INDEX

NAME: Sailesh STD.: SEC.: 6D ROLL NO.: SUB.: BDA lab Manual.

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks

BYJ0719083 A

Cassandra

1. Employee Information:

Setting Cassandra up:

→ cd cassandra/apache-cassandra-3.11.0/
cd bin
cqlsh → starts Cassandra.

Create Namespace:

→ CREATE KEYSPACE EMPLOYEE WITH REPLICATION = {'class': 'SimpleStrategy', 'replication_factor': 1};
DESCRIBE KEYSPACE;
use EMPLOYEE;

Creating Column Family:

→ Create table employee_info (emp_id int, emp_name text, designation text, date_of_joining timestamp, salary int, primary key (emp_id));
describe tables;
describe table employee_info;

Inserting Values:

→ begin batch insert into employee_info (emp_id, emp_name, designation, date_of_joining, salary)
values (1, 'Snehita', 'chairperson', '2020-03-8', 400000) apply batch;
select * from employee_info; → shows all entered data.

Update values:

→ Update employee_info set emp_name = 'Rohit' where emp_id = 121;
select * from employee_info; → shows all data, with updated value.

Alter & update. → adding new row in table.

→ alter employee_info add project_list set{

→ update employee_info set project_list = project_list + {'voice recognition', 'Project 3'} where emp_id = 3;

⋮

+{'Image recognition', 'Project 4'}
where emp_id = 4';
select * from employee_info;

TTL

→ insert into employee_info(emp_id, empname, designation, date_of_joining, salary, dept_name) values(7, 'Rohit', 'kathmandu', '2020-05-04', '400000', 'Computer Engineer') using ttl 10;

~~select~~ :
SELECT EMP_NAME, TTL(EMP_NAME) FROM EMPLOYEE_INFO WHERE EMP_ID = 7;

↓

~~shows ttl~~ shows table.

(i.e)

emp_name	ttl(emp_name)
Navin	10

2. Library Information :

① Create Keyspace and Column family :

→ create table library_info(studIdint, StudName text, bookName text, bookID int, date of Issue timestamp, count value count, primary key(studId));

describe tables;
describe table employe_info;

② Inserting Values

→ begin batch insert into library_info(studId, StudName, bookName, bookID, date of Issue) values(1, 'Ankit', 'Big data', 345, '2020-01-02') apply batch;

select * from library_info; → shows table.

③ Counter → updating table → set to counter value = 2 from specified column.

→ update library_info set countervalue = countervalue + 2 where studId = 3 and StudName = 'Ronit' and bookname = 'BDA' and bookid = 111 and date of issue = '2020-01-02';

select * from library_info;

↓

shows updated table.

⑤ ~~Update~~ :

-> ~~Update~~

⑤ Copy :

-> Copy library_info (StudId, StudName, bookID, date of Issue) to '/home/bnse/Desktop/[redacted]/library_data.csv'

Mongo DB CRUD

(1) Database, table creation and inserting values:

→ use students;

db.details.insertOne({"roll no":"1BM19CS210", "age":21, "Contact no":"912356", "email id":"saileshtrust@gmail.com"})

db.stu_details.find().Pretty(). → shows details stored

(2) Update:

→ db.stu_details.update({"_id":ObjectId("608...46")}, {$set:{"email id":"new email id"}});

(3) Import:

→ import --db student --collection stu_details --file stu_details.json.

(4) Export.

→ mongoexport --collection=stu_details --db=student --out=stu_details.json,

Scala:

1. Program to run wordcount on scala shell.

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

2. 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("/home/bhoom/Desktop/wc.txt")
val counts = textFile.flatMap(line => line.split(" "))
map(word,1)).reduceByKey(_+_)
import scala.collection.immutable.ListMap
val sorted = ListMap(counts.collect.sortWith(_._2
>_._2):_*)
Println(sorted)
for ((k,v) <- sorted)
{
  if (v>4)
  {
    Print(k+" ,")
    Print(v)
    Println()
  }
}
```

4. Program to run wordcount on scala IDE.

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.rdd.RDD.rddToPairRDDFunctions.
object WordCount{

def main(args: Array[String]) = {

  val conf = new SparkConf().setAppName
   ("WordCount").setMaster("local")
val sc = new SparkContext(conf)

val test = sc.textFile("input.txt")

test.flatMap{line => line.split(" ")}.map{
word => (word,1)
}.

.reduceByKey(_+_)
.saveAsTextFile("output.txt")
sc.stop
 }
}
```

3 Execute Hello World Program in SCALA IDE. Follow the steps given in:

https://www.dataneb.com/post/hello-world-with-scala-ide.

=>

1. Open Eclipse Scala IDE.

2. Create a new Scala project "helloworld"
   - Go to file
   - New
   - Project and enter helloworld in project name field and click finish.

3. Create a new Scala Package "helloworld".
   - Right click on the helloworld project in the Package Explorer panel
   - New
   - Package and enter name helloworld and finish.

4. Create a Scala object "hello".
   - Expand helloworld project tree and right click on the helloworld package
   - New
   - Scala object.
   - enter hello in the Object name field and press finish.

5. Write Program to Print Hello World message.

```
Package helloworld
object hello {
    def main (args: Array [String]) {
        Println ("Hello World")
    }
}
```

6. Create a Run configuration for Scala application :

- Right click on hello.scala in package explorer
- Run As
- Scala Application. Select first matching item, the HelloWorld class and click ok.
- OR you can also define configuration manually and run it. Just mention the project and class name.

∴ This is how we can run Helloworld Program in SCALA.

Hadoop Word Count:

Commands:

-> hdfs namenode -format
-> cd hadoop-3.3.0/sbin
-> start-dfs
-> start-yarn
-> jps.
-> hdfs dfs -mkdir /input_dir.
-> hdfs dfs -ls /
-> hdfs dfs -copyfromLocal D:/input_file.txt /input_dir.
-> hdfs dfs -cat /input_dir/input_file.txt
-> MapReduceClient.jar
-> hadoop jar D:/MapReduceClient.jar wordcount /input_dir /output_dir.
-> hdfs dfs -cat /output_dir/*

Input file:

Hello World
Hi World
Hello World
Wassup World

Hadoop Temperature: (Mean Max Temperature)

Driver

```
package meanMaxTemp;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapredue.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
Public class Mean Max Driver
{
    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);
        System.exit(job.waitForCompletion(true)?
        0:1);
    }
}
```

Mapper

```
Package mean MaxTemp;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import java.io.IOException;
Public Class MeanMaxMapper extends Mapper<Long
Writable, Text, Text, IntWritable> {
    Public static final int MISSING=9999;
    Public void map(Long Writable key, Text
    Value, Context context) throws IOException,
    InterruptedException
    {
        String line = value.toString();
        String month = line.substring(13,21);
        int temperature;
        if (line.charAt(87)=='+')
            temperature = Integer.parseInt
            (line.substring(88,92));
        else
        temperature = Integer.parseInt(line.substring
                (87,92));
        String quality = line.substring(92,93);
        if(temperature!=MISSING && quality.
        matches("[01459]"))
            context.write(new Text(month),new
            IntWritable(temperature));
    }
}
```

Reducer:

```
Package mean MaxTemp;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred*;
import java.io.IOException;

Public class MeanMaxReducer extends Reducer
<Text, IntWritable, Text, IntWritable>{
   Public void reduce (Text key, Iterable<
IntWritable> values, Context context) throws
IOException, InterruptedException{
      int max_tem = 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 += 1;
      if (count == 3){
         total_tempt = max_temp;
         max_temp = 0;
         count = 0
         days += 1;
      }
   }
   Context.write (key, new IntWritable(
         total_temp/days));
}
}
```