INDEX

Name Nithya Lakshmi.V Std 6th SEM Sec D

1BM22CS186

Roll No. Subject BDA LAB School/College BMSCE

School/College Tel. No. Parents Tel. No.

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1.	4/3/25	LAB-1		
2.	11/3/25	LAB-2		
3.	1/4/25	LAB-3		
4.	8/4/25	LAB-4		
5.	15/4/25	LAB-5		
6.	29/4/25	LAB-6		
7.	6/5/25	LAB-7		
8.	20/5/25	LAB-8		
9.	20/5/25	LAB-9		

4/3/25

Date __/__/__
Page _____

LAB-1

1) db.creatCollection("Student");
{ok: 1}

2) use myDB;
switched to db myDB

3) db.Student.drop();
true

4) db.Student.insert(
{_id:1,
StudName: "Nithya",
Grade: "VII",
Hobbies: "InteenetSurfing"});
{acknowledged: true, insertedIds: {'0': 1}}

5) var mystudent=
[{_id: 4, StudName: "ganesh", Grade: "V",
Hobbies: "Dance"},
{_id: 5, StudName: "Kumar", Grade: "VI",
Hobbies: "Singing"}]

6) db.Student.insert(mystudent)
{acknowledged: true, insertedIds:
{'0': 4, '1': 5}}

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

Date __/__/__
Page _____

{upsert:true});

{
acknowledged: true,
insertedId: 3,
matchedCount: 0,
modifiedCount: 0,
upsertedCount: 1
}

7) db.Student.find()
[
{
_id: 1,
StudName: 'Nithya'
Grade: 'VII',
Hobbies: 'InternetSurfing'
},
{_id: 4, StudName: 'ganesh', Grade: 'V', Hobbi
{_id: 5, StudName: 'Kumar', Grade: 'VI', Hobbi
{_id: 8, Grade: 'VII', StudName: 'Aryan
Hobbies: 'Skating'
]

8) db.Student.find({StudName: "Aryan"})
[
{_id: 3, Grade: 'VII', StudName:
'Aryan',
Hobbies:
'Skating'
]

Date __/__/__
Page _____

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

```
[
  { StudName : 'Nithya', Grade : 'VII' },
  { Grade : 'VII', StudName : 'Aryan' }
]
```

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

```
[
  { StudName : 'Nithya', Grade : 'VII' }
  { Grade : 'VII', StudName : 'Aryan' }
```

11) db.Students.find({ Hobbies: { $in: ['chess', 'skating'] } })

```
[
  {
    _id : 3,
    Grade : 'VII',
    StudName : 'Aryan',
    Hobbies : 'Skating'
  }
]
```

4/3/2025

11/3/25

Date __/__/__
Page _____

Lab-02

1. MongoDB lab Exercise perform the following operations using the MongoDB

A:- use myDb;

1. Create a collection by name customers with the following attributes, Cust_id, Acc_Bal, Acc_Type

A:- db.createCollection("Customers");

2. Insert at least 5 values into the table

A:- db:Customers.insertMany([{"cust_id":1, "Acc_Bal": 1500, "AccType": "z"},
{"cust_id":2, "Acc_Bal": 1800, "Acc_Type": "z"}
{"cust_id": 3, "Acc_Bal": 500, "Acc_Type": "x"}])

output

{ acknowledged: true,
InsertedIds: {'0': ObjectId('67...14')
...}}

3. Write a query to display those records whose total account balance is greater than 1200 of acc type "z"

Date __/__/__
Page _____

Ans: db.Customerss.find({"Acc_Bal":{$gt:1200}, "Acc_Type":'2'});

[{_id: ObjectId(67cff...'),
Cust_id:1,
Acc_bal:1500,
AccType:'2'}

{_id: objectId('67...18')
Cust_id:4
Acc_Bal:1300
Acc_Type:'2'}]

4. Determine minimum and maximum account balance for each Customer_id

db.customerss.aggregate([
{$group:{
_id:"$cust_id",
min_bal:{$min:"$Acc_Bal"},
max_bal:{$max:"$Acc_Bal"}
}}]);

[
{_id:2, min_bal:1800, max_bal:1800}
{_id:4, min_bal:1300, max_bal:1300}
{_id:3, min_bal:1500, max_bal:800}
]
// output //

Date __/__/__
Page _____

Lab-03

1. Create keyspace students with replication
{'class' = SimpleStrategy, 'replication_factor' : 1};
2. Describe keyspaces
students system. system_schema
3. SELECT * FROM system_schema.keyspaces;
4. USE students;
5. CREATE TABLE students_info (Roll no int PRIMARY KEY, Studname text, Date of Joining Timestamp, Last_exam_percent double);
6. DESCRIBE TABLE students_info;
7. Insert values
BEGIN BATCH
insert into students_info (Roll_no, StudName Date of Joining, Last exam_Percent) values (1, 'Dash', '2012-03-12', 79.8)
Insert into students_info (Rollno, Studentname, Date of Joing, Last exam Percent)
values (2, Tarun, 2012-03-12, 90.9)

Date __/__/__
Page _____

APPLY BATCH;

(8) select * from Students_info;

(9) SELECT * FROM Students_info WHERE Roll_no IN (1,3,3);

(10) CREATE INDEX ON Students-info (Stud-Name)

(11) Select * from Student_Info where Studname "Asha"

UPDATE

UPDATE Student_info SET StudName 'David_Ahsen' WHERE Roll_no = 2

DELETE LastExamDate FROM Students info WHERE Roll_No = 2

ALTER TABLE Students_info ADD Hobbies set <text>

UPDATE Student-info SET Hobbies = Hobbies + {'Class Table Tennis'}

Where RollNo. = 1;

CREATE TABLE lib_book (CountervalId, Counter, bookname, valdat, PK (bookname, Studname)

Date __/__/__
Page _____

Loading Data

Update lib_book SET counter_value = counter_value + 1 WHERE book_name = 'BDA'

Time to live

Create Table login(userid int PRIMARY KEY, text) USING TTL = 30;

INSERT INTO login(userid, p) values('1', 'int') USING TTL = 30;

SELECT TTL(password) FROM login WHERE userid = 1

Export to CSV

COPY classlist (id, courseid, coursename) TO 'd:\class.csv'

Import to CSV

COPY classlist (id, courseid, coursename) FROM 'd:\class.csv';

8/4/25

LAB-4

Perform the following DB operations using Cassandra:-

1) Create a keyspace by name Library

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

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

```
USE Library;

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)
);
```

Date / /
Page

```
CREATE TABLE Book_Counter (
    Stud_Id int,
    Book_Name text,
    Counter_Value counter,
    PRIMARY KEY (Stud_Id, Book_Name)
);
```

3) Insest the values into the table in batch

```
BEGIN BATCH
INSERT INTO Library_Info (Stud_Id, Stud_Name, Book_Name, Book_Id, Date_of_Issue)
VALUES (112, 'Rahul', 'BDA', 'BK102', '2025-04-08');

INSERT INTO Book_Counter (Stud_Id, Book_Name)
VALUES (112, 'BDA');

UPDATE Book_Counter SET
Counter_Value = Counter_Value + 1
WHERE Stud_Id = 112 AND Book_Name = 'BDA';

APPLY BATCH
```

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

Date / /

Page

X) Write a query to show that a student with id 112 has taken a book "BDA" 2 times

```
SELECT * FROM Library_Info;

UPDATE Book_Counter SET Counter_
value = Counter_value + 1
WHERE Stud_Id = 112 AND Book_Name
= 'BDA';

SELECT * FROM Book_Counter;
```

Output:

Stud_id	book_name	book_id	date_of_issue
112	BDA	BK102	2025-04-08

Stud_name
Rahul

Stud_id	book_name	counter_value
112	BDA	2

Date / /
Page

5) Write a query to show that a student with id 112 has taken a book 'BDA" 2 times

```
SELECT Counter_Value FROM Book_Counter
WHERE Stud_Id = 112 AND Book_Name = 'BDA';
```

output

Counter_value
2

6) Export the created column to a csv file

```
COPY Library_Info TO 'library_Info.CSV'
WITH HEADER = TRUE;
```

output

1 rows exported to 1 files in 0.135 secs

Date __/__/__
Page _____

7) Import a given csv dataset from local file system into cassandra column family

```
COPY Library_Info (Stud_Id, Stud_Name,
                   Book_Name,
                   Book_Id,
                   Data_of_Issue)

FROM 'Library_Info_new.csv'
WITH HEADER = TRUE;
```

8/4/25

15/4/25

LAB-5

Hadoop Commands

1) mkdir

start-dfs.sh

hdfs dfs -mkdir /cdf

2) ls

hadoop fs -ls /

3) put

mkdir -p /home/hadoop/Desktop
"Hello Hadoop World!" > /home/hadoop/Desktop/Welcome.txt

4) copyFromLocal

hdfs dfs -copyFromLocal -f /home/hadoop/Desktop/Welcome.txt /cdf/WC.txt

5) get

hdfs dfs -get /cdf/WC.txt /home/hadoop/Desktop/WWC.txt

Date __/__/__
Page _____

6) copyToLocal

hdfs dfs -copyToLocal /cdf/WC.txt/ home/hadoop/ Desktop

7) cat

hdfs dfs -cat /cdf/WC.txt
Hadoop World!

8) mv

hadoop fs -mv /cdf /FFF

9) cp

~~hadoop fs -ls /~~
hadoop fs -cp /rgs / /FFF /

15/4

29/4/25 LAB - 6

Date __/__/__
Page

Word Count

mapper.py

```
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()

    for word in words:

        print('%s\t%s' % (word, 1)
```

Reduce step:
reducer.py

```
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:

    line = line.strip()
    words, count = line.split('\t', 1)

    try:
        count = int(count)
    except ValueError:
        continue
```

Date __/__/__
Page _____

```
if current_word == word:
    current_count += count
else:
    if current_word:
        print '%s\t%s' % (current_word,
                          current_count)
    current_count = count
    current_word = word

if current_word == word:
    print '%s\t%s' % (current_word,
                      current_count)
```

TopN

mapper.py

```
import sys
import re

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

reducer.py

```
import sys
from collection import defaultdict

current_word = None
current_count = 0
```

Date __/__/__
Page

LAB-7

```
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t
                {current_count}")

        current_word = word
        current_count = count

    if current_word == word:
        print(f"{current_word}\t
            {current_count}")
```

MeanMax

Mapper.py

```
import sys

for line in sys.stdin:
    line = line.strip()
    if len(line) < 93:
        continue
    try:
        month = line[19:21]
        if line[87] == '+':
```

```
        temperature = int(line[88:92])
    else:
        temperature = int(line[87:92])
    quality = line[92:93]
    if temperature != 9999 and quality
                                in ['0', '1',
                                '4', '5',
                                '9'];
        print(f"{month}\t{temperature}")
except ValueError:
    continue
```

reducer.py

```
import sys

current_month = None
temps = []

def emit_mean_max(month, temps):
    if not temps:
        return
    total_temp = 0
    days = 0
    for i in range(0, len(temps), 3):
        chunk = temps[i:i+3]
        max_temp = max(chunk)
        total_temp += max_temp
        days += 1
    if days > 0:
        mean_max = total_temp
        print(f"{month}\t{mean_
                            max}")
```

Date __/__/__
Page _____

```
for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    month, temp = line.split('\t')
    temp = int(temp)
    if current_month != month:
        if current_month is not None:
            emit_mean_max(current_month, temps)
        current_month = month
        temps = [temp]
    else:
        temps.append(temp)

if current_month is not None:
    emit_mean_max(current_month, temps)
```

Date __/__/__
Page _____

LAB-8

Scala

Install. scala

```
val x : Int 5
x = Int = 5

vary y = "scala"
   y : String = scala

if ( x > 3) println ("cricket")
  Gender

for (i <- 1 to 5) println (i)
1
2
3
4
5

def add(a Int, b: Int) Int = a+b
add (a : Int, b : Int) Int

println (add(3, 4))

val List = List (1,2,3)
List : List (Int) = List (1,2,3)
```

Date __/__/__
Page _____

list.foreach(println)
1
2
3

val map = Map("a" -> 1, "b" -> 1)
map scala.collection.mutable.Map
(a -> 1, b -> 2)

println(map("a"))

class person(name : String)
def sed() = println("Helloworld")
defined class person

val p = new person("Alice")
p.person.person = 78124

print(no.chars)
def read().int

class English Greater extends greater
def greet() = println("Hi")
defined class English Greet

g.greet()
Hello

def describe(x : Any) : String = x match
{ case 1 => "one"
case one = "The"
case _ => "The cluster"}
describe(x : Int) : String

Date / /
Page

LAB-9

1. object print no{
    def name(args : array(str))
        for (i <- 1 to 100){
            println(i)
        }
    }
}

2. import org.apache.spark.{spark conf, spark content}
val lines = sc.textFile("input text)
val words = lines.flatMap(line})
val wordPairs = words.map(word)
(word,1))

val wordcounts = word counts.filter
{case (word,count)
=> count ≥ 4)

Extend words - count() word each
(count (word count)
println (s" print - $ wordcount}
(spark, b)

3. Hadoop
Mapper.py

import sys
import re
for line in sys.stdin.
    line = line.strip().lower()
    words = append(a/b(a-2)+'b']b)

Date __/__/__
Page _____

for word in words:
print("words")

Reducer.py
import sys
from collection import default dict
word_count = default(int)
for line in sys.stdin:
word, count = line.split('\t')
word_count[word] = int(count)
Mapwords = sorted(word_count.items())
(x[1], x[0]), [:10]
for count, count in sorted(top_words, key=lambda x: x[1], reverse=True):
print("%s\t%s" % (word, count))

a 5 is 5
and 7 for 5
apple 4 of 5
data 8 words
hadoop 5