INDEX

Name : Arbaj Wadagera Class : V th sem

Section : A Roll No. : 1BM22CS051 Subject : BDA Lab

Sl. No.	Date	Title	Page No.	Teacher's Sign. / Remarks
1	3/3/25	LAB-I		
2	10/3/25	LAB - II		
3	17/3/25	LAB - III		
4	24/3/25	LAB - IV		
5	7/4/25	LAB - V		
6	21/4/25	LAB - VI		
7	05/8/25	LAB - VII		
8	19/5/25	LAB - VIII		

Lab - I

→ mongosh

→ use myDB

→ db.createCollection("Student");

→ db.Student.insert({RollNo: 1, Age: 21, Cout: 9876, email: "abcd@gmail.com"});

→ db.student.find()

```
[
  {
    _id : objectId("63bFef...."),
    RollNo : 1,
    Age : 21,
    cout : 9876,
    email : "abcd@gmail.com"
  }
]
```

→ db.student.update({RollNo: 10}, {$set: {email: "abc@gmail.com"}})

```
{
  acknowleged : true,
  insertedId : null,
  matched count : 1,
  modified count : 1,
  upserted count : 0
}
```

→ db.Student.insert({ RollNo : 11, Age : 22,
Name : "ABC", Cout : 2276,
email : "xyz@gmail.com"})

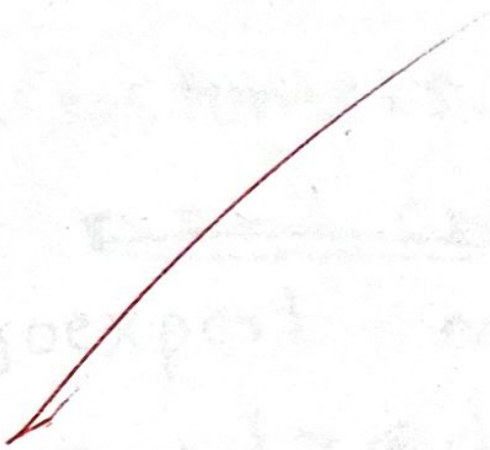
```
{
  -id : ObjectId ("63bfe1...."),
  RollNo : 11,
  Age : 22,
  Name : "ABC"
  Cout : 2276
  email : "xyz@gmail.com"
}
```

→ db.student.update({RollNo : 11, Name : "ABC"},
{$set : {Name : "FEM"}})

```
{
  -id:objectId ("63bfd4...."),
  RollNo : 11,
  Age : 22,
  Name : "FEM",
  Cout : 2276
  email : "xyz@gmail.com"
}
```

→ mongoexport <copied connection url/Database Name> -- collection = Tablename
-- out <output file name on local system>
Connected to : <connection url/Database Name>
Exported 3 records

→ mongoimport <copied connection url/ Database Name>
--collection = Table name --type json --file
<file name>
connected to : <connection Url/ Database name>
3 documents imported successfully. 0 documents failed to import.



LAB - II

→ mongosh

→ use bankDB;

→ db.Customers.insertOne({

cust_id:1,
Acc_Bal: 1500,
Acc_Type: 'z'

});

→ db.Customers.insertMany([

{ cust_id:1, Acc_Bal: 1500, Acc_Type: 'z' },
{ Cust_id:2, Acc_Bal: 1100, Acc_Type: 'z' },
{ cust_id:3, Acc_Bal: 1300, Acc_Type: 'x' },
{ Cust_id: 4, Acc_Bal: 1600, Acc_Type: 'z' },
{ cust_id: 5, Acc_Bal: 900, Acc_Type: 'y' }

]);

→ db.Customers.find({

Acc_Bal: { $gt: 1200 },
Acc_Type: 'z'

});

→ db.Customers.aggregate([

{

$group: {

```
    _id : "$cust_id",
    minBalance : { $min : "$Acc_Bal" },
    maxBalance : { $max : "$Acc_Bal" }
    }

  }
]);
```

→ db.Customers.find()

new cmd

→ mongoexport mongodb+srv://arbajcs22:<password>@cluster0.fsgaw.mongodb.net/bankDB --collection=Customers --out C:\Users\student\Downloads\output2.csv

exported 6 records.

back to earlier (other) cmd.

→ db.Customers.drop();

for import

mongoimport _____

_____ = Customersabc --file

10:13:25

6 document(s) imported successfully.

Lab-3

→ neo4j $ CREATE (a: STUDENT {name: "ram", id: 123})

→ MATCH (n) RETURN n

→ CREATE (a: STUDENT {name: "raj", id: 13})

→ CREATE (a: STUDENT {name: "abhi", id: 14})

→ CREATE (a: STUDENT {name: "anjali", id: 15})

→ MATCH (n) RETURN n

→ CREATE (a: COURSE {name: "DBMS", credits: 3})
RETURN a

→ CREATE (a: COURSE {name: "BDA", credits: 4})
RETURN a

→ CREATE (a: TEACHERS {name: 'VBM', 3})

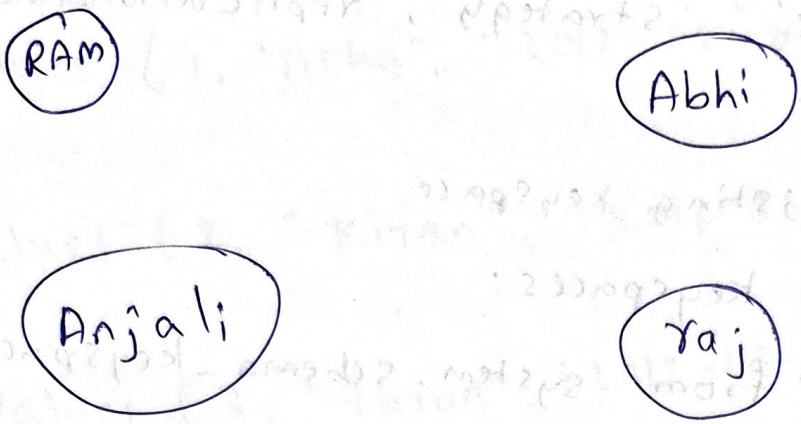
→ MATCH (a: STUDENT), (b: COURSE) where a.name = "Anjali" AND b.name = "BDA"
CREATE (a) -[r: ENROLLED-IN] → (b)

→ MATCH (n) RETURN n

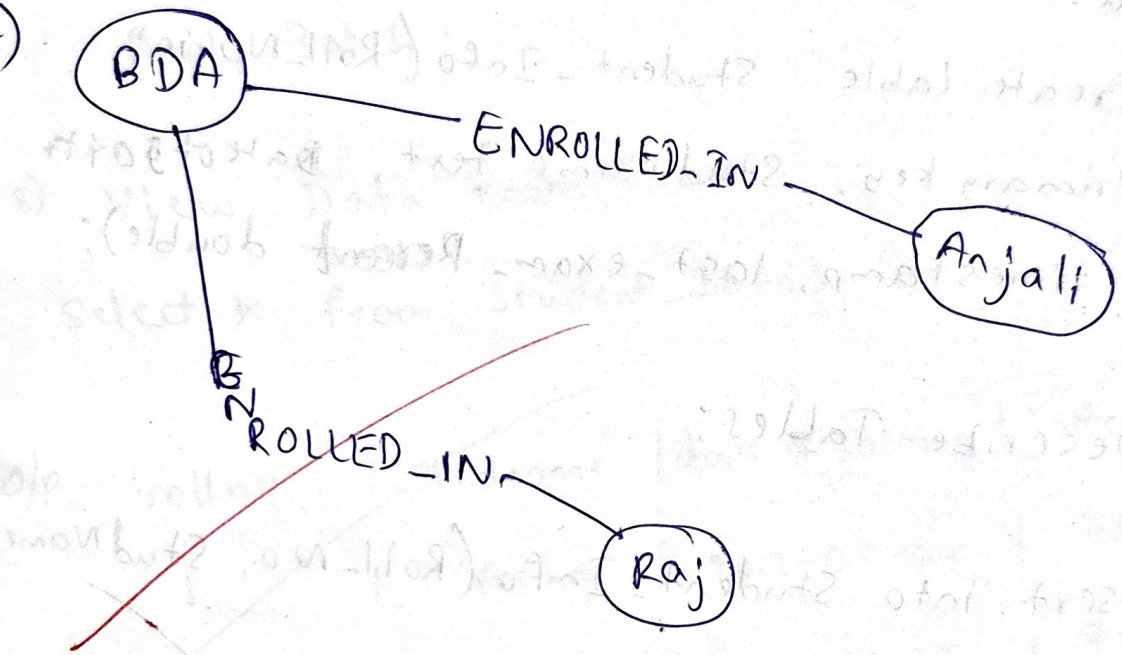
→ MATCH (a: TEACHER), (b: COURSE) where a.name = "VBM" AND b.name = "BDA"
CREATE (a) - [r: ENROLLED_IN] → (b)

Q) P –

1)



2)



17/3/25

Lab-4

1) Create keyspace
Create keyspace Students with REPLICATION =
{'class': 'strategy', replication_factor': 1};

2) Check existing keyspace

3) Describe keyspaces:

4) Select * from system.schema_keyspaces;

use Students;

5) Create table Student_Info (Roll_No int
Primary key, Stud Name text, Date of Birth
timestamp, last_exam_percent double);

6) Describe Tables;

7) Insert into Student_Info (Roll_No, Stud Name,
Date of Birth, last_exam_percent)
Value (1, 'Asha', '2012-03-12');

(Select * from select_Info where
Roll_No. IN (1, 2, 3);)

insert multiple records using Batch Begin
Batch

```
Insert Into Students_info ( roll_no, stud name,
                date of joining, last exam present)

Values (1, "Asha", "2012-05-04, 78);

Values (2, "Kiran", "2012-06-13', 85);

Values (3, 'Tarun', '2012-03-12', 74);

Apply Batch;
```

8) View Data from Table

```
·select * from Student_Info;
```

O/P

rollno	studname	date of Joining	lastexampercent
1	Asha	2012-05-04	78.0
2	Kiran	2012-06-13	85.0
3	Tarun	2012-08-13	74.0

9) Filter using where clause.

```
· select * from Student_info where roll_no
  in (1, 2, 3);
```

o/p

roll no	student name	date of joining	last exam percent
1	Asha	2012-05-04	78.0
2	Kiran	2012-06-13	85.0
3	Tarun	2012-08-12	74.0

10. Errors when querying non-primary key column

```
Select * from students_info where
studentname = 'Asha';
```

11) Create an index on student name

```
Create Index on student_info (student name);
```

12) Query using the index

```
select * from student_info where
Studentname = 'Asha';
```

o/p

roll_no	studname	date of joining	last exam percent
1	Asha	2012-05-04	78.0

13) Update a record

```
update student_info set Studentname = 'David'
where roll_no = 2;
```

14) Delete a record

```
Delete from student_info where roll_no = 3;
```

15) Alter table to add a column

```
Alter table student_info add hobbies text;
```

16) Export Data to csv

```
copy students_info TO 'student_data.csv';
```

17) import data from csv.

```
copy students_info from 'student_data.csv';
```

7/4/2025

Lab-05

① Perform the foll. DB operations using Cassandra

(1)→ Create keyspace by name Employee

Ans:→ Create keyspace employee with
replication = {'class': 'SimpleStrategy',
replication_factor : 1}

② Create a Column family by name Employee_Info. with attributes Emp_id, primarykey, Emp_name, Designation, Date_of_Joining, Salary, Dept_name.

Ans:→ Create Table employee_Info (
Emp_Id int,
Emp_name text,
Designation text,
Date_of_joining date,
Salary double,
Dept_name text,
PRIMARY KEY (Dept_name, Salary, Emp_Id)
with Clustering Order By
(salary DESC);

(3) Insert values

```
Begin Batch
  Insert INTO Employee_Info (Dept_name,
    Salary, Emp_ID, Emp_name, Designation,
    Date_of_joining)
Values ( R&D ', 75000, 121, Alicia',
    'Software Eng.', '2021-01-01');

Select * from Employee_Info;
```

output

Output

dept_name	salary	Emp_Id	designation	Emp_name
HR	90,000	123	Manager	carol
IT	70000	121	Developer	Alia
R&D	75000	121	soft. Eng.	Alicia
Finance	60000	122	Analyst	Bob
Analytics	65000	122	Data Analyst	Bob

(4) Sort the details of Employee records based on Salary

(5) Alter schema of table Employee_Info to add a column Project which stores a set of Project done by corr. Employee.

Ans → ALTER table Employer_Info ADD Projects set <text>;

6. Update the altered table to add project names

⇒ UPDATE Employee_Info
SET Projects = {'Project Alpha', 'Project Beta'}
where Dept_Name = 'RAD' AND
Salary = 75000 And Emp_Id = 121,

7. Create a TTL of 15 seconds to display the values of Employees

⇒ Update Employee_Info using TTL 15 SET
Emp_Name = 'Temporary name'
where Dept_name = 'HR' AND
Salary = 90000 AND Emp_ID = 123;

(2) Perform the foll. DB operation using CASANDRA

1. Create a keyspace by name Library

⇒ Create keyspace Library with
replication = {'class': 'simple strategy',
'replication_factor': 3};

2. Create a column family by name Library-info with attributes Stud_Id, Primary key, counter_value, of type counter, Stud_Name, book_Name, Book_Id, Date_of_Issue,

3. Insert values

BEGIN Batch

Insert into Employee_INFO (Dept_name, Salary_ Emp-ID).

Create table Library_Info

Stud_Id int,
Stud_name text,
Book_Name text,
Book_Id text,
Date_of_issue date
PRIMARY KEY (Stud_Id, Book_Name));

Create table Book_Counter (
Stud_Id int,
Book_name text,
Counter_value counter,
PRIMARY KEY (Stud_Id, Book_Name));

4) Display Details & Increase the counter,

⟹ Select * from Library_Info;

Select * from Book_Counter where

Update Book_Counter SET Counter_value
counter_value + where Stud_Id = 112 &
Book_Name = 'BDA';

5) Show student with ID 112 has taken Book "BDA" 2 times

Ans → select counter_value from Book_counter where stud_Id = 112 & Book_Name = BDA;

Output:

counter_value
2

6) Cqlsh -e "Copy Library.Library_Info INTO 'Library_info.csv' with HEADER = "TRUE";

7) Cqlsh -e "Copy Library.Library_Info (Book_Name, Id, Issue) FROM 'Library info.csv with HEADER = TRUE';

Lab-06

Word Count Program Map reduce

1) Mapper code. java

```
Public Class mapperCode extends mapper
  <long variable, text, Text, Inwriteable>
public void map (longwritable key,
  Text value, contact context throw Id = Laptop
{  string[] words = value.toString().Split
                    (" ");
for (String word : words) {
      context.write (new text (word) new
                  Intwritable (1));
}  }
```

(2) Reducer code. java

```
Public class Reducer code extends Reducer
<Text, Int writable> values, context context)
throws Io Exception {
      int sum = 0;
    For (Intwritable val: values) {
        sum += val.get()
    }
context.write(key, new Inwritable
                    (sum));
}
}
```

3) Driver Code

```
Public Class DriverCode {
    publicstatic void main (String[] args)
        throws Exception {
    Configuration conf = new configuration();
    Job job = Job.getInstance(conf,
        "word conf");
    job.setJarByClass(DriverCode.class);
    job.setMapperClasses(mapper.class);
    job.setReducerClass(Reducer.class);
    job.setoutputkeyclass(Text.class);
    job.setoutputvalueclass(Invwritable.
        class);
    FileInputFormat.addInput(Job, newPath
        (args[0]));
    FileOutputFormat.setoutpath(Job, new path
        (arg[1]));
    System.exit(job.waitforcompetition
        (true)?0:1);
}
}
```

Hadoop codes:

```
> hdfs dfs -mkdir -p /input
> hdfs dfs -put /home/hadoop/sample.txt /input
> hadoop jar wordcount.jar Drivercode /input/
  sample.text /output
> hdfs dfs -cat /output/part-r-0000
```

Output

Are	3
Hadoop	1
Hello	1
How	2
I	3
am	1
an	1
are	2
assist	1
Can	2
Coding	1
days	1
during	1
engineer	1
for	2
help	1
interview	1
looting	2
questions	1
strengths	1
there	1
what	2
you	6
your	1

lab 7

MinMax program map reduce

MinMax Mapper

```
public class MinMaxMapper extends Mapper
<object , Text, Text, DoubleWritable> {
    private DoubleWritable value2d = new DoubleWritable();
    private Double localMin = Double.MAX_VALUES;
    private double localMax = Double.MIN_VALUES;
    Contact() throw IoException, InterruptedExcept&;
        double val = Double.parseDouble(value.
                                        toString());
        localMin = Math.min(localMin, val);
        localMax = Math.max(localMax, val);
    }
    protected void cleanup(Context, context) throws
    IoException, InterruptedException {
        contextwrite(new Text("min"), new DoubleWrit
                                        able
        (localMin));
        Context.write(new Text("max"),
                new DoubleWritable(localMax));
    }
}
```

MinMax Reducer

```
public class MinMaxReducer extends Reducer<Text,
 DoubleWritable>{
protected void reduce(Text key,
 Iterable<DoubleWritable> values,
 Context context) throws IOException,
 InterruptedException {
      if (key.toString().equals("min")) {
          double min = Double.MAX_VALUE
           for (DoubleWritable val: values){
               min = Math.min(min, val.get());
      }
     context.write(new Text("Global min"),
      new DoubleWritable(min));
  } else if (key.toString().equals("max")){
          double max = Double.MIN_VALUES;
          for (DoubleWritable val; values) {
               max = Math.max(max, val.get());
          }
          context.write(new Text("Global
                                  max"),
               new DoubleWritable(max));
    }
  }
}
```

bash

```
hadoop com.sun.tools.javac.Main
  MinMaxMapper.java
  MinMaxReducer.java
  MinMaxDriver.java
  jar cf minmax.jar MinMax*.class
  hadoop jar minmax.jar
  MinMax priya /input/data.txt /output_minmax
```

Sample Input File:

```
10
3
55
23
8
91
17
```

Output

```
Mapper 1 exits!
min 3
max 55
mapper 2 exits;
  min 8
  max 91
```

Reducer received:

```
min 3
min 8
max 55
max 91
```

- global min = min(3,8) = 3
- global max = max(55,91) = 91

lab VIII

1) write a scala print no's 1 to 100

```
for (i <- 1 to 100) {
        print (all)
  }
```

o/p

```
1
2
3
.
.
.
98
99
100
```

2) >cd Desktop

>nano file.txt

>realpath ~/Desktop/file.txt

/home/bmsceese/desktop/file.txt

```
> val rdd = sc.textFile("/home/bmsce/
  Desktop/file.txt")

  val words = rdd.flatMap (line => line
    split ("\\s+"))
val pairs = words.map (word => (word,1))
val counts = pairs.reduceByKey(_+_)
filter { case(_, count) => count > 1}
```

```
counts.collect(1, foreach { case(word, count) =>
    println(s"$word $count")
```

3

(3) Add sbt repository keys & repo.

bash

> curl -sL "https://keyserver.ubuntu.com"

```
echo "deb..."
echo "deb..."
> update apt
> sudo apt install sbt
> sbt sbtVersion
```

* make directory

```
>mkdir streaming textcleaner
cd streaming textcleaner
sbt new scalera/scala-seed.g8
```

* Add spark & Lucence Dependicis

```
> nanobuild.sbt
> dependencies added to Streaming
                        textcleaner
```

* $ cd ~/Streaming TextCleaner/ scala-sbt project

$ cd src/main/scalar

$ nano Streaming TextCleaner.scala

$ nano ../../build.sbt

$ nc -lk 9999

On new terminal

$ cd ~/Streaming TextCleaner/scalar-sbt_project

$ sbt run

Back to nc terminal

Enter any input & will be reflected on the new terminal

19/5/25

INDEX

Name : Arbaj Wadagera Class : V th sem

Section : A Roll No. : 1BM22CS051 Subject : BDA Lab

Sl. No.	Date	Title	Page No.	Teacher's Sign. / Remarks
1	3/3/25	LAB-I		
2	10/3/25	LAB - II		
3	17/3/25	LAB - III		
4	24/3/25	LAB - IV		
5	7/4/25	LAB - V		
6	21/4/25	LAB - VI		
7	05/5/25	LAB - VII		
8	19/5/25	LAB - VIII		