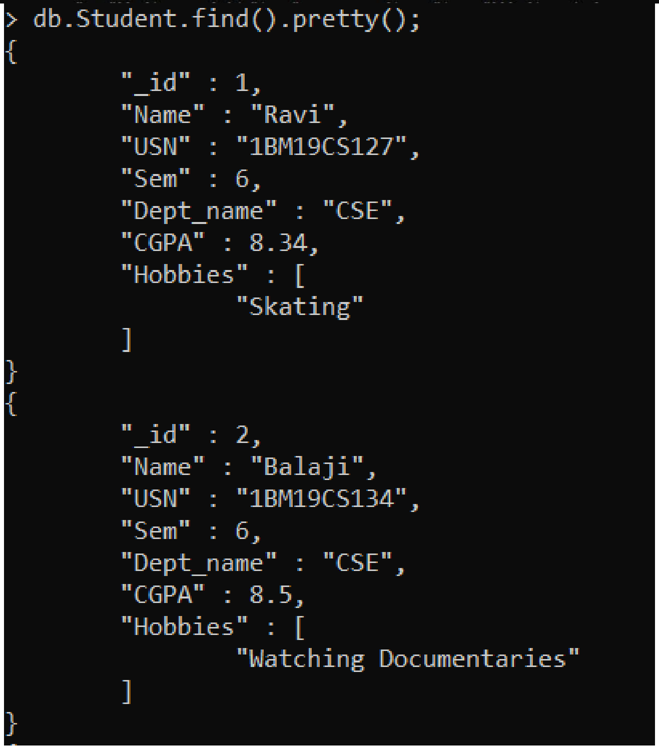


```
> db.createCollection(“Student”);
2022-06-06T16:47:20.532+0530 E QUERY    [thread1] SyntaxError: illegal character @(shell):1:20
> db.createCollection('Student');
{ "ok" : 1 }
> db.Student.insert({_id:1,Name:"Ravi", USN:"1BM19CS127",Sem:6,Dept_name:"CSE",CGPA:8.34,Hobbies:["Skating"]});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name:"Balaji", USN:"1BM19CS134",Sem:6,Dept_name:"CSE",CGPA:8.5,Hobbies:["Watching Documentaries"]});
WriteResult({ "nInserted" : 1 })
```

ii) Insert required documents to the collection.



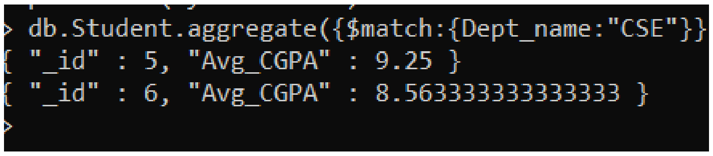
```
> db.Student.insert({_id:5,Name:"Sagar", USN:"1BM20CS097",Sem:5,Dept_name:"ME",CGPA:7.95,Hobbies:["Collecting Coins"]});
WriteResult({ "nInserted" : 1 })
> db.Student.find();
{ "_id" : 1, "Name" : "Ravi", "USN" : "1BM19CS127", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.34, "Hobbies" : [ "Skating" ] }
{ "_id" : 2, "Name" : "Balaji", "USN" : "1BM19CS134", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.5, "Hobbies" : [ "Watching Documentaries" ] }
{ "_id" : 3, "Name" : "Skanda", "USN" : "1BM19CS137", "Sem" : 6, "Dept_name" : "CSE", "CGPA" : 8.85, "Hobbies" : [ "Solving Puzzles" ] }
{ "_id" : 4, "Name" : "Nagraj", "USN" : "1BM20CS137", "Sem" : 5, "Dept_name" : "CSE", "CGPA" : 9.25, "Hobbies" : [ "Stamp Collection" ] }
{ "_id" : 5, "Name" : "Sagar", "USN" : "1BM20CS097", "Sem" : 5, "Dept_name" : "ME", "CGPA" : 7.95, "Hobbies" : [ "Collecting Coins" ] }
```



```
> db.Student.find().pretty();
{
        "_id" : 1,
        "Name" : "Ravi",
        "USN" : "1BM19CS127",
        "Sem" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.34,
        "Hobbies" : [
                "Skating"
        ]
}
{
        "_id" : 2,
        "Name" : "Balaji",
        "USN" : "1BM19CS134",
        "Sem" : 6,
        "Dept_name" : "CSE",
        "CGPA" : 8.5,
        "Hobbies" : [
                "Watching Documentaries"
        ]
}
```

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.

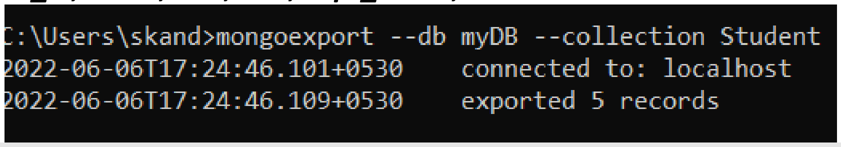
db.Student.aggregate({$match:{Dept_name:"CSE"}},{$group:{_id:"$Sem",Avg_CGPA:{$avg:"$CGPA"}}},{ $match:{Avg_CGPA:{$gt:7.5}}}).pretty();



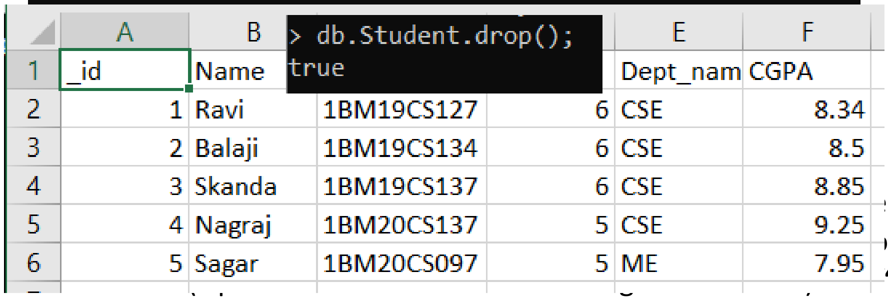
```
> db.Student.aggregate({$match:{Dept_name:"CSE"}}
{ "_id" : 5, "Avg_CGPA" : 9.25 }
{ "_id" : 6, "Avg_CGPA" : 8.563333333333333 }
>
```

iv) Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV file “Output.txt”.

mongoexport --db myDB --collection Student --type=csv --out C:\Users\skand\Desktop\Output.csv -f "_id,Name,USN,Sem,Dept_name,CGPA"



```
C:\Users\skand>mongoexport --db myDB --collection Student
2022-06-06T17:24:46.101+0530    connected to: localhost
2022-06-06T17:24:46.109+0530    exported 5 records
```

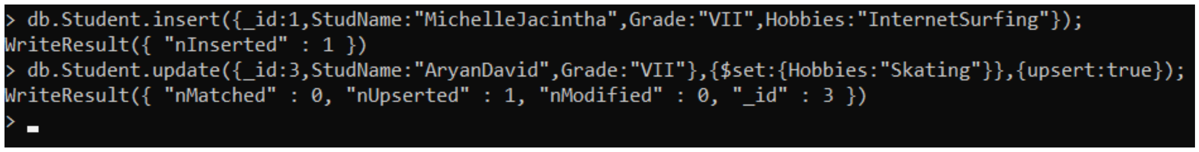


```
> db.Student.drop();
true
```

	A	B			E	F
1	_id	Name			Dept_nam	CGPA
2	1	Ravi	1BM19CS127	6	CSE	8.34
3	2	Balaji	1BM19CS134	6	CSE	8.5
4	3	Skanda	1BM19CS137	6	CSE	8.85
5	4	Nagraj	1BM20CS137	5	CSE	9.25
6	5	Sagar	1BM20CS097	5	ME	7.95

ction only if it does not already exist in
n, then update the document with new
‘Update else insert” (if there is an existing
document, it will attempt to update it, if there is no existing document then it will insert it).

db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});

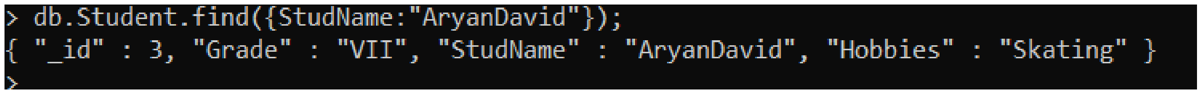


```
> db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
>
```

4.FIND METHOD

A. To search for documents from the “Students” collection based on certain search criteria.

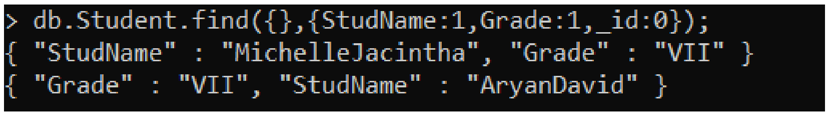
db.Student.find({StudName:"Aryan David"});



```
> db.Student.find({StudName:"AryanDavid"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

B. To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

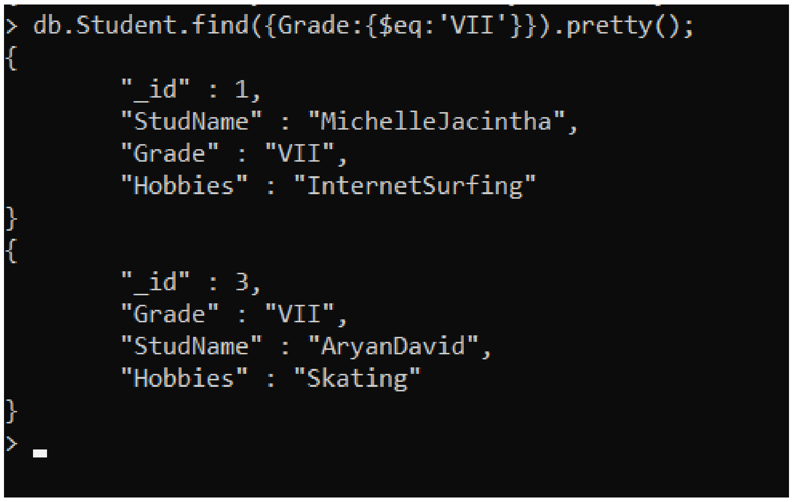
db.Student.find({},{StudName:1,Grade:1,_id:0});



```
> db.Student.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "AryanDavid" }
```

C. To find those documents where the Grade is set to ‘VII’

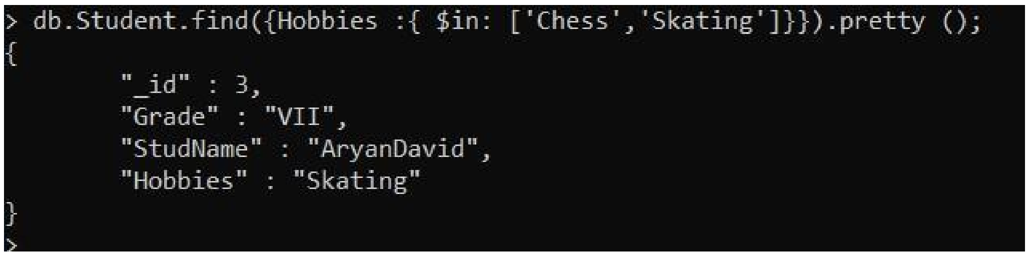
db.Student.find({Grade:{$eq:'VII'}}).pretty();



```
> db.Student.find({Grade:{$eq:'VII'}}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

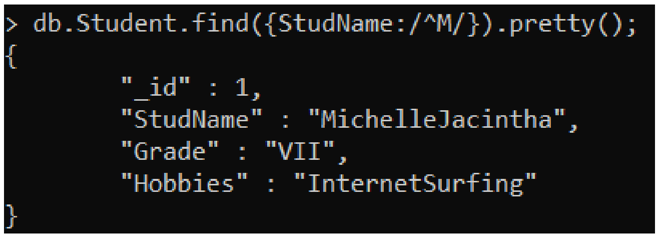
db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();



```
> db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

E. To find documents from the Students collection where the StudName begins with “M”.

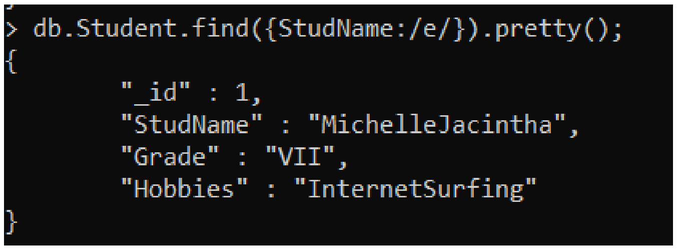
db.Student.find({StudName:/^M/}).pretty();



```
> db.Student.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
```

F. To find documents from the Students collection where the StudName has an “e” in any position.

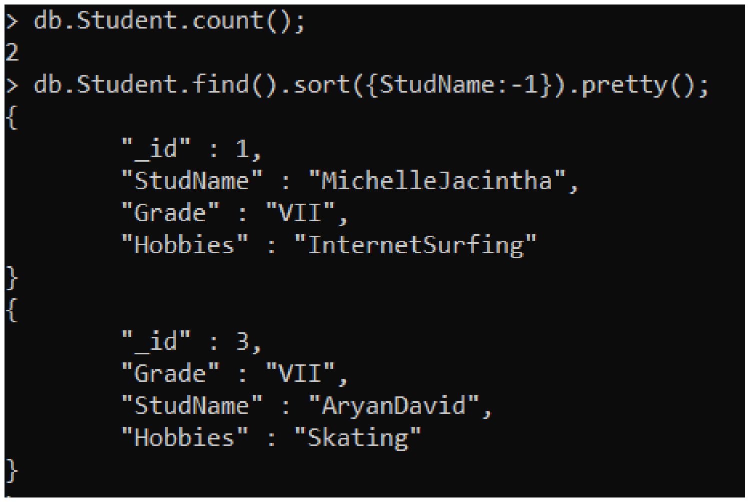
db.Student.find({StudName:/e/}).pretty();



```
> db.Student.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
```

G. To find the number of documents in the Students collection.
db.Student.count();

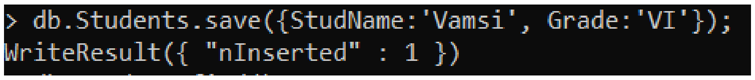
H. To sort the documents from the Students collection in the descending order of StudName.
db.Student.find().sort({StudName:-1}).pretty();



```
> db.Student.count();
2
> db.Student.find().sort({StudName:-1}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
```

I. Save Method :
Save() method will insert a new document, if the document with the _id does not exist. If it exists it will replace the exisiting document.

db.Students.save({StudName:”Vamsi”, Grade:”VI”})



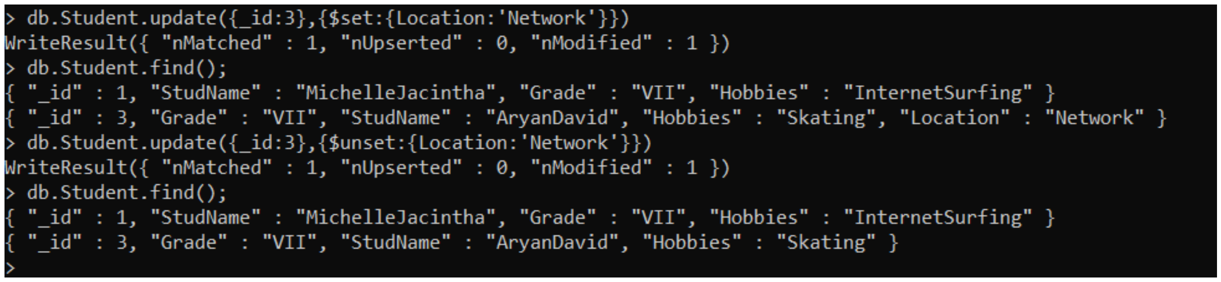
```
> db.Students.save({StudName:'Vamsi', Grade:'VI'});
WriteResult({ "nInserted" : 1 })
```

II. Add a new field to existing Document:

db.Students.update({_id:3},{$set:{Location:"Network"}})

III. Remove the field in an existing Document

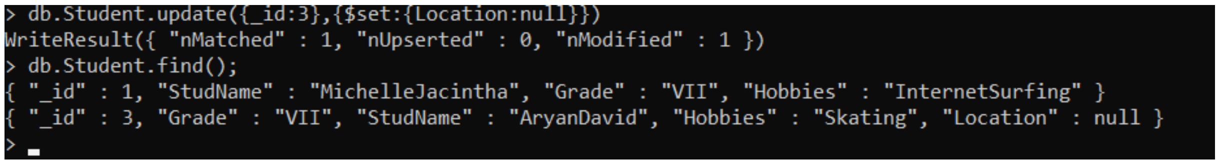
db.Students.update({_id:3},{$unset:{Location:"Network"}})



```
> db.Student.update({_id:3},{$set:{Location:'Network'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating", "Location" : "Network" }
> db.Student.update({_id:3},{$unset:{Location:'Network'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

To set a particular field value to NULL

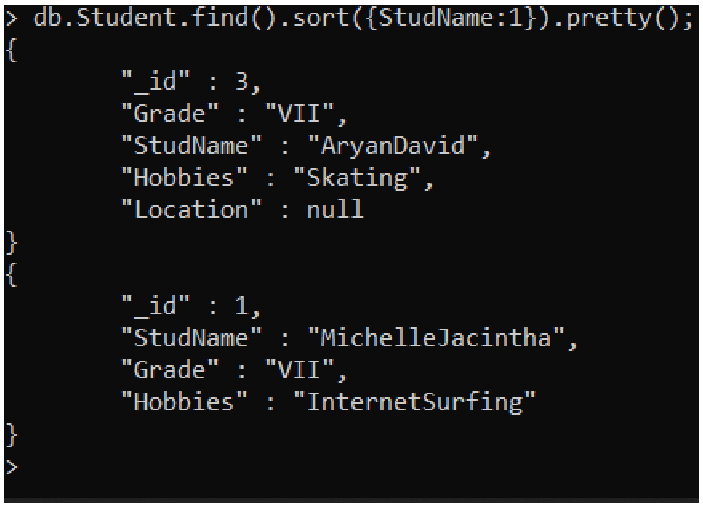
db.Students.update({_id:3},{$set:{Location:null}})



```
> db.Student.update({_id:3},{$set:{Location:null}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating", "Location" : null }
> _
```

Sort the document in Ascending order

db.Students.find().sort({StudName:1}).pretty();



```
> db.Student.find().sort({StudName:1}).pretty();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating",
        "Location" : null
}
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
>
```

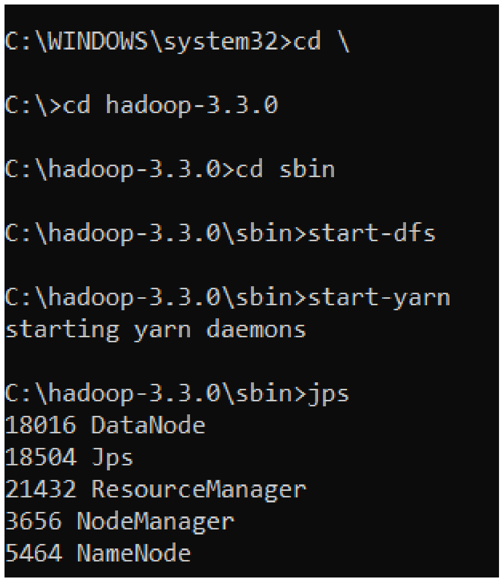
Note:
for desending order : db.Students.find().sort({StudName:-1}).pretty();

4.Screenshot of Hadoop installed

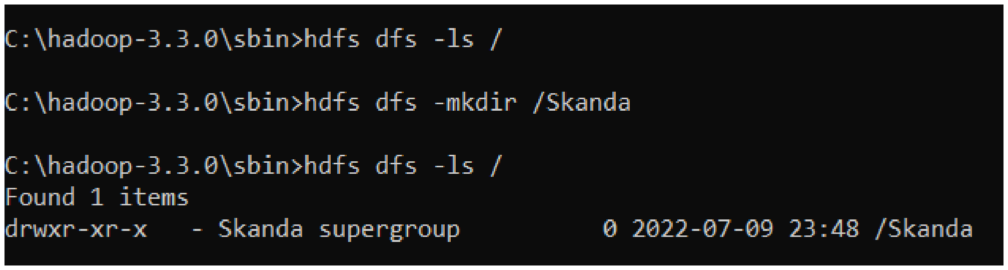
Steps to Install Hadoop

- Install Java JDK 1.8
- Download Hadoop and extract and place under C drive
- Set Path in Environment Variables
- Config files under Hadoop directory
- Create folder datanode and namenode under data directory
- Edit HDFS and YARN files
- Set Java Home environment in Hadoop environment
- Setup Complete. Test by executing start-all.cmd

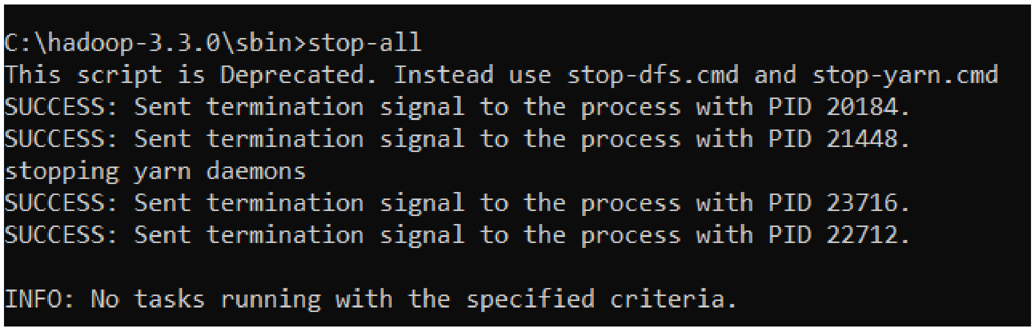
Screenshots



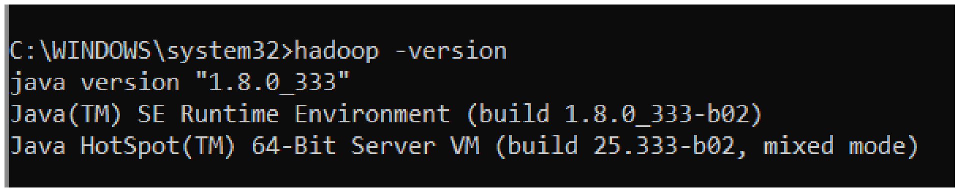
```
C:\WINDOWS\system32>cd \

C:\>cd hadoop-3.3.0

C:\hadoop-3.3.0>cd sbin

C:\hadoop-3.3.0\sbin>start-dfs

C:\hadoop-3.3.0\sbin>start-yarn
starting yarn daemons

C:\hadoop-3.3.0\sbin>jps
18016 DataNode
18504 Jps
21432 ResourceManager
3656 NodeManager
5464 NameNode
```



```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /Skanda

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - Skanda supergroup          0 2022-07-09 23:48 /Skanda
```



```
C:\hadoop-3.3.0\sbin>stop-all
This script is Deprecated. Instead use stop-dfs.cmd and stop-yarn.cmd
SUCCESS: Sent termination signal to the process with PID 20184.
SUCCESS: Sent termination signal to the process with PID 21448.
stopping yarn daemons
SUCCESS: Sent termination signal to the process with PID 23716.
SUCCESS: Sent termination signal to the process with PID 22712.

INFO: No tasks running with the specified criteria.
```



```
C:\WINDOWS\system32>hadoop -version
java version "1.8.0_333"
Java(TM) SE Runtime Environment (build 1.8.0_333-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.333-b02, mixed mode)
```

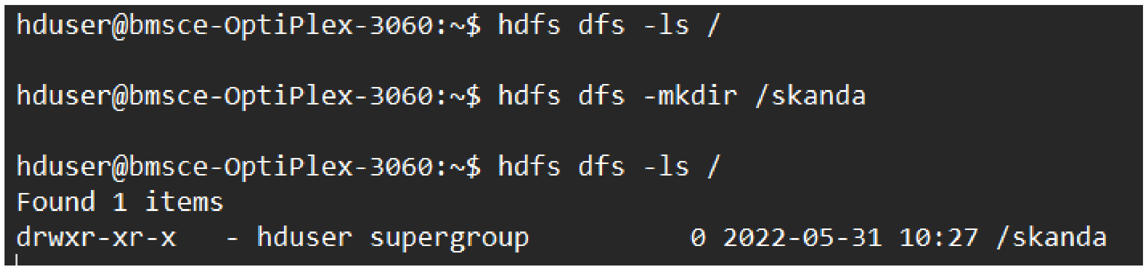
5.Execution of HDFS Commands

1. mkdir

Hadoop HDFS mkdir Command Example
hdfs dfs -mkdir /abc
Hadoop HDFS mkdir Command Description

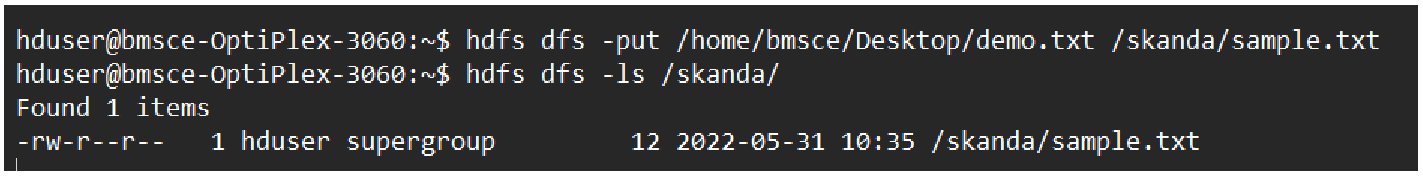
This HDFS command takes path URI's as an argument and creates directories.

2. ls
Hadoop HDFS ls Command Example
hadoop fs -ls /
This Hadoop HDFS ls command displays a list of the contents of a directory specified by path provided by the user, showing the names, permissions, owner, size and modification date for each entry.



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /

hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -mkdir /skanda

hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:27 /skanda
```

3. put
Hadoop HDFS put Command Example
hdfs dfs -put /home/hduser/Desktop/Welcome.txt /abc/WC.txt
This hadoop basic command copies the file or directory from the local file system to the destination within the DFS.
Display the contents of the file WC.txt
hdfs dfs -cat /abc/WC.txt



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -put /home/bmsce/Desktop/demo.txt /skanda/sample.txt
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /skanda/
Found 1 items
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:35 /skanda/sample.txt
```

4. copyFromLocal

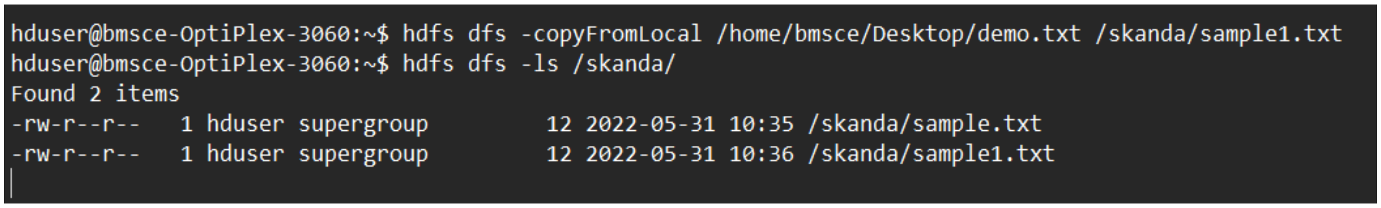
Hadoop HDFS copyFromLocal Command Example

hdfs dfs -put /home/hduser/Desktop/Welcome.txt /abc/WC.txt

This hadoop shell command is similar to put command, but the source is restricted to a local file reference.

Display the contents of the file WC2.txt

hdfs dfs -cat /abc/WC2.txt



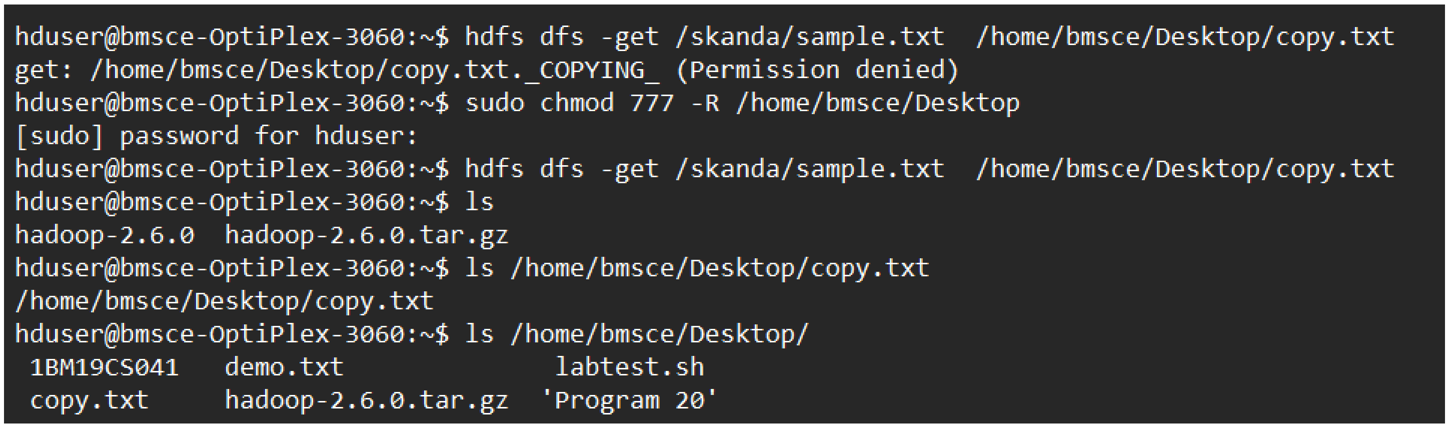
```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -copyFromLocal /home/bmsce/Desktop/demo.txt /skanda/sample1.txt
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /skanda/
Found 2 items
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:35 /skanda/sample.txt
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:36 /skanda/sample1.txt
```

5. get

i. Hadoop HDFS get Command Example

hdfs dfs -get /abc/WC.txt /home/hduser/Downloads/WWC.txt

This HDFS fs command copies the file or directory in HDFS identified by the source to the local file system path identified by local destination.



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -get /skanda/sample.txt  /home/bmsce/Desktop/copy.txt
get: /home/bmsce/Desktop/copy.txt._COPYING_ (Permission denied)
hduser@bmsce-OptiPlex-3060:~$ sudo chmod 777 -R /home/bmsce/Desktop
[sudo] password for hduser:
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -get /skanda/sample.txt  /home/bmsce/Desktop/copy.txt
hduser@bmsce-OptiPlex-3060:~$ ls
hadoop-2.6.0  hadoop-2.6.0.tar.gz
hduser@bmsce-OptiPlex-3060:~$ ls /home/bmsce/Desktop/copy.txt
/home/bmsce/Desktop/copy.txt
hduser@bmsce-OptiPlex-3060:~$ ls /home/bmsce/Desktop/
 1BM19CS041   demo.txt              labtest.sh
 copy.txt     hadoop-2.6.0.tar.gz  'Program 20'
```

ii. Hadoop HDFS get Command Example

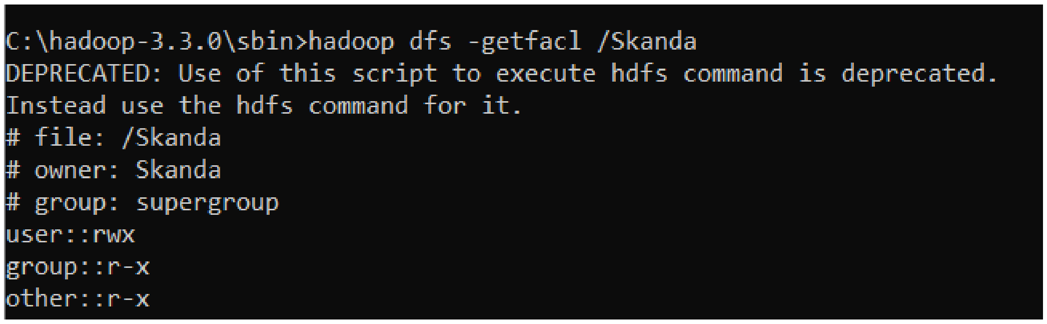
hdfs dfs -getmerge /abc/WC.txt /abc/WC2.txt /home/hduser/Desktop/Merge.txt

This HDFS basic command retrieves all files that match to the source path entered by the user in HDFS, and creates a copy of them to one single, merged file in the local file system identified by local destination.

iii. Hadoop HDFS get Command

Example hadoop fs -getfacl /abc/

This Apache Hadoop command shows the Access Control Lists (ACLs) of files and directories.



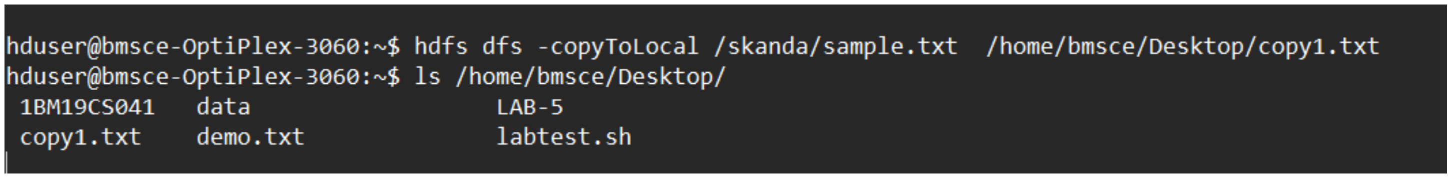
```
C:\hadoop-3.3.0\sbin>hadoop dfs -getfacl /Skanda
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
# file: /Skanda
# owner: Skanda
# group: supergroup
user::rwx
group::r-x
other::r-x
```

6. copyToLocal

Hadoop HDFS copyToLocal Command Example

hdfs dfs -copyToLocal /abc/WC.txt /home/hduser/Desktop

Similar to get command, only the difference is that in this the destination is restricted to a local file reference.



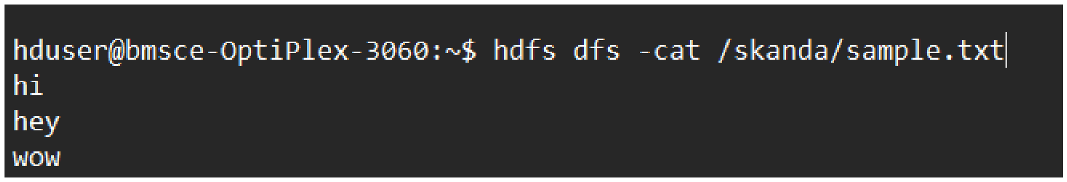
```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -copyToLocal /skanda/sample.txt  /home/bmsce/Desktop/copy1.txt
hduser@bmsce-OptiPlex-3060:~$ ls /home/bmsce/Desktop/
 1BM19CS041   data                 LAB-5
 copy1.txt    demo.txt             labtest.sh
```

7. cat

Hadoop HDFS cat Command Example

hdfs dfs -cat /abc/WC.txt

This Hadoop fs shell command displays the contents of the filename on console or stdout.



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -cat /skanda/sample.txt
hi
hey
wow
```

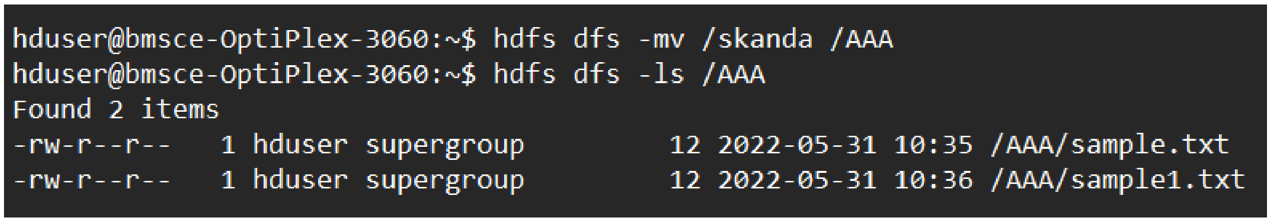
8. mv

Hadoop HDFS mv Command Example

hadoop fs -mv /abc /FFF

hadoop fs -ls /FFF

This basic HDFS command moves the file or directory indicated by the source to destination, within HDFS.



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -mv /skanda /AAA
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /AAA
Found 2 items
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:35 /AAA/sample.txt
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:36 /AAA/sample1.txt
```

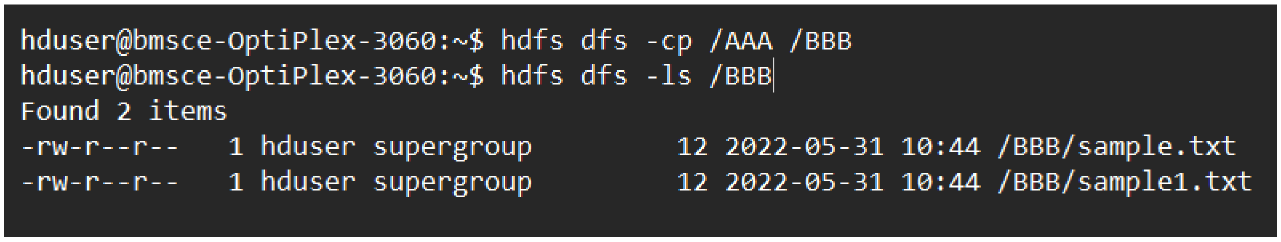
9. cp

Hadoop HDFS cp Command Example

hadoop fs -cp /CSE/ /LLL

hadoop fs -ls /LLL

The cp command copies a file from one directory to another directory within the HDFS.



```
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -cp /AAA /BBB
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /BBB
Found 2 items
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:44 /BBB/sample.txt
-rw-r--r--   1 hduser supergroup         12 2022-05-31 10:44 /BBB/sample1.txt
```

6. Map Reduce program to

a) Find average temperature for each year from NCDC data set.

AverageDriver

```
package temp;

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);
  }
}
```

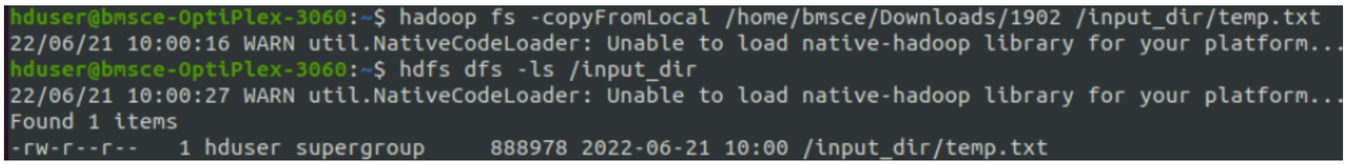
AverageMapper

```
package temp;

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));
  }
}
```

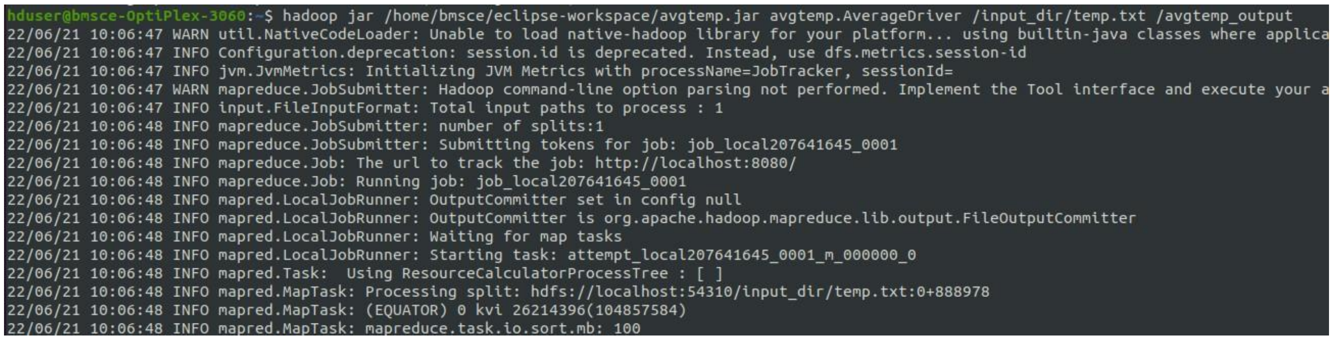
AverageReducer

```
package temp;

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));
  }
}
```

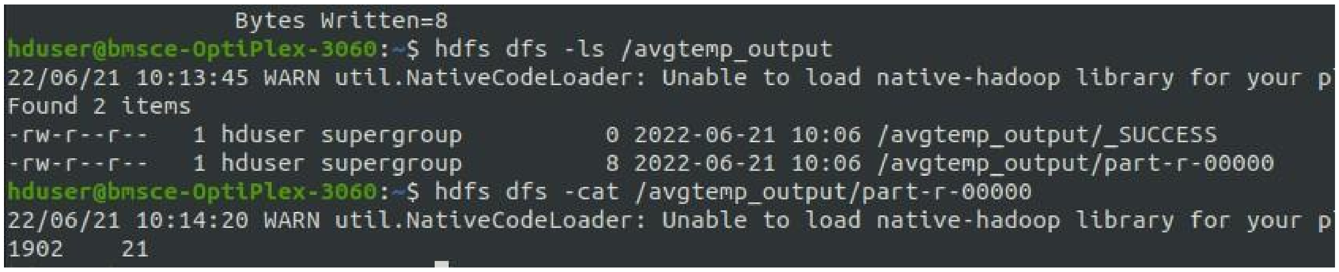
OUTPUT



```
hduser@bmsce-OptiPlex-3060:~$ hadoop fs -copyFromLocal /home/bmsce/Downloads/1902 /input_dir/temp.txt
22/06/21 10:00:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform..
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /input_dir
22/06/21 10:00:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform..
Found 1 items
-rw-r--r--   1 hduser supergroup     888978 2022-06-21 10:00 /input_dir/temp.txt
```



```
hduser@bmsce-OptiPlex-3060:~$ hadoop jar /home/bmsce/eclipse-workspace/avgtemp.jar avgtemp.AverageDriver /input_dir/temp.txt /avgtemp_output
22/06/21 10:06:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
22/06/21 10:06:47 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
22/06/21 10:06:47 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
22/06/21 10:06:47 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your a
22/06/21 10:06:47 INFO input.FileInputFormat: Total input paths to process : 1
22/06/21 10:06:48 INFO mapreduce.JobSubmitter: number of splits:1
22/06/21 10:06:48 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local207641645_0001
22/06/21 10:06:48 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
22/06/21 10:06:48 INFO mapreduce.Job: Running job: job_local207641645_0001
22/06/21 10:06:48 INFO mapred.LocalJobRunner: OutputCommitter set in config null
22/06/21 10:06:48 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
22/06/21 10:06:48 INFO mapred.LocalJobRunner: Waiting for map tasks
22/06/21 10:06:48 INFO mapred.LocalJobRunner: Starting task: attempt_local207641645_0001_m_000000_0
22/06/21 10:06:48 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
22/06/21 10:06:48 INFO mapred.MapTask: Processing split: hdfs://localhost:54310/input_dir/temp.txt:0+888978
22/06/21 10:06:48 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
22/06/21 10:06:48 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
```



```
                Bytes Written=8
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -ls /avgtemp_output
22/06/21 10:13:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your p
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2022-06-21 10:06 /avgtemp_output/_SUCCESS
-rw-r--r--   1 hduser supergroup          8 2022-06-21 10:06 /avgtemp_output/part-r-00000
hduser@bmsce-OptiPlex-3060:~$ hdfs dfs -cat /avgtemp_output/part-r-00000
22/06/21 10:14:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your p
1902    21
```

b) Find the mean max temperature for every month

MeanMaxDriver.class

```
package meanmax;

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);
  }
}
```

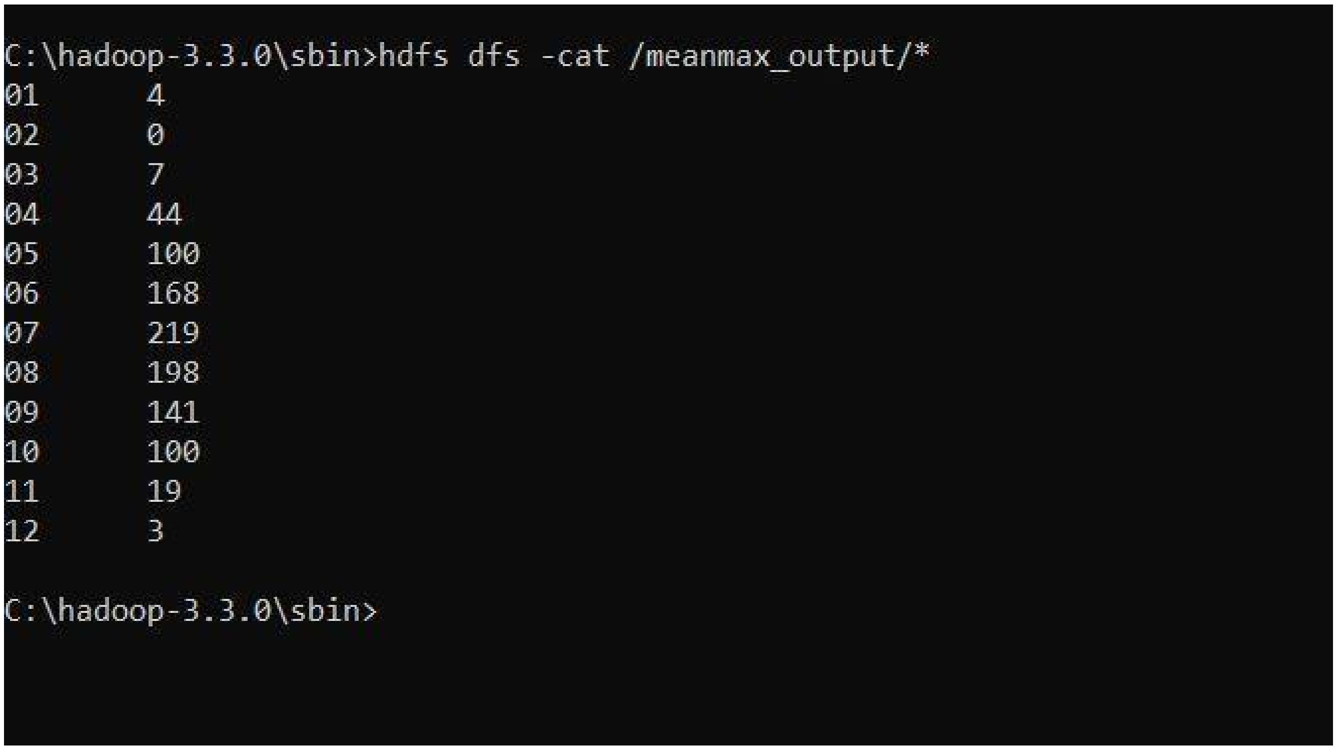
MeanMaxMapper.class

```
package meanmax;

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.class

```
package meanmax;

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));
  }
}
```

Output



```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /meanmax_output/*
01      4
02      0
03      7
04      44
05      100
06      168
07      219
08      198
09      141
10      100
11      19
12      3

C:\hadoop-3.3.0\sbin>
```

7. Map Reduce program to sort

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.

Driver-TopN.class

```
package samples.topn;

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);
  }
```

TopNCombiner.class

```
package samples.topn;

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.class

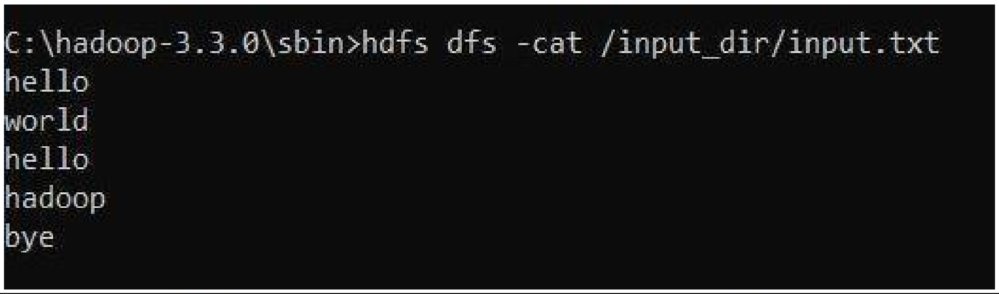
```
package samples.topn;

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 vo```\\id 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);
                                             whi
                                               l
                                               e
```

```
(itr.hasMoreTokens()){this.word.set(itr.nextToken().trim());context.write(this.word,one);
```

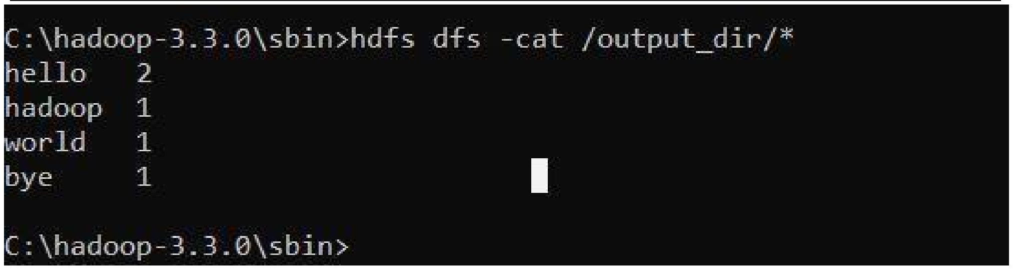
```
    }
  }
}
```

TopNReducer.class

```
package samples.topn;

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));
    }
  }
}
```



```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input.txt
hello
world
hello
hadoop
bye
```



```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /output_dir/*
hello   2
hadoop  1
world   1
bye     1

C:\hadoop-3.3.0\sbin>
```

8． Map Reduce program to demonstrate join operation

```
// 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);
```

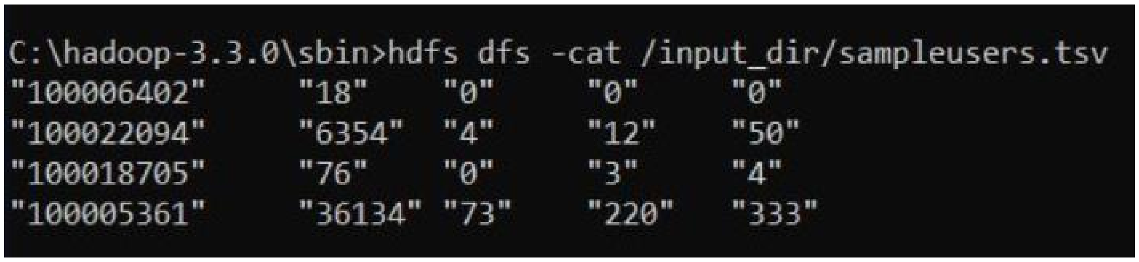
```
}
```

```
// ^^ 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(WritableComparable a, WritableComparable b) {
if (a instanceof TextPair && b instanceof TextPair) {
```

```
return ((TextPair) a).first.compareTo(((TextPair) b).first);
}
return super.compare(a, b);
}
} }
```

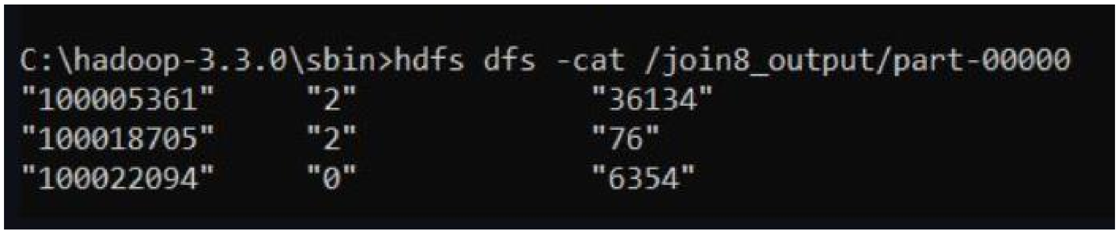
Output

Given input:



```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/sampleusers.tsv
"100006402"     "18"    "0"     "0"     "0"
"100022094"     "6354"  "4"     "12"    "50"
"100018705"     "76"    "0"     "3"     "4"
"100005361"     "36134" "73"    "220"   "333"
```

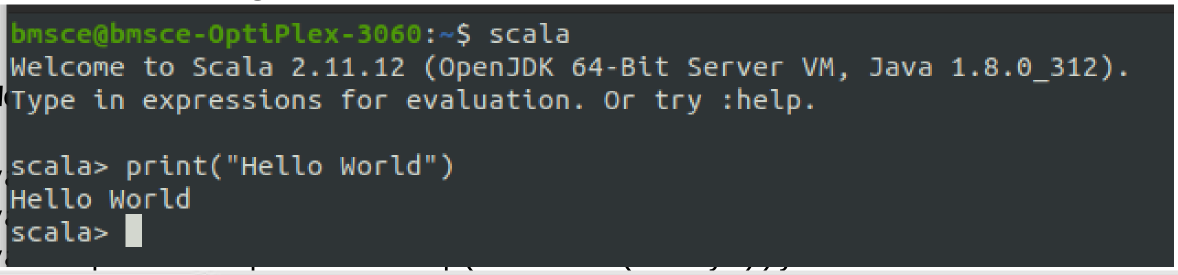
Produced Output:



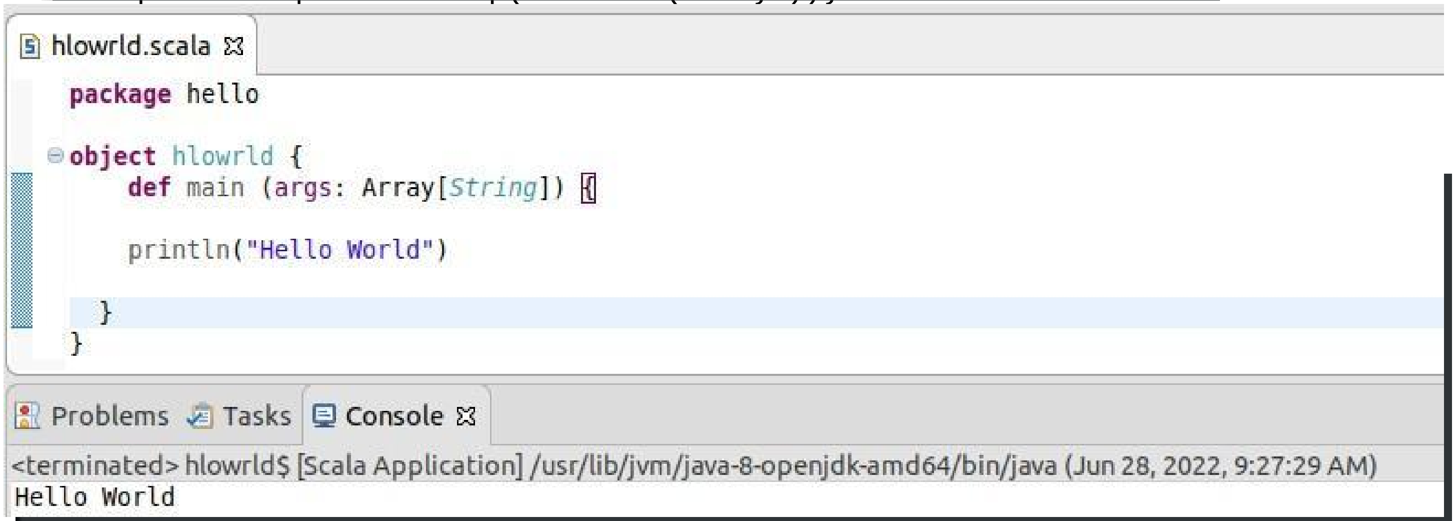
```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /join8_output/part-00000
"100005361"     "2"             "36134"
"100018705"     "2"             "76"
"100022094"     "0"             "6354"
```

9. Hello world (IDE) and word count on scala shell

Hello World Program

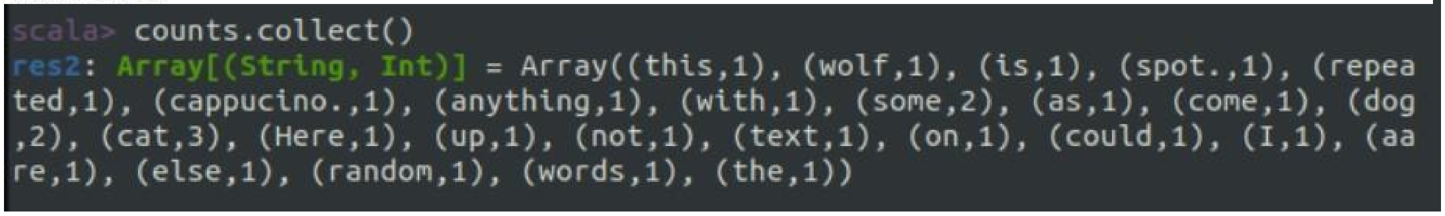


```
bmsce@bmsce-OptiPlex-3060:~$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_312).
Type in expressions for evaluation. Or try :help.

scala> print("Hello World")
Hello World
scala> 
```



```
package hello

object hlowrld {
    def main (args: Array[String]) {

    println("Hello World")

  }
}
```

```
<terminated> hlowrld$ [Scala Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jun 28, 2022, 9:27:29 AM)
Hello World
```



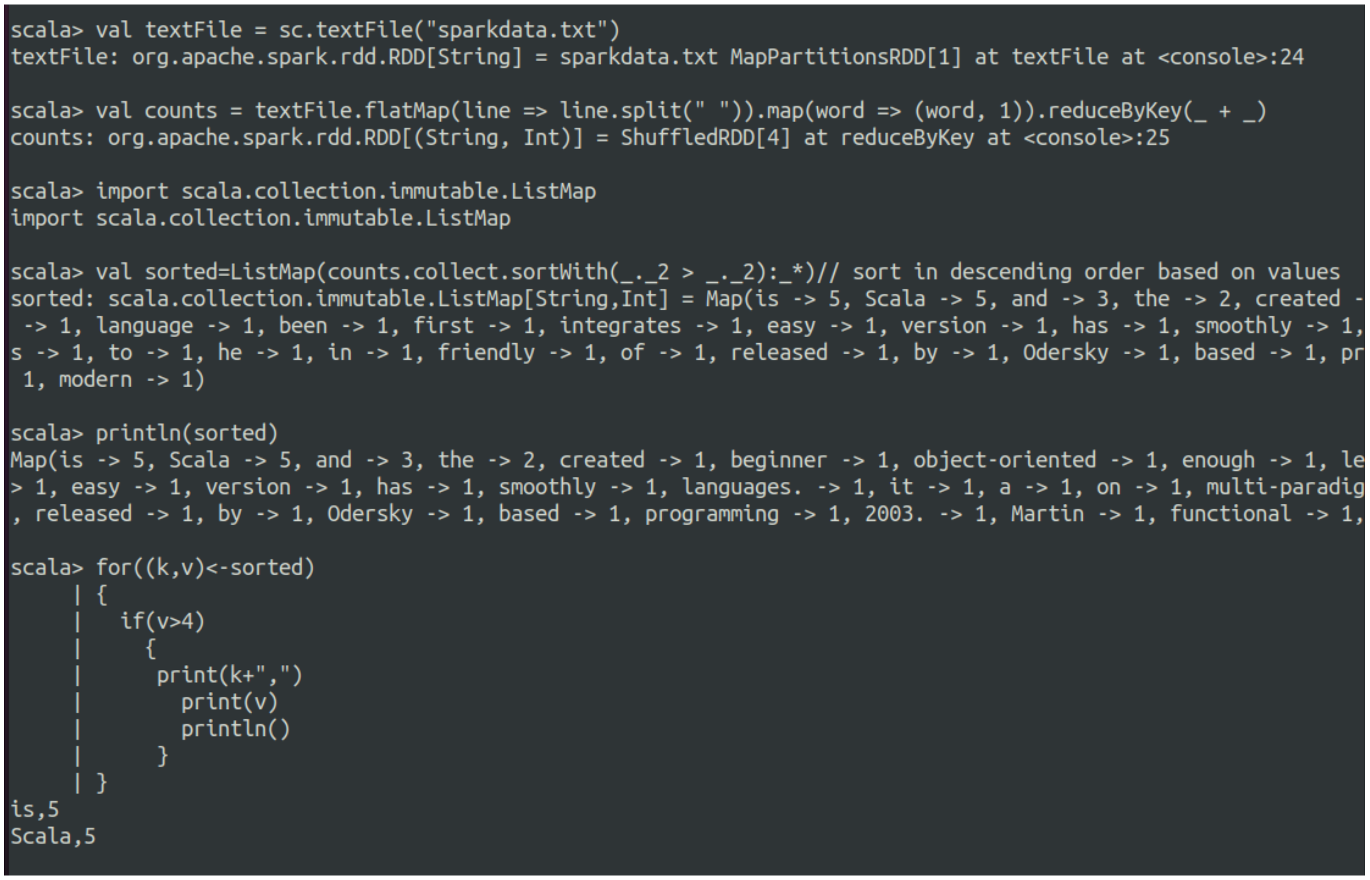
```
scala> counts.collect()
res2: Array[(String, Int)] = Array((this,1), (wolf,1), (is,1), (spot.,1), (repea
ted,1), (cappucino.,1), (anything,1), (with,1), (some,2), (as,1), (come,1), (dog
,2), (cat,3), (Here,1), (up,1), (not,1), (text,1), (on,1), (could,1), (I,1), (aa
re,1), (else,1), (random,1), (words,1), (the,1))
```

10. Words whose count is strictly greater than 4

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

```
val textFile = sc.textFile("sparkdata.txt")
val counts = textFile.flatMap(line => line.split(" ")).map(word => (word,
1)).reduceByKey(_ + _)
import scala.collection.immutable.ListMap
val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)// sort in
descending order based on values
for((k,v)<-sorted)
     | {
     |   if(v>4)
     |     {
     |      print(k+",")
     |        print(v)
     |        println()
     |      }
     | }
```

Output



```
scala> val textFile = sc.textFile("sparkdata.txt")
textFile: org.apache.spark.rdd.RDD[String] = sparkdata.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:25

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

scala> val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)// sort in descending order based on values
sorted: scala.collection.immutable.ListMap[String,Int] = Map(is -> 5, Scala -> 5, and -> 3, the -> 2, created -
 -> 1, language -> 1, been -> 1, first -> 1, integrates -> 1, easy -> 1, version -> 1, has -> 1, smoothly -> 1,
s -> 1, to -> 1, he -> 1, in -> 1, friendly -> 1, of -> 1, released -> 1, by -> 1, Odersky -> 1, based -> 1, pr
 1, modern -> 1)

scala> println(sorted)
Map(is -> 5, Scala -> 5, and -> 3, the -> 2, created -> 1, beginner -> 1, object-oriented -> 1, enough -> 1, le
> 1, easy -> 1, version -> 1, has -> 1, smoothly -> 1, languages. -> 1, it -> 1, a -> 1, on -> 1, multi-paradig
, released -> 1, by -> 1, Odersky -> 1, based -> 1, programming -> 1, 2003. -> 1, Martin -> 1, functional -> 1,

scala> for((k,v)<-sorted)
     | {
     |   if(v>4)
     |     {
     |      print(k+",")
     |        print(v)
     |        println()
     |      }
     | }
is,5
Scala,5
```