VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data Analytics

Submitted by

Mohammed Zeeshan Umar (1BM22CS160)

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
Feb-2025 to July-2025

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

CERTIFICATE

This is to certify that the Lab work entitled "LAB COURSE **Big Data Analytics**" carried out by **Mohammed Zeeshan Umar (1BM22CS160),** 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 2024. The Lab report has been approved as it satisfies the academic requirements in respect of a **Big Data Analytics - (23CS6PCBDA)** work prescribed for the said degree.

Amruta
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

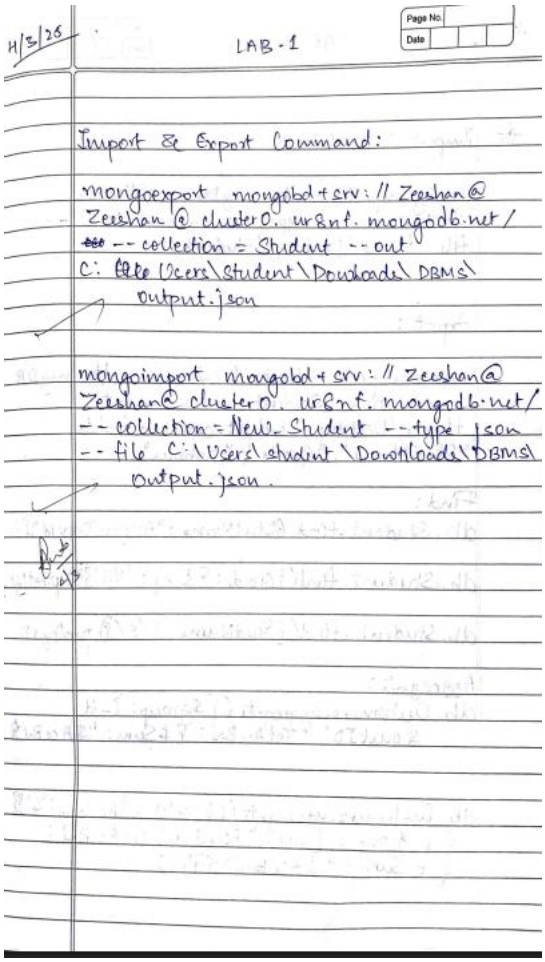
Index Sheet

Course Outcome

CO1	Apply the concepts of NoSQL, Hadoop, Spark for a given task
CO2	Analyse data analytic techniques for a given problem .
CO3	Conduct experiments using data analytics mechanisms for a given problem.

Experiment – 1

MongoDB- CRUD Operations Demonstration (Practice and Self Study)

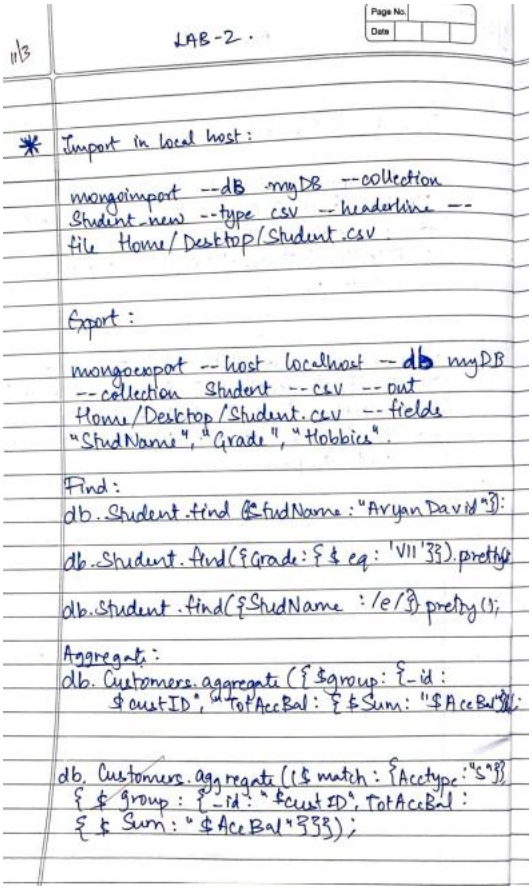


LAB - 1

Import & Export Command:

mongoexport mongodb+srv://Zeeshan@Zeeshan@cluster0.ur8nf.mongodb.net/ --collection=Student --out C:\Users\Student\Downloads\DBMS\output.json

mongoimport mongodb+srv://Zeeshan@Zeeshan@cluster0.ur8nf.mongodb.net/ --collection=New_Student --type json --file C:\Users\student\Downloads\DBMS\output.json.



LAB-2.

* Import in local host:

mongoimport --db myDB --collection Student_new --type csv --headerline --file Home/Desktop/Student.csv

Export:

mongoexport --host localhost --db myDB --collection Student --csv --out Home/Desktop/Student.csv --fields "StudName", "Grade", "Hobbies".

Find:

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

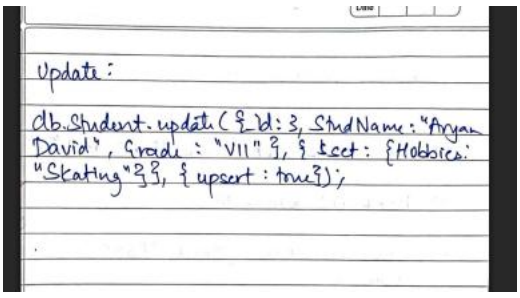
db.Student.find({Grade: {$eq: 'VII'}}).pretty();

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

Aggregate:

db.Customers.aggregate({$group: {_id: $custID, TotAccBal: {$Sum: "$AccBal"}}});

db.Customers.aggregate({$match: {Acctype: "S"}}, {$group: {_id: "$custID", TotAccBal: {$Sum: "$AccBal"}}});



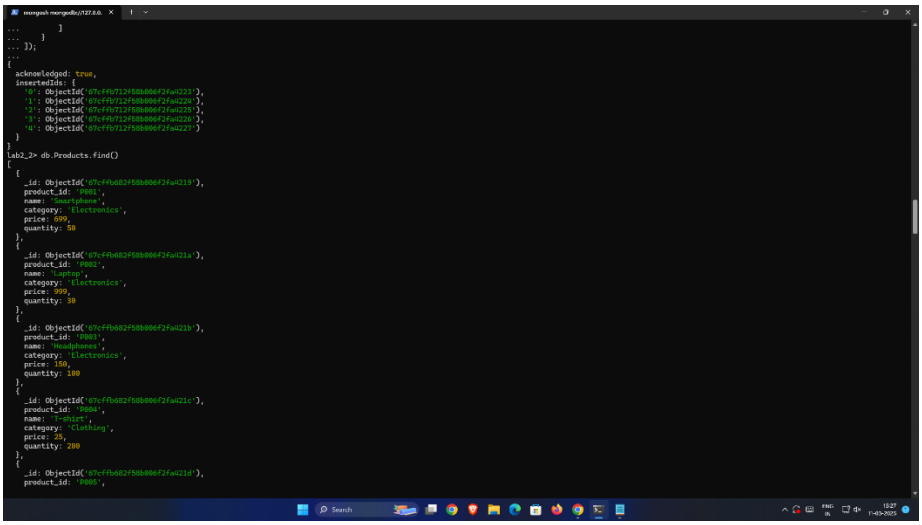
Update:

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

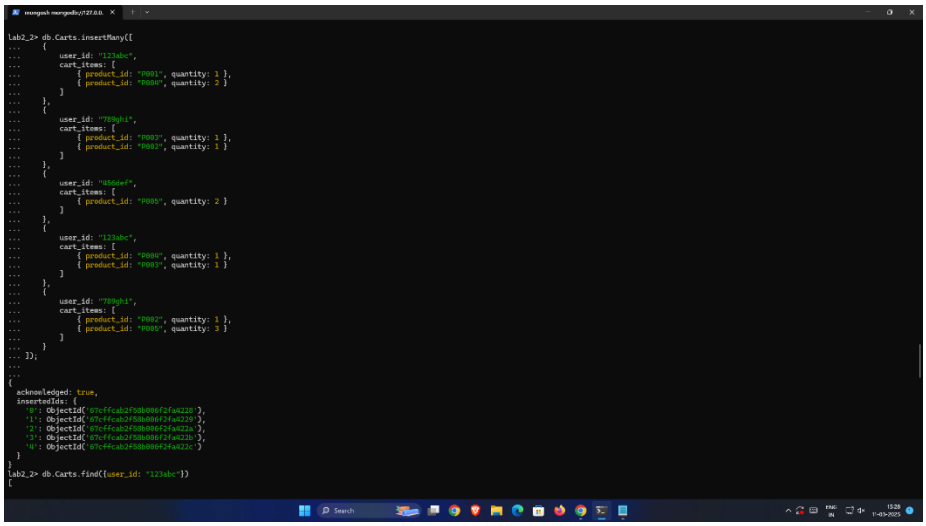
Code Outputs:

```
Atlas atlas-wanmtx-shard-0 [primary] Student> use Students
switched to db Students
Atlas atlas-wanmtx-shard-0 [primary] Students> show collections

Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.insertMany([
...     { "Rollno": 10, "Name": "John", "Age": 20, "ContactNo": "1234567890", "Email-Id":
"john@example.com", "grade": "A", "hobby": "Reading" },
...     { "Rollno": 11, "Name": "Alice", "Age": 21, "ContactNo": "9876543210", "Email-Id":
 "alice@example.com", "grade":
 "B", "hobby": "Painting" },
...     { "Rollno": 12, "Name": "Bob", "Age": 22, "ContactNo": "2345678901", "Email-Id": "
bob@example.com", "grade": "C", "hobby": "Cooking" },
...     { "Rollno": 13, "Name": "Eve", "Age": 23, "ContactNo": "3456789012", "Email-Id": "
eve@example.com", "grade": "A"
 },
...     { "Rollno": 14, "Name": "Charlie", "Age": 24, "ContactNo": "4567890123", "Email-Id
": "charlie@example.com", "hobby": "Gardening" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("661ce9dc76a00ff8cc51dae1"),
    '1': ObjectId("661ce9dc76a00ff8cc51dae2"),
    '2': ObjectId("661ce9dc76a00ff8cc51dae3"),
    '3': ObjectId("661ce9dc76a00ff8cc51dae4"),
    '4': ObjectId("661ce9dc76a00ff8cc51dae5")
  }
}
```

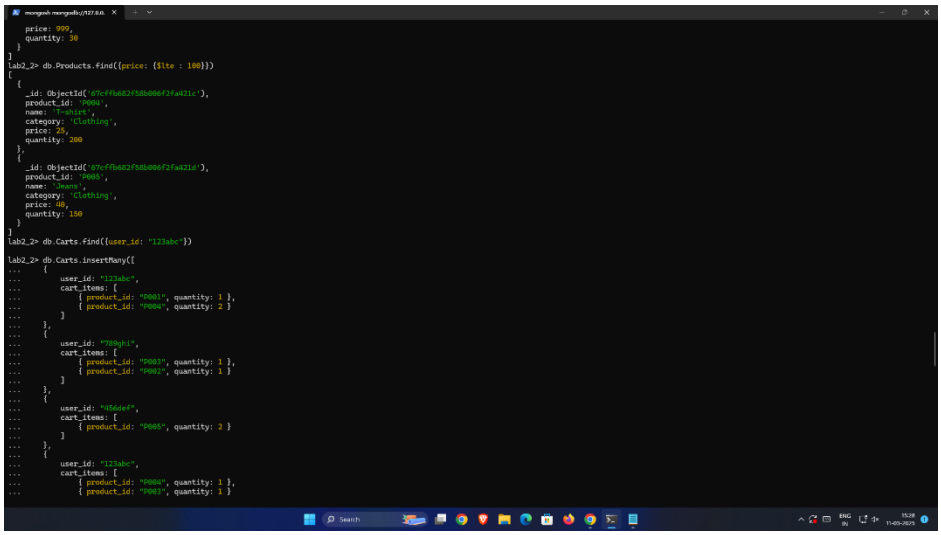
```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.find({ "hobby": { $nin: ["Chess
", "Skating"] } })
[
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae1"),
    Rollno: 10,
    Name: 'John',
    Age: 20,
    ContactNo: '1234567890',
    'Email-Id': 'john.doe@example.com',
    grade: 'A',
    hobby: 'Reading'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae2"),
    Rollno: 11,
    Name: 'Alicee',
    Age: 21,
    ContactNo: '9876543210',
    'Email-Id': 'alice@example.com',
    grade: 'B',
    hobby: 'Painting'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae3"),
    Rollno: 12,
    Name: 'Bob',
    Age: 22,
    ContactNo: '2345678901',
    'Email-Id': 'bob@example.com',
    grade: 'C',
    hobby: 'Cooking'
  },
```



```
...         ]
...     }
... ]);
...
[
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67cffb712f58b006f2fa4223'),
    '1': ObjectId('67cffb712f58b006f2fa4224'),
    '2': ObjectId('67cffb712f58b006f2fa4225'),
    '3': ObjectId('67cffb712f58b006f2fa4226'),
    '4': ObjectId('67cffb712f58b006f2fa4227')
  }
}
lab2_2> db.Products.find()
[
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
```



```
lab2_2> db.Carts.insertMany([
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P001", quantity: 1 },
...             { product_id: "P004", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P003", quantity: 1 },
...             { product_id: "P002", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "456def",
...         cart_items: [
...             { product_id: "P005", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P004", quantity: 1 },
...             { product_id: "P003", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P002", quantity: 1 },
...             { product_id: "P005", quantity: 3 }
...         ]
...     }
... ]);
...
...
{
  acknowledged: true,
  insertedIds: [
    '0': ObjectId('67cffcab2f58b006f2fa422B'),
    '1': ObjectId('67cffcab2f58b006f2fa4229'),
    '2': ObjectId('67cffcab2f58b006f2fa422a'),
    '3': ObjectId('67cffcab2f58b006f2fa422b'),
    '4': ObjectId('67cffcab2f58b006f2fa422c')
  ]
}
lab2_2> db.Carts.find({user_id: "123abc"})
[
```

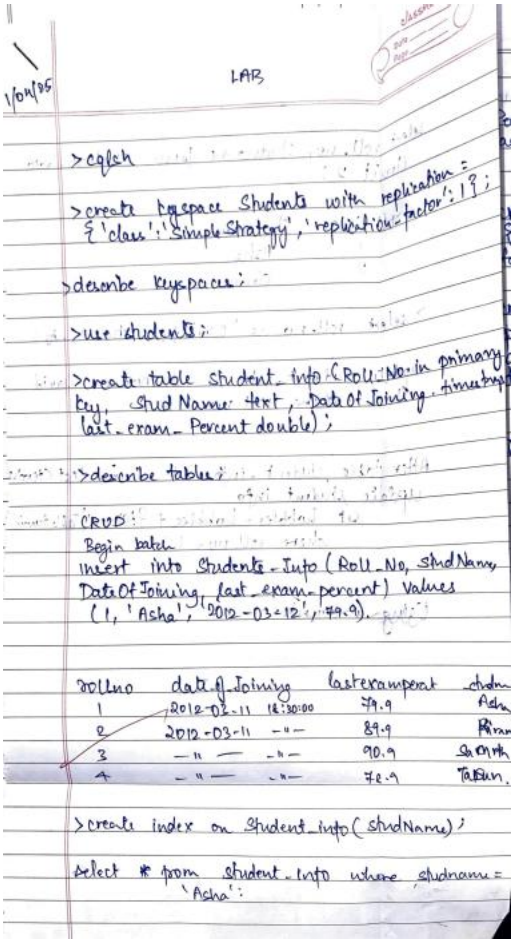


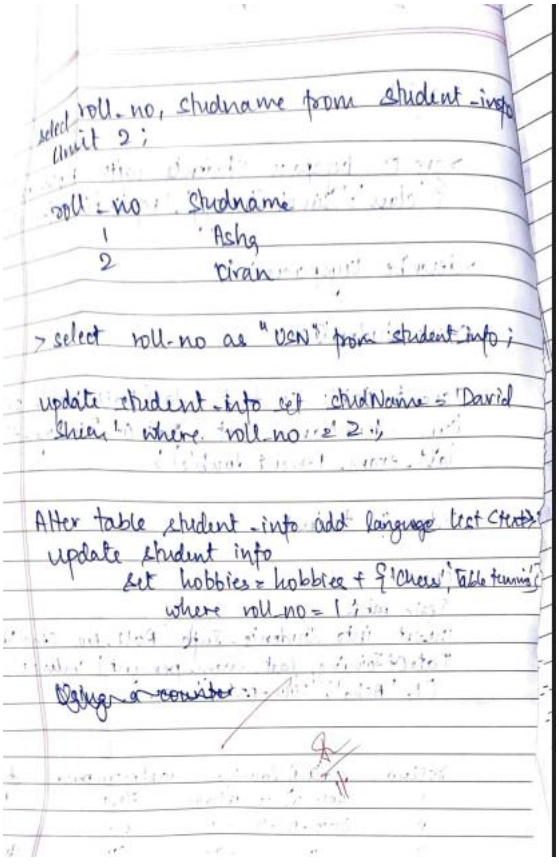
```
    price: 999,
    quantity: 30
  }
]
lab2_2> db.Products.find({price: {$lte : 100}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  }
]
lab2_2> db.Carts.find({user_id: "123abc"})

lab2_2> db.Carts.insertMany([
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P001", quantity: 1 },
...             { product_id: "P004", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P003", quantity: 1 },
...             { product_id: "P002", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "456def",
...         cart_items: [
...             { product_id: "P005", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P004", quantity: 1 },
...             { product_id: "P003", quantity: 1 }
```

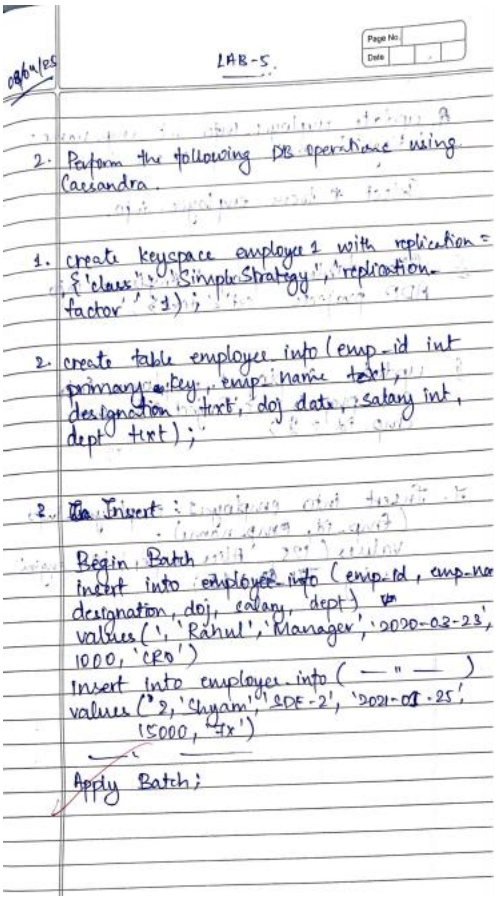
Experiment – 2

Perform the following DB operations using Cassandra.

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







LAB-5.

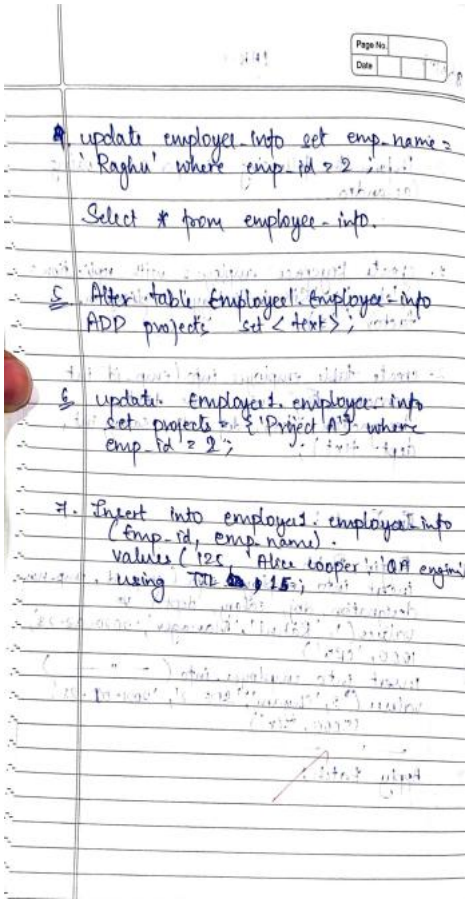
2. Perform the following DB operations using Cassandra.

1. create keyspace employee2 with replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

2. create table employee_info (emp_id int primary key, emp_name text, designation text, doj date, salary int, dept text);

3. Insert :

Begin Batch
insert into employee_info (emp_id, emp_name, designation, doj, salary, dept) values (1, 'Rahul', 'Manager', '2020-03-23', 1000, 'CEO')
insert into employee_info (— " —) values (2, 'Shyam', 'SDE-2', '2021-05-25', 15000, '4x')
Apply Batch;



4. update employee_info set emp_name = 'Raghu' where emp_id = 2;

Select * from employee_info.

5. Alter table employee1.employee_info ADD projects set<text>;

6. update employee1.employee_info set projects = {'Project A'} where emp_id = 2;

7. Insert into employee1.employee_info (emp_id, emp_name) values (125, 'Alice cooper') using TTL 15;

Codes Output:

```
bnscecse@bnscecse-HP-Elite-Tower-800-G9-Desktop-PC: $ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> create keyspace Employee with replication = {'class':'SimpleStrategy',:replicationfactor':1};
SyntaxException: line 1:80 mismatched input ':' expecting '}' (...with replication = {'class':'SimpleStrategy',:replicationfactor'[:]1...)
cqlsh> create keyspace Employee WITH replication={'class':'SimpleStrategy','replicationfactor':1};
ConfigurationException: Unrecognized strategy option {replicationfactor} passed to SimpleStrategy for keyspace employee
cqlsh> create keyspace Employee WITH replication={'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES

employee  system_auth         system_schema  system_views
system    system_distributed  system_traces  system_virtual_schema

cqlsh> CREATE TABLE IF NOT EXISTS Employee_Info(
   ... Emp_Id INT PRIMARY KEY,
   ... Emp_name TEXT,
   ... designation TEXT,
   ... date_of_joining DATE,
   ... Salary FLOAT,
   ... Dep_name TEXT,
   ... Projects SET<TEXT>);
InvalidRequest: Error from server: code=2200 [Invalid query] message="No keyspace has been specified. USE a keyspace, or explicitly specify keyspace.tablename"
cqlsh> USE eMPLOYEE
   ...
cqlsh> USE Employee
   ...
cqlsh> USE Employee;
cqlsh:employee> CREATE TABLE IF NOT EXISTS Employee_Info( Emp_Id INT PRIMARY KEY, Emp_name TEXT, designation TEXT, date_of_joining DATE, Salary FLOAT, Dep_name TEXT, Projects SET<TEXT>);
cqlsh:employee> describe keyspace employee

CREATE KEYSPACE employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;

CREATE TABLE employee.employee_info (
    emp_id int PRIMARY KEY,
    date_of_joining date,
    dep_name text,
    designation text,
    emp_name text,
    salary float,
    projects set<text>
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
```

```
cqlsh:employee> update employee_info using ttl 15 set salary = 0 where emp_id = 121;
cqlsh:employee> select * from employee_info;

 emp_id | bonus | date_of_joining | dep_name    | designation | emp_name    | projects                    | salary
--------+-------+-----------------+-------------+-------------+-------------+-----------------------------+---------
    120 | 12000 |      2024-05-06 | Engineering |   Developer | Priyanka GH |  {'Project B', 'ProjectA'} |   1e+06
    123 |  null |      2024-05-07 | Engineering |    Engineer |     Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |  null |      2024-05-06 |  Management |          HR |     Rachana | {'Project C', 'Project M'} |   9e+05
    121 | 11000 |      2024-05-06 |  Management |   Developer |      Shreya |  {'Project C', 'ProjectA'} |       0

(4 rows)
cqlsh:employee> select * from employee_info;

 emp_id | bonus | date_of_joining | dep_name    | designation | emp_name    | projects                    | salary
--------+-------+-----------------+-------------+-------------+-------------+-----------------------------+---------
    120 | 12000 |      2024-05-06 | Engineering |   Developer | Priyanka GH |  {'Project B', 'ProjectA'} |   1e+06
    123 |  null |      2024-05-07 | Engineering |    Engineer |     Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |  null |      2024-05-06 |  Management |          HR |     Rachana | {'Project C', 'Project M'} |   9e+05
    121 | 11000 |      2024-05-06 |  Management |   Developer |      Shreya |  {'Project C', 'ProjectA'} |    null

(4 rows)
cqlsh:employee>
```

```
   AND speculative_retry = '99p';
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining | dep_name    | designation | emp_name | projects                    | salary
--------+-----------------+-------------+-------------+----------+-----------------------------+---------
    120 |      2024-05-06 | Engineering |   Developer | Priyanka |  {'Project B', 'ProjectA'} |   1e+06
    123 |      2024-05-07 | Engineering |    Engineer |  Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |      2024-05-06 |  Management |          HR |  Rachana | {'Project C', 'Project M'} |   9e+05
    121 |      2024-05-06 |  Management |   Developer |   Shreya |  {'Project C', 'ProjectA'} |   9e+05

(4 rows)
cqlsh:employee> update employee_info set emp_name = 'Priyanka GH' Where emp_id = '120';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Invalid STRING constant (120) for "emp_id" of type int"
cqlsh:employee> update employee_info set emp_name = 'Priyanka GH' Where emp_id=120;
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining | dep_name    | designation | emp_name    | projects                    | salary
--------+-----------------+-------------+-------------+-------------+-----------------------------+---------
    120 |      2024-05-06 | Engineering |   Developer | Priyanka GH |  {'Project B', 'ProjectA'} |   1e+06
    123 |      2024-05-07 | Engineering |    Engineer |     Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |      2024-05-06 |  Management |          HR |     Rachana | {'Project C', 'Project M'} |   9e+05
    121 |      2024-05-06 |  Management |   Developer |      Shreya |  {'Project C', 'ProjectA'} |   9e+05

(4 rows)
cqlsh:employee> select * from employee_info order by salary;
InvalidRequest: Error from server: code=2200 [Invalid query] message="ORDER BY is only supported when the partition key is restricted by an EQ or an IN."
cqlsh:employee> alter table employee_info add bonus INT;
cqlsh:employee> select * from employee_info;

 emp_id | bonus | date_of_joining | dep_name    | designation | emp_name    | projects                    | salary
--------+-------+-----------------+-------------+-------------+-------------+-----------------------------+---------
    120 |  null |      2024-05-06 | Engineering |   Developer | Priyanka GH |  {'Project B', 'ProjectA'} |   1e+06
    123 |  null |      2024-05-07 | Engineering |    Engineer |     Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |  null |      2024-05-06 |  Management |          HR |     Rachana | {'Project C', 'Project M'} |   9e+05
    121 |  null |      2024-05-06 |  Management |   Developer |      Shreya |  {'Project C', 'ProjectA'} |   9e+05

(4 rows)
cqlsh:employee> update employee_info set bonus = 12000 where emp_id = 120;
cqlsh:employee> select * from employee_info;

 emp_id | bonus | date_of_joining | dep_name    | designation | emp_name    | projects                    | salary
--------+-------+-----------------+-------------+-------------+-------------+-----------------------------+---------
    120 | 12000 |      2024-05-06 | Engineering |   Developer | Priyanka GH |  {'Project B', 'ProjectA'} |   1e+06
    123 |  null |      2024-05-07 | Engineering |    Engineer |     Sadhana | {'Project M', 'Project P'} | 1.2e+06
    122 |  null |      2024-05-06 |  Management |          HR |     Rachana | {'Project C', 'Project M'} |   9e+05
    121 |  null |      2024-05-06 |  Management |   Developer |      Shreya |  {'Project C', 'ProjectA'} |   9e+05

(4 rows)
cqlsh:employee> update employee_info set bonus = 11000 where emp_id = 121;
cqlsh:employee> select * from employee_info using ttl 15 where emp_id = 123;
SyntaxException: line 1:28 mismatched input 'using' expecting EOF (select * from employee_info [using] ttl...)
cqlsh:employee> select * from employee_info where emp_id = 121 using ttl 15;
SyntaxException: line 1:47 no viable alternative at input 'using' (...employee_info where emp_id = 121 [using]...)
cqlsh:employee> update employee_info using ttl 15 set salary = 0 where emp_id = 121;
cqlsh:employee> select * from employee_info;
```

Experiment – 3

Perform the following DB operations using Cassandra:

● Create a keyspace by name Library

● 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

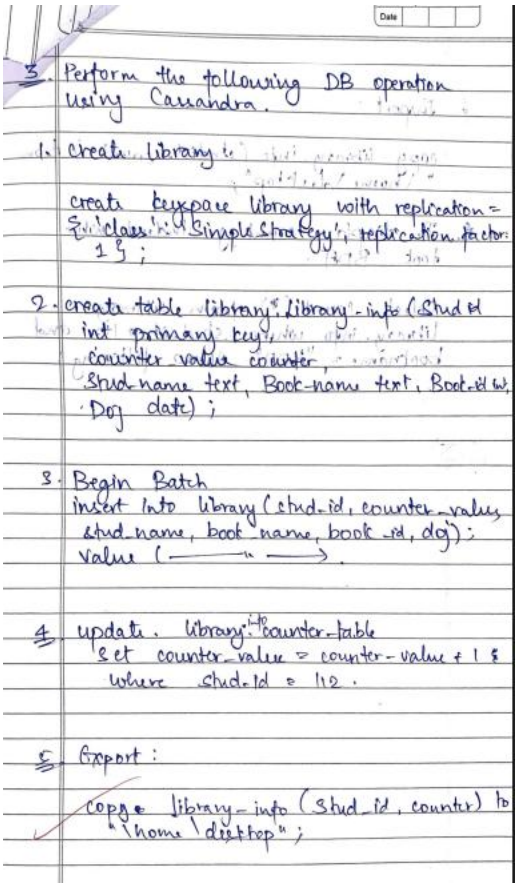
● Insert the values into the table in batch

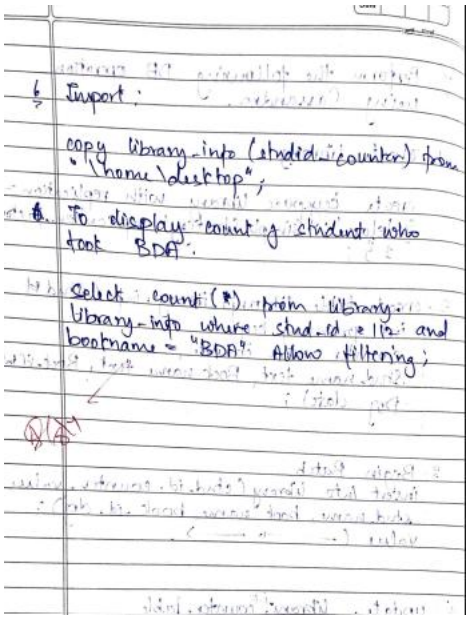
● Display the details of the table created and increase the value of the counter

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

● Export the created column to a csv file

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





Codes Output:

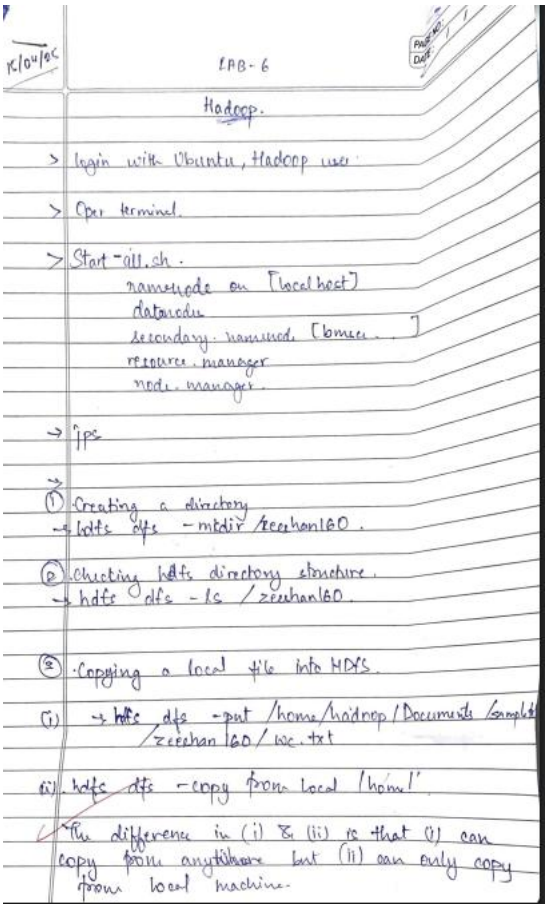
```
bmscecse@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Students WITH REPLICATION={
   ... 'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES

students  system_auth         system_schema  system_views
system    system_distributed  system_traces  system_virtual_schema

cqlsh> SELECT * FROM system.schema_keyspaces;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table schema_keyspaces does not exist"
cqlsh> use Students;
cqlsh:students> create table Students_info(Roll_No int Primary key,StudName text,DateOfJoining timestamp,last_exam_Percent double);
cqlsh:students> describe tables;

students_info

cqlsh:students> describe table students;
Table 'students' not found in keyspace 'students'
cqlsh:students> describe table students_info;

CREATE TABLE students.students_info (
    roll_no int PRIMARY KEY,
    dateofjoining timestamp,
    last_exam_percent double,
    studname text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    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 = 'BLOCKING'
    AND speculative_retry = '99p';
```

```
cqlsh:students> Begin batch insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(1,'Sadhana','2023-10-09', 98) insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam
_Percent) values(2,'Rutu','2023-10-10', 97) insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(3,'Rachana','2023-10-10', 97.5) insert into Students_info(Roll_no, StudName,DateOf
Joining, last_exam_Percent) values(4,'Charu','2023-10-06', 96.5) apply batch;
cqlsh:students> select * from students_info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |              98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |              97 |     Rutu
       4 | 2023-10-05 18:30:00.000000+0000 |            96.5 |    Charu
       3 | 2023-10-09 18:30:00.000000+0000 |            97.5 |  Rachana

(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |              98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |              97 |     Rutu
       3 | 2023-10-09 18:30:00.000000+0000 |            97.5 |  Rachana

(3 rows)
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query d
espite the performance unpredictability, use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       4 | 2023-10-05 18:30:00.000000+0000 |            96.5 |    Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;
```

```
(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |              98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |              97 |     Rutu
       3 | 2023-10-09 18:30:00.000000+0000 |            97.5 |  Rachana

(3 rows)
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query d
espite the performance unpredictability, use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       4 | 2023-10-05 18:30:00.000000+0000 |            96.5 |    Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;

 roll_no | studname
---------+----------
       1 |  Sadhana
       2 |     Rutu

(2 rows)
cqlsh:students> SELECT Roll_no as "USN" from Students_info;

 USN
-----
   1
   2
   4
   3
```

Experiment-4

Execution of HDFS Commands for interaction with Hadoop Environment.


LAB-6

Hadoop.

> login with Ubuntu, Hadoop user

> Open terminal.

> Start-all.sh.
namenode on [localhost]
datanode
secondary namenod. [bmsce...]
resource manager
node manager.

→ jps

→

(1) Creating a directory
→ hdfs dfs -mkdir /zeeshan160.

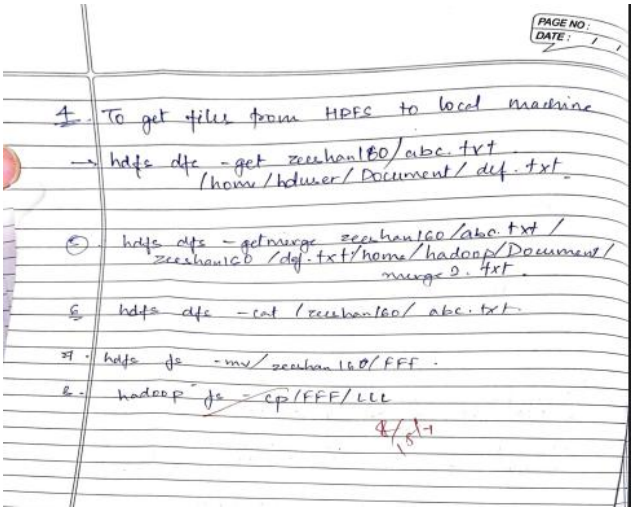
(2) Checking hdfs directory structure.
→ hdfs dfs -ls /zeeshan160.

(3) Copying a local file into HDFS.

(i) → hdfs dfs -put /home/hadoop/Documents/sample /zeeshan160/wc.txt

(ii) hdfs dfs -copyFromLocal /home/

The difference in (i) & (ii) is that (i) can copy from anywhere but (ii) can only copy from local machine.


PAGE NO:
DATE: / /

4. To get files from HDFS to local machine
→ hdfs dfs -get zeeshan160/abc.txt /home/hduser/Document/def.txt

5. hdfs dfs -getmerge zeeshan160/abc.txt / zeeshan160/def.txt/home/hadoop/Document/ merge9.txt.

6. hdfs dfs -cat /zeeshan160/abc.txt

7. hdfs fs -mv/zeeshan160/FFF.

8. hadoop fs -cp/FFF/LLL

Codes Output:

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cd ./Desktop/
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -mkdir /Lab05
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ touch test.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ nano text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -put ./text.txt /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Lab05
Found 1 items
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cat /Lab05/text.txt
Hello
How are you?
```

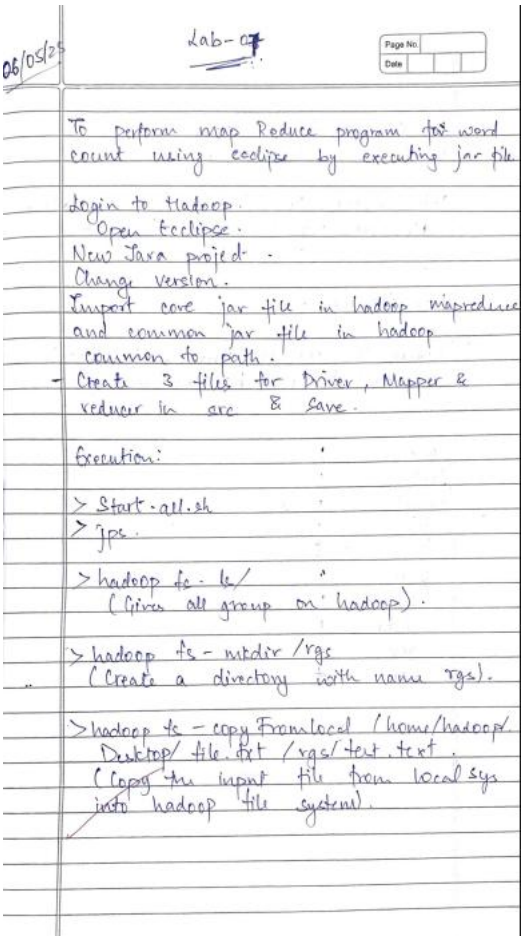
```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -getmerge /Lab05 /text.txt /Lab05 /test.txt ../
Downloads/Merged.txt
getmerge: `/text.txt': No such file or directory
getmerge: `/test.txt': No such file or directory
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -getmerge /Lab05/text.txt /Lab05/test.txt ../Do
wnloads/Merged.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -getfacl /Lab05
# file: /Lab05
# owner: hadoop
# group: supergroup
user::rwx
group::r-x
other::r-x
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cat /Lab05/text.txt
Hello
How are you?
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -mv /Lab05 /test_Lab05
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /test_Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /test_Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /test_Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cp /test_Lab05/ /Lab05
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:51 /Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:51 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /test_Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /test_Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /test_Lab05/text.txt
```

Experiment-5

Implement Wordcount program on Hadoop framework



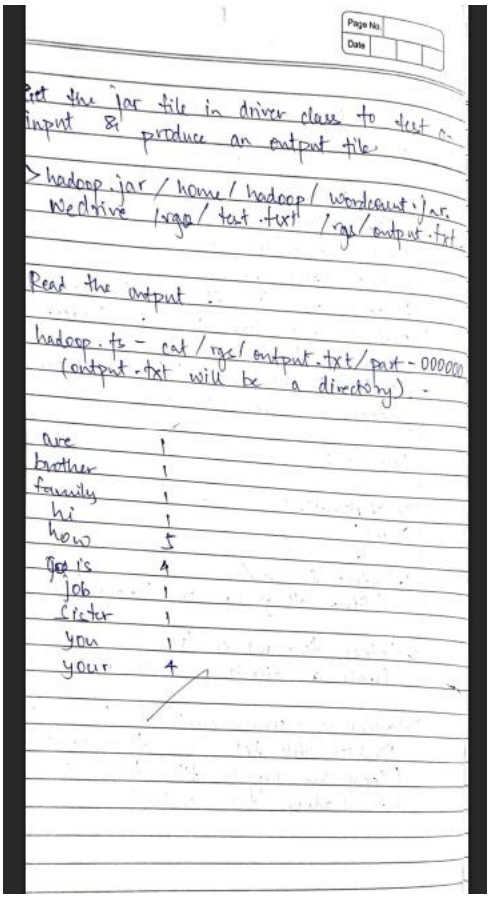
Lab-07

To perform map Reduce program for word count using eclipse by executing jar file

login to Hadoop
Open Eclipse.
New Java project.
Change version.
Import core jar file in hadoop mapreduce and common jar file in hadoop common to path.
Create 3 files for Driver, Mapper & reducer in src & save.

Execution:

> Start-all.sh
> jps
> hadoop fs -lsr /
(Gives all group on hadoop).
> hadoop fs -mkdir /rgs
(Create a directory with name rgs).
> hadoop fs -copyFromLocal /home/hadoop/Desktop/file.txt /rgs/test.text.
(Copy the input file from local sys into hadoop file system).



set the jar file in driver class to test c- input & produce an output file.

> hadoop.jar /home/hadoop/wordcount.jar. Wedrive /rgs/test.txt /rgs/output.txt

Read the output.

hadoop.fs - cat /rgs/output.txt/part-000000
(output.txt will be a directory).

are	1
brother	1
family	1
hi	1
how	1
is	1
job	1
sister	1
you	1
your	1

Mapper:

```
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text,
Text,
IntWritable> {
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter rep)
throws IOException
```

```
{
String line = value.toString();
for (String word : line.split(" "))
{
if (word.length() > 0)
{
output.collect(new Text(word), new IntWritable(1)); } } } }
```

Reducer:

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WCReducer extends MapReduceBase implements Reducer<Text,IntWritable, Text,
IntWritable> {
// Reduce function
public void reduce(Text key, Iterator<IntWritable> value,
OutputCollector<Text, IntWritable> output,
Reporter rep) throws IOException
{
int count = 0;
// Counting the frequency of each words
while (value.hasNext())
{
IntWritable i = value.next();
count += i.get();
}
output.collect(key, new IntWritable(count));
}}
```

Driver:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WCDriver extends Configured implements Tool {
```

```
public int run(String args[]) throws IOException
{
if (args.length < 2)
{
System.out.println("Please give valid inputs");
return -1;
}
JobConf conf = new JobConf(WCDriver.class);
FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));
conf.setMapperClass(WCMapper.class);
conf.setReducerClass(WCReducer.class);
conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(IntWritable.class);
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
JobClient.runJob(conf);
return 0;
}
public static void main(String args[]) throws Exception
{
int exitCode = ToolRunner.run(new WCDriver(), args);
System.out.println(exitCode);
}
}
```

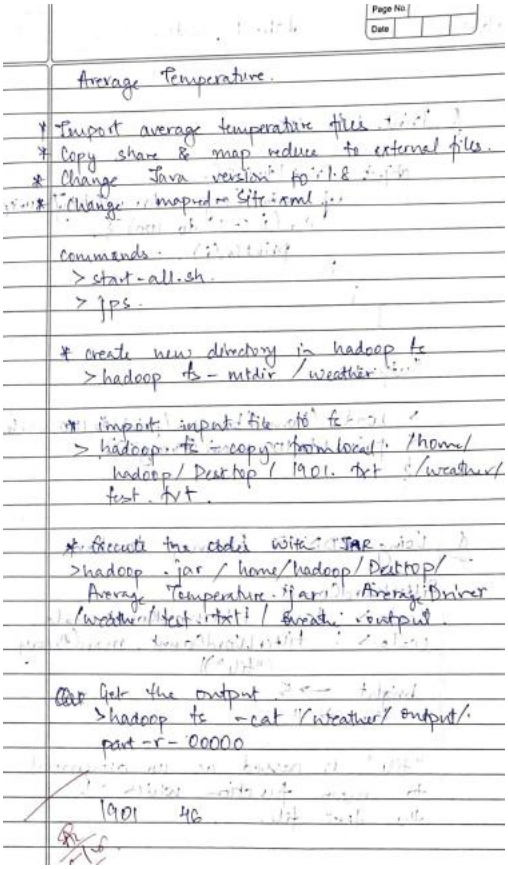
Codes Output:

Experiment-6

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

Create a Map Reduce program to:

- Find average temperature for each year from NCDC data set.
- Find the mean max temperature for every month.



Mapper:

```
#!/usr/bin/env python3
import sys

for line in sys.stdin:
    line = line.strip()
```

```
    parts = line.split()
    date, temp = parts
    temp = float(temp)
    print(f"{date}\t{temp}")
```

Reducer1:

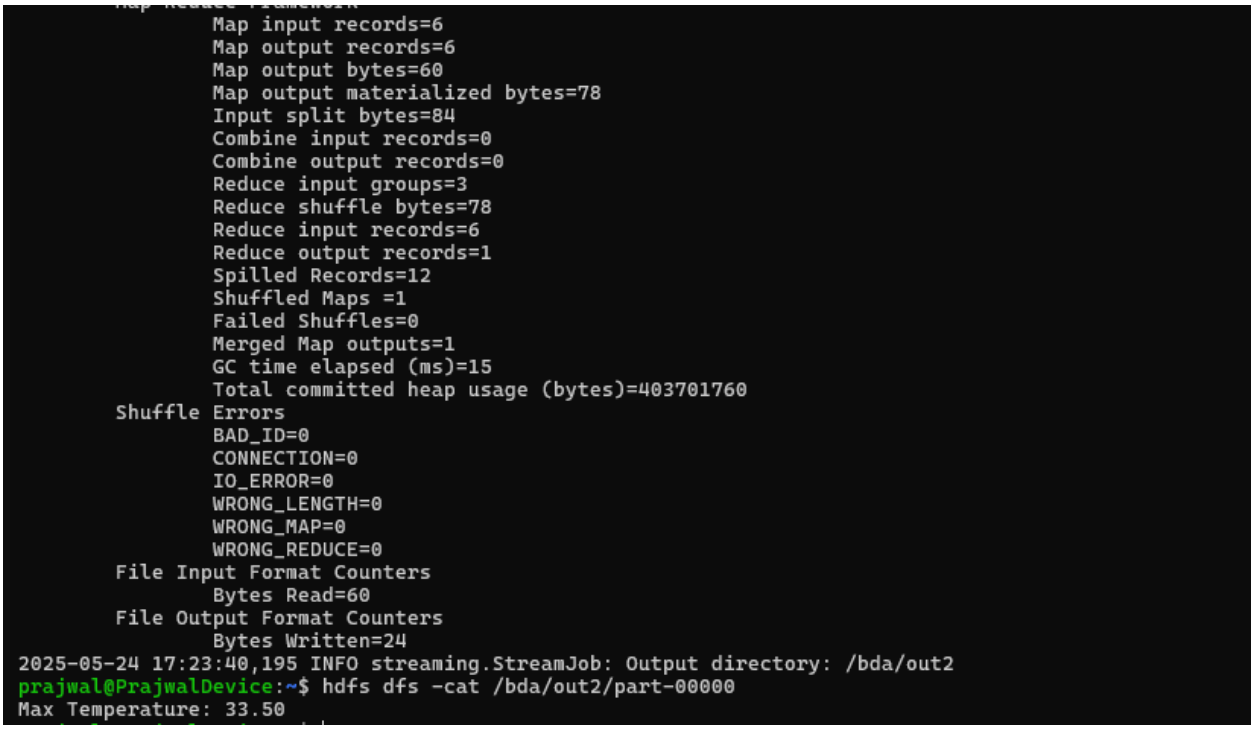
```
#!/usr/bin/env python3
import sys
count = 0
total_temp = 0.0
for line in sys.stdin:
    line = line.strip()
    key, value = line.split("\t")
    try:
        total_temp += float(value)
        count += 1
    except ValueError:
        continue

if count > 0:
    mean_temp = total_temp / count
    print(f"Mean Temperature: {mean_temp:.2f}")
else:
    print("No valid temperature records.")
```

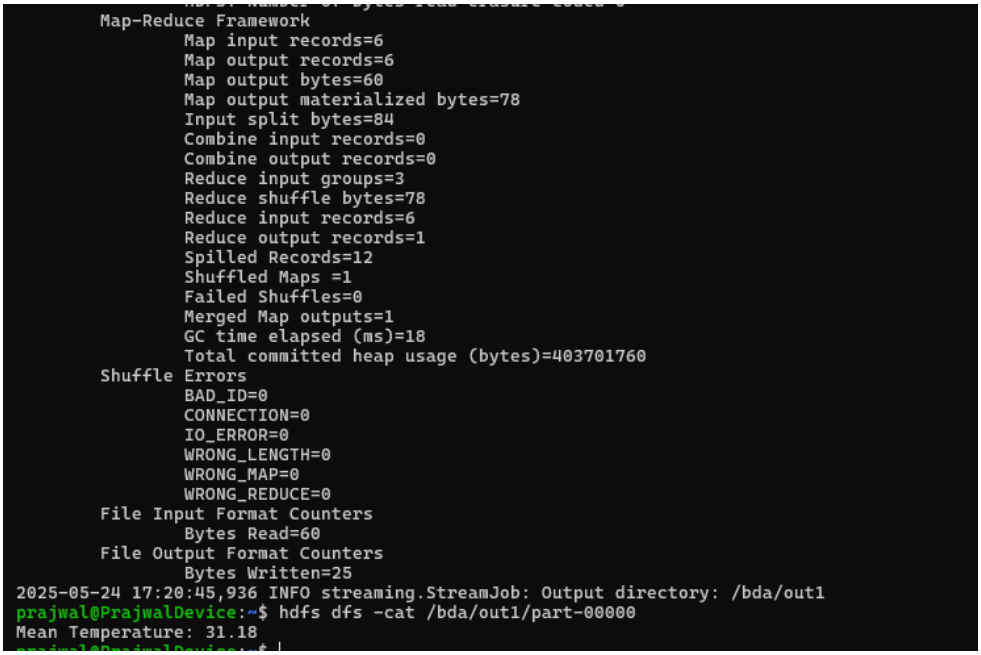
Reducer2:

```
#!/usr/bin/env python3
import sys

max_temp = float('-inf')

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    try:
        key, value = line.split("\t")
        temp = float(value)
        if temp > max_temp:
            max_temp = temp
    except ValueError:
```

```
        continue

if max_temp != float('-inf'):
    print(f"Max Temperature: {max_temp:.2f}")
else:
    print("No valid temperature records.")
```

Codes Output:





Experiment-7

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

Mapper:

```
#!/usr/bin/env python3

import sys
import re

for line in sys.stdin:
    words = re.findall(r'\w+', line.lower())  # normalize case
    for word in words:
        print(f"{word}\t1")
```

Reducer:

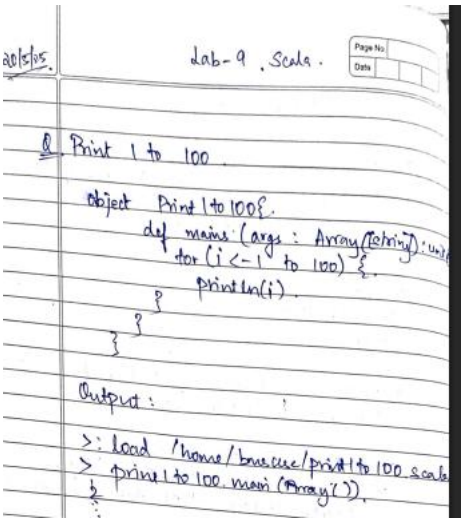
```
#!/usr/bin/env python3
import sys
from collections import defaultdict

N = 10  # change this to desired Top-N

word_counts = defaultdict(int)

# Aggregate word counts
for line in sys.stdin:
    word, count = line.strip().split("\t")
    word_counts[word] += int(count)

# Sort by frequency desc, then word asc
top_n = sorted(word_counts.items(), key=lambda x: (-x[1], x[0]))[:N]

# Output Top-N
for word, count in top_n:
    print(f"{word}\t{count}")
```

Codes Output:

```
                Reduce input groups=18
                Reduce shuffle bytes=239
                Reduce input records=25
                Reduce output records=10
                Spilled Records=50
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=15
                Total committed heap usage (bytes)=421527552
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=137
        File Output Format Counters
                Bytes Written=72
2025-05-24 17:25:13,559 INFO streaming.StreamJob: Output directory: /bda/out3
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out3/part-00000
the     3
foxes   2
hares   2
jumps   2
quick   2
than    2
are     1
blue    1
brown   1
dog     1
```

Experiment-8

Write a Scala program to print numbers from 1 to 100 using for loop.



Scala Code:

```
Scala> for(i <- 0  to 100){
        println(i)
        }
```

0

1

2

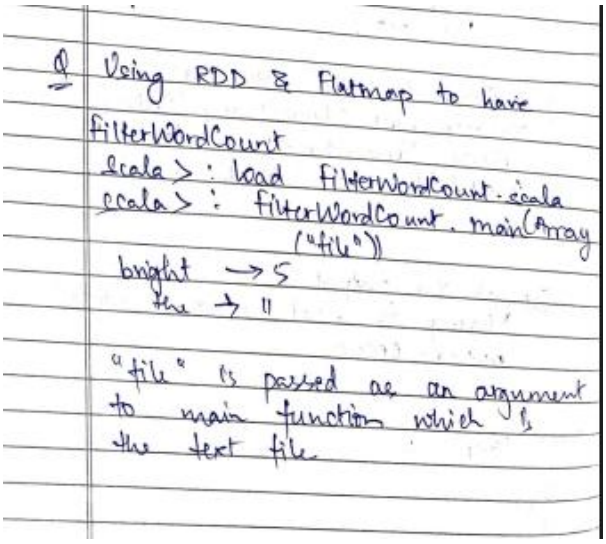
.

.

Codes Output:

Experiment-9

Using RDD and FlatMap 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.



Codes Output:

```
prajwal@PrajwalDevice:~$ spark-shell
25/05/24 17:41:38 WARN Utils: Your hostname, PrajwalDevice resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on interface lo)
25/05/24 17:41:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/05/24 17:41:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://10.255.255.254:4040
Spark context available as 'sc' (master = local[*], app id = local-1748088707553).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.5
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 21.0.7)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val file=sc.text25/05/24 17:42:00 WARN GarbageCollectionMetrics: To enable non-built-in garbage collector(s) List(G1 Concurrent GC), users should configure it(them) to spark.e
 spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
val file=sc.textFile("i1.txt")
file: org.apache.spark.rdd.RDD[String] = i1.txt MapPartitionsRDD[1] at textFile at <console>:23

scala> val words=file.flatMap(line=>line.split("\\W+"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:23

scala> val wordpairs=words.map(word=>(word.toLowerCase,1))
wordpairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:23

scala> val wordc=wordpairs.reduceByKey(_+_)
wordc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

scala> val fil2=wordc.filter{case(word,count)=>count>2}
fil2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at filter at <console>:23

scala> fil2.collect().foreach(println)
```