Date 4/3/25
Page

Lab-1

Working with MongoDB

1. Create Database in MONGODB

→ use myDB

2. CRUD

- Create a Database

→ db.createCollection("student");

{ ok : 1 }

- Delete a Database

→ db.student.drop()

true

- insert a record

→ db: Student.insert({_id : 1, USN : 174, Name : "Navneeth", age : 20})

{ ok : 1 }

- db.Student.update({_id: 2, USN : 203, Name : "Pranav", age : 21}, {$set : { Name : "Pranav"}, {upsert : true}});

• to find

→ db.student.find({Name: "Navneeth"})

{
_id: ObjectId("...."),
USN: 194
Name: 'Navneeth',
age: 20
}

• to get the student count

db.Student.count()

→ 3

• to sort records

db.student.find().sort({Name: 1}).pretty();

→ [

{_id: 1, USN: 174, Name: 'Navneeth', ag
{_id: 2, USN: 203, Name: "Pranav"

]

Date __/__/__
Page _____

• add a new.

db.Student.update(
{_id: 2}, {$set: {Location:
"Network"}})

{ok: 1}

• Remove the field

db.Student.update({
_id: 4}, {$Location: "Network"})

{ok: 1}

• to set a particular field to null,

db.Student.count()

→ 3

th", age: 20},
2 av", age: 21}

- to find mango in fruit array

db.food.find({ fruits: ['grapes', 'mango', 'apple'] }).pretty()

→ mango

- Aggregate function

→ db.customers.aggregate([{ $group: { _id: "$custId", TotAccBal: { $sum: "$AccBal" } } }]);

→ [
  {
    "_id": "customer123",
    "TotAccBal": 15000
  },
  {
    "_id": "customer124",
    "TotaAccBal": 20000
  }
]

4/3/25

11/3/25

Date / /
Page

Lab - 2

Cassandra

* ~~MongoDB~~ operation for Customers collection

1) Creat collection and insert data

```
db.Customer.insertMany([
  { Cust_id: "C001", Acc_Bal: 1500,
    Acc_Type: "Z" },
  { Cust_id: "C002", Acc_Bal: 800,
    Acc_Type: "Z" }
]);
```

2) Query for Acc_Bal > 1200 & Acc_Type = 'Z'

```
db.customers.find({
  Acc_Bal = { $gt: 1200 },
  Acc_Type = "Z"
});
```

3) Minimum & maximum acc balance per customer

```
db.Customers.aggregate([
  {
    $group: {
      _id: "$cust_id",
      minBalance: { $min: "$Acc_Bal" },
      maxBalance: { $max: "$Acc_Bal" }
    }
  }
]);
```

E-commerce Platform design

schema design

```
db.createCollection("Products");
{
    _id: ObjectId(),
    productId: String,
    name: String,
    category: String,
    price: Number,
    quantity: Number
}

db.createCollection('carts');
{
    _id: ObjectId()
    userId: String,
    products: [
        {
            products: [
                {
                    productId: String,
                    quantity: Number
                }
            ]
        }
    ]
}
```

```
db.createCollection("order");
{
    _id: ObjectId(),
    userId: String,
    products: [
        {
            productId: String,
            quantity: Number
            price: Number
        }
    ],
    orderDate: Date,
    totalPrice: Number
}
```

Insertion

```
db.products.insertMany([
  {prdId: P001, name: "Laptop",
   category: "Electronics", price: 999,
   quantity: 5},
  {productId: "P002", name: "shirt",
   category: "clothing", price: 25,
   quantity: 20},
]);

db.carts.insertOne({
  userId: "719ghi",
  products: [
    {productId: "P001", quantity: 1}
    {productId: "P002", quantity: 2}
  ]
});
```

```
db.orders.insertMany([
  { userId: "123abs", product:
    [{ productId: "P001" }]
])
```

Queries

1. Retrieve All Products

```
db.Products.find();
```

2. Retrieve Products in category electronics

```
db.Products.find({ category:
"Electronics" });
```

3. Retrieve products with quantity > 0

```
db.Products.find({ quantity:
  { $gt: 0 } });
```

4. Retrieve Products sorted by ascending order

```
db.Products.find().sort({ sort: 1 });
```

5. ~~db.Products.find({ p~~ Retrieve all products with price <= 100

6 Retrieve Products

```
db.carts.aggregate([
  { $match: { userId: "789ghi" } },
  { $lookup: {
      from: "products",
      localField: "productId",
      foreign: "productId"
    }
  }
])
```

7 Retrieve orders by user

```
db.orders.find({ userId: "123abc" });
```

8 Retrieve total price

```
db.orders.aggregate([
  { $match: { userId: "123abc" } },
  { $group: { _id: "$userId"
```

```
]);
```

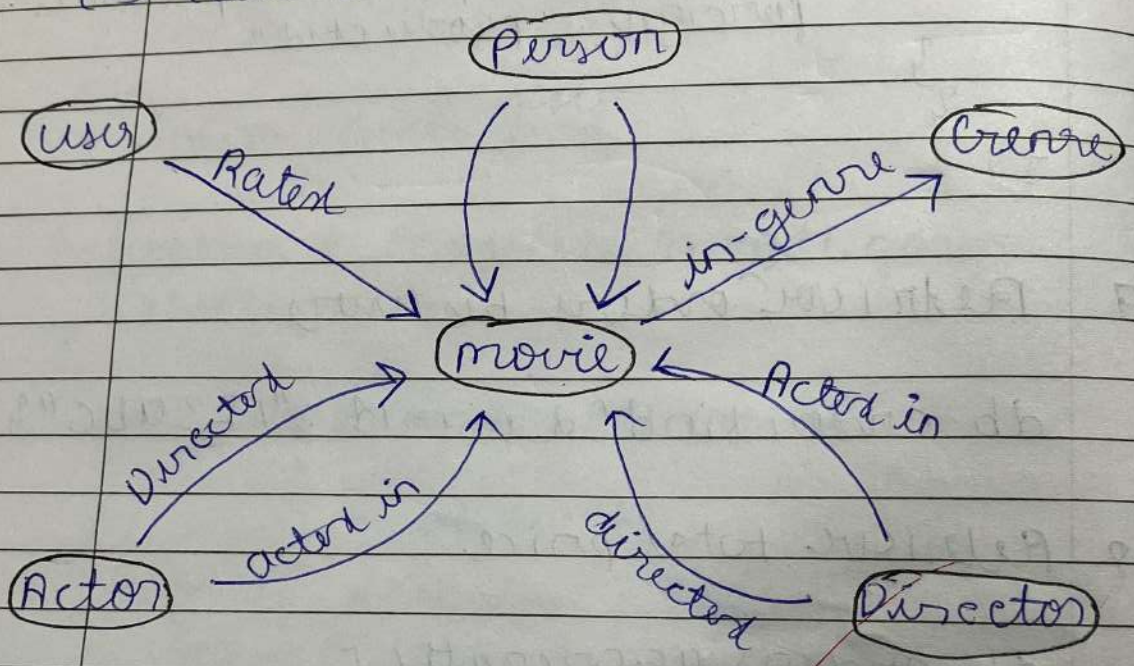
18/3/25

Lab 3

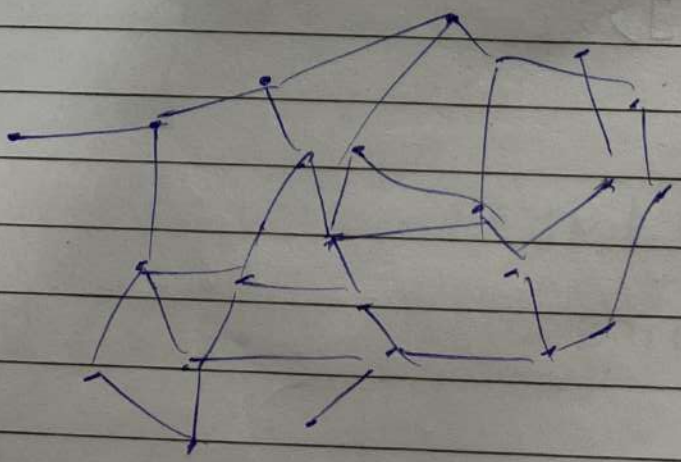
Follow the commands for 3D visualization in ~~no~~ neo4j@bolt

(1) use neo4j

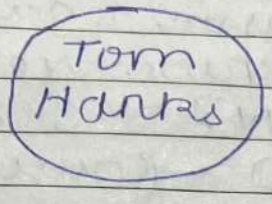
(2) call db.schema.visualization



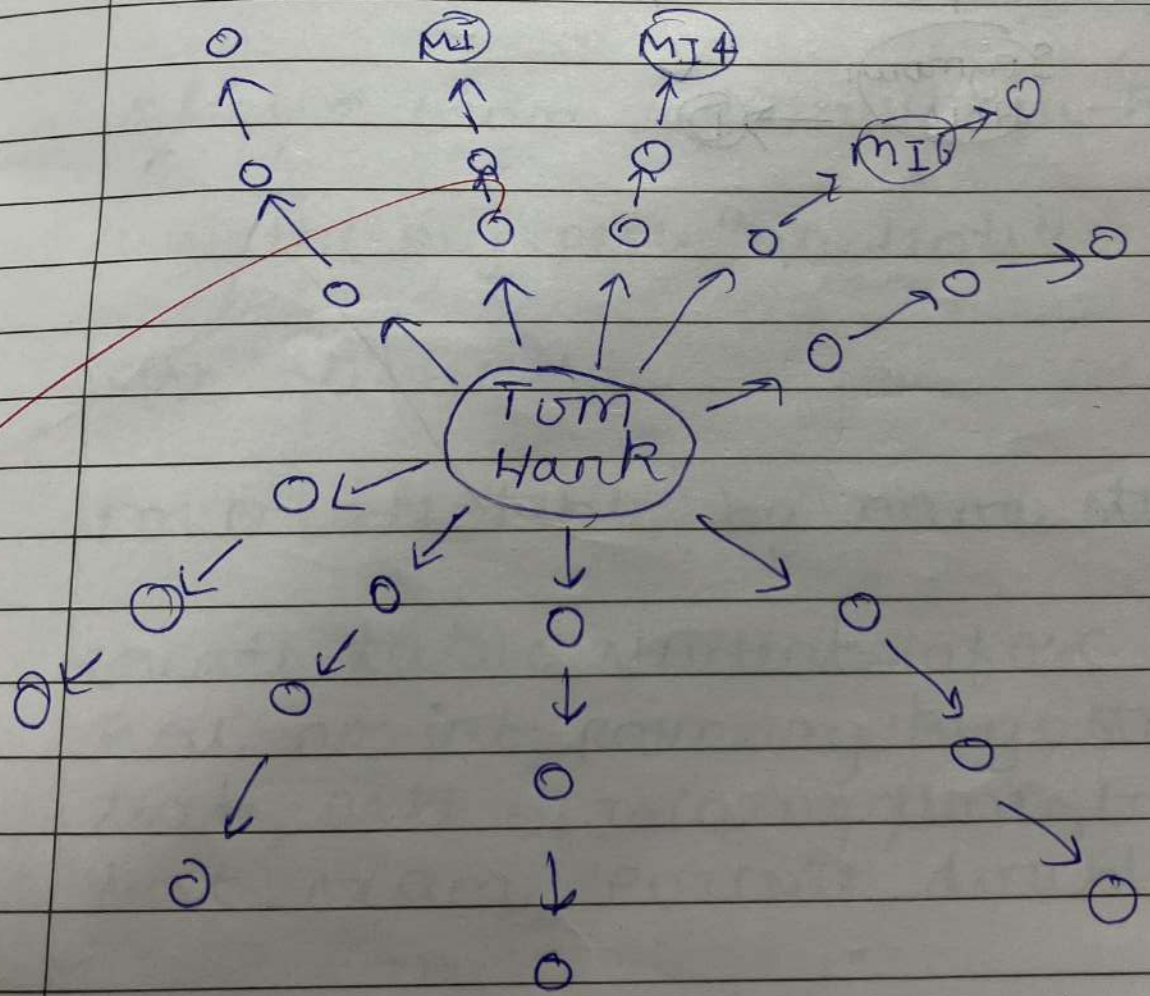
(3) match (n) return n limit 100



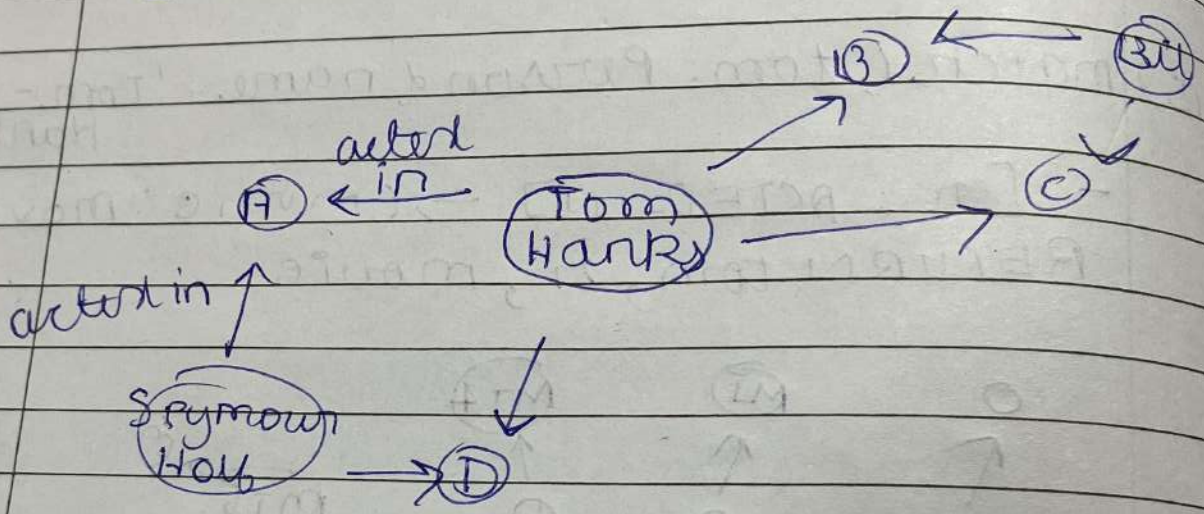
(4) match (tom: Person { name: 'Tom Hanks' }) Return tom



(5) match (tom: Person { name: 'Tom Hanks' })
- [r : ACTED_IN] -> (movie : Movie)
RETURN tom, r, movie



(6) Match (tom: Person {name: 'Tom Hanks'}) <-[:ACTED_IN]- (coActor: Person) -[:ACTED_IN]-> (cruise: Person {name: 'Tom Cruise'}) WHERE NOT (tom) -[:ACTED_IN]-> (:Movie) <-[:ACTED_IN]- (cruise)



1/4/25

Date __/__/__
Page

Cassandra

Create Keyspace

create Keyspace students with replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

Describe the existing Keyspace

describe keyspace

For more details on existing Keyspace

select * from system_schema.keyspaces

use the Keyspace "students"

use students;

To create table by name student info:

create table Student_info (
Roll_no int primary key, Studname text, Date_of_joining timestamp, last_exam_Percent double)

Describe the table information

describe table <table_name>

CRUD

insert:

insert into student_info (Roll_no, StudentName, DateOfJoining, last_exame_percent)
values (1, 'Asha', '2012-03-12', 79.4)

view data

select * FROM student_info;

view data from table "student_info" where RoomNo column either has a value 1 or 2 or 3

select * from student_info
where Roll_no IN (1, 2, 3)

So create an index on the column as below

Create index on student_info (Stuname)

Now execute the query based on the indexed column

select * from students_info where

Date / /
Page

studname = 'Asha'

update

- update students-info set StudName = 'David Shen' where RollNo = 2;

delete

- delete LastExamPercent from student_info RollNo = 2

set collection

- alter table students_info add hobbies set<text>

USING A COUNTER

a counter is a special column that is changed in increments

create table library-book (counter_value counter, book name, stud_name));

Date
Page

load data into the counter column

update library_book SET
counter_value + 1 where book_name
= 'Big data Analytics' and
student_name = 'jeet'

Time to live

Import and Export

export to csv

COPY learninglists (id_course_
order, course_id, title)

Date 8/4/25
Page

Lab-5

1. Create keyspace

```
create keyspace Library with replication
= {'class': 'SimpleStrategy', 'replication
  _factor': 1};
```

2. Table for student and book

```
create table Library.Library-info(
  Stud_Id int,
  Stud_Name text,
  Book_Name text,
  Book_Id int,
  Date-of-Issue date,
  Primary Key (Stud-Id, Book-Id)
);
```

~~Counter table~~

```
create table Library.BOOK-Counter(
  Stud_Id int
  Book_Name text,
  Counter_value counter,
  Primary Key (Stud-Id, Book_Name)
);
```

Date
Page

3. Insert the value into table in batch

```
begin batch
insert into Library.Library_Info (
stud_id, stud_name, Book_name, Book_id,
values (112, 'Rahul', 'BDA', 201);

update Library.Book_counter
set counter_value = counter_value + 1
where stud_id = 112 and book_name = 'BDA';

apply batch
```

4. Display the table and increment counter

```
select * from Library.library_info;

update library.book_counter
set counter_value = counter_value + 1
where stud_id = 112 and book_name
= 'BDA';

select * from Library.Book_counter;
```

5. select counter_value from Library.Book_counter where stud_id = 112 and Book_name = 'BDA';

6.

7.

6. export column family to csv

copy Library.Library_info to 'Library_info.csv' with header = true;

7. import a csv file into column family

copy library.library_info (stud_id, stud_name, book_name, book_id, date_of_issue) from 'Library_info.csv' with header = true;

15/4/25

Lab 6

Hadoop Commands

1. mkdir
→ Creates one/more directories in HDFS
→ hdfs dfs -mkdir <path>

2. ls
→ list the contents of a dir
→ hadoop fs -ls <path>

3. put
→ copies files/directories from the local file system to HDFS
→ hdfs dfs -put <source> <destination>

4. copyfromLocal
→ copies files from local file system to HDFS
→ hdfs dfs -copyfromLocal <source> <destination>

5. get
→ retrieves files/dir from HDFS to local file system
→ hdfs dfs -get <source> <des>

6 copyToLocal
→ copies files from HDFS to local file system
→ hdfs dfs -copyToLocal <source> <des>

7. cat
→ displays the contents of a file in HDFS
→ hdfs dfs -cat <file-path>

8 mv
→ moves file/dir to with HDFS
→ hadoop fs -mv <source> <des>

9. cp
→ copies/files within HDFS
→ hadoop fs -cp <source> <destination>

Commands

1. start-all.sh
→ initializes all Hadoop

2. create a dir in HDFS
→ hdfs dfs -mkdir /bda-hadoop

3 list HDFS contents
→ hadoop fs -lp/

4. copy file from local to HDFS
→ hdfs dfs -put /home/hduser/Desktop

5. hdfs dfs -copy From Local /home/hduser

6. ~~hdfs dfs -cp~~

6 Display file contents
→ hdfs -dfs /bda_hadoop/file-cp.txt

traffic

Lab - 7

From the following link the extract the weather data

Create a map reduce program to,

a) Find avg temp for each yr from NCDC data set
b) Find the mean max temp for every month

Create a three java classes into the project.

A.

```
import java.io.IOException

public class WCMapper extends
MapReduceBase implements Mapper
<LongWritable, Text, Text, IntWritable>{

  String line = value.toString();

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

Date __/__/__
Page _____

```
// Reducer code

public class WCReducer extends
MapReduceBase implements
Reducer <Text, IntWritable, Text, I

  public void reduce (Text key, Iterator
  <IntWritable>) throws IOException
  {
    int count = 0
    while (value.hasNext()){
      IntWritable i = value.next();
      count += i.get()
    }

    output.collect(key, new IntWritable());
  }
}

// Driver code

public class WCDriver extends
Configured implements Tool{

  public int run (String args[])
  throws IOException{
    if (args.length < 2)
      return -1;
    }
  JobConf conf = new JobConf (WCDriver
                              .class
```

Date
Page

20/5/2

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

count.setMapperClass(WCMapper.class);
conf.setMapperClass(WCReducer.class);
return 0;
}

// main method

public static void main(String args[]){
int exitCode = ToolRunner.run(
new WCDrivers(), args);
}
}

20/5/25

Lab - 8

Q Write a scala programs to print numbers from 1 to 100 using for loop

→ code

```
object PrintNumbers {
  def main(args: Array[String]):
      Unit = {

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

output

1
2
.
.
.
100

Date
Page

2015125

Lab-9

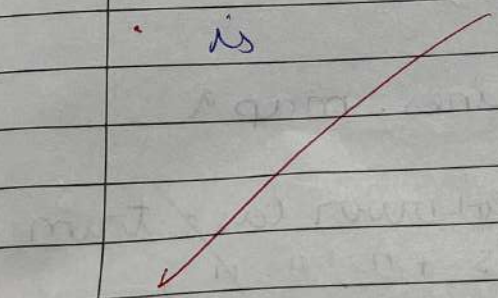
· using RDD and flatmap to count the occurance of each word

```
import org.apache.spark.
    {SparkConf, SparkContext}

object WordCount {
  def main(args: Array[String]): {
    val conf = new SparkConf().
      setAppName("WordCount").
      setMaster("local")
    val sc = new SparkContext(conf)

    val file = sc.textFile("path/file.txt")

    val wordCounts = file.flatMap
    (line => line.split("\\W+")).
    map(word => (word.toLower
      (case(_)))).reduceByKey(_+_=>

    val filteredWords = wordCounts.
      filter{case(word, count)=>
        count > 4}

    filteredWords.collect().foreach
    {case(word, count) =>
      println(s"$word: $count")
    }
```

```
    sc.stop()
  }
}
```

output:

word	count
hello	1
this	2
new	3
work	1
is	1



20/5/25

Date / /
Page

2015125

Lab - 10

Write a simple streaming prg in Spark to receive text data streams on a particular port, perform basic cleaning and print the output.

```
import org.apache.spark._
import org.apache.spark.streaming._

val ssc = new StreamingContext(
  sc, Seconds(5))

val lines = ssc.socketTextStream
("localhost", 9999)

val cleanedLines = lines.map(
line =>
  val low = line.toLowerCase.trim
  .replaceAll("\\s+", " ")

  val noPunct = low.replaceAll
      ("[^a-zA-Z\\s]", "")

val tokens = noPunct.split(" ").
  filter(token => !stopWords.
    contains(token))

val lemmalized = token.map(
word => word)
```

lemmatized.mkString(" ")

ssc.start()
ssc.awaitTermination()

output

stream

Hello World the weather is cold

→ Hello World the weather cold