USN | 1 | C | R | 2 | 2 | I | S | 0 | 6 | 2 |

BIS701

Seventh Semester B.E./B.Tech. Degree Examination, Dec.2025/Jan.2026
Big Data Analytics

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
		Module – 1			
Q.1	a.	Describe the 3 V's of Big Data and discuss the challenges faced with Big data.	10	L2	CO1
	b.	Explain the classification of digital data.	10	L2	CO1
		OR			
Q.2	a.	Discuss Big Data Analytics and explain the following terminologies : i) Symmetric Multiprocessor System ii) CAP Theorem.	10	L2	CO1
	b.	Explain the features and advantages and NOSQL. Discuss the types of NOSQL data bases.	10	L2	CO1
		Module – 2			
Q.3	a.	Discuss the need for Hadoop and its high level architecture.	10	L2	CO2
	b.	Illustrate MapReduce process with a word count example.	10	L3	CO2
		OR			
Q.4	a.	Discuss the limitation of HDFS and its solution. Explain the YARN architecture.	10	L3	CO2
	b.	Implement a MapReduce program in Java/Python/R to implement matrix multiplication.	10	L4	CO2
		Module – 3			
.5	a.	Discuss replication and Sharding in MongoDB.	10	L2	CO3
	b.	Illustrate the CRUD operations using MongoDB query language with examples.	10	L3	CO3
		OR			
.6	a.	Demonstrate the following operations in MongoDB query language with examples : i) Count ii) Limit iii) Sort iv) Skip.	10	L2	CO3
	b.	Explain the application of the following in MongoDB i) Cursors ii) Indexes iii) MongoExport iv) Aggregate function.	10	L2	CO3
		Module – 4			
7	a.	Discuss the features of Hive. Explain the Hive architecture.	10	L2	CO3
	b.	Explain the DDL and DML commands in Hive.	10	L2	CO3
		OR			
8	a.	Express the features and philosophy of Pig. Discuss ETL processing.	10	L2	CO3
	b.	Discuss the following in Pig. i. Relational operators – Foreach and Limit ii. Complex data types – Tuple and Map.	10	L2	CO3
		Module – 5			
9	a.	Discuss the features of spark. Explain the spark software stack.	10	L2	CO4
	b.	Explain the steps involved between acquisition of data from multiple sources and its application in spark.	10	L2	CO4
		OR			
10	a.	Discuss text mining and its applications. Explain the process of text minng.	10	L2	CO4
	b.	Implement a word count program in Hadoop and spark using Java/Python/R.	10	L4	CO4

* * * * *

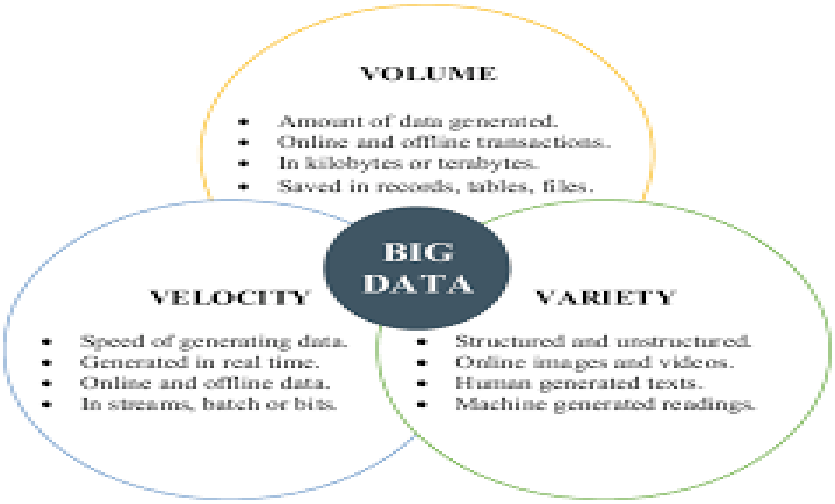
USN										

VTU Examination – Dec. 2025 / Jan. 2026
Scheme of Evaluation

Sub:	**Big Data Analytics**					Sub Code:	**BIS701**	Branch:	**ISE**
Date:	**14/01/2026**	Duration:	**3 hrs**	Max Marks:	**100**	Sem/Sec:	**VII/ A, B & C**		**OBE**

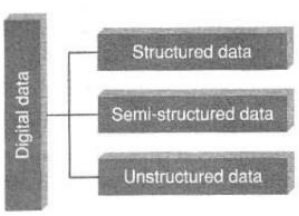
Answer any FIVE FULL Questions

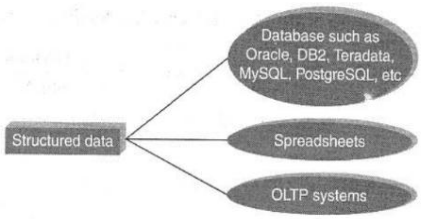
MODULE -1

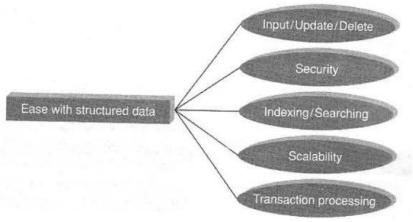
		MARKS	CO	RBT
1.	a. Describe the **3 V's of Big Data** and discuss the challenges faced with Big Data. **Scheme: Definition and explanation of 3V's + Challenges – 5+5 marks** **Solution:**   	10	1	L2
	b. Explain the **classification of digital data**. **Scheme : List + Explanation – 2+8 marks**			

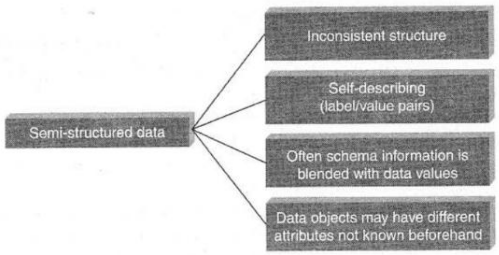
Solution:

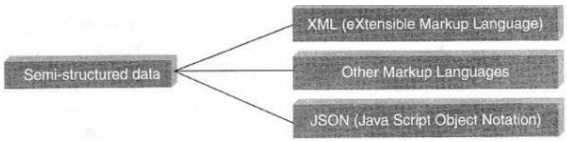
1. **Unstructured data:** This is the data which does not conform to a data model or is not in a form which can be used easily by a computer program. About 80–90% data of an organization is in this format; for example, memos, chat rooms, PowerPoint presentations, images, videos, letters, researches, white papers, body of an email, etc.
2. **Semi-structured data:** This is the data which does not conform to a data model but has some structure. However, it is not in a form which can be used easily by a computer program; for example, emails, XML, markup languages like HTML, etc. Metadata for this data is available but is not sufficient.
3. **Structured data:** This is the data which is in an organized form (e.g., in rows and columns) and can be easily used by a computer program. Relationships exist between entities of data, such as classes and their objects. Data stored in databases is an example of structured data.

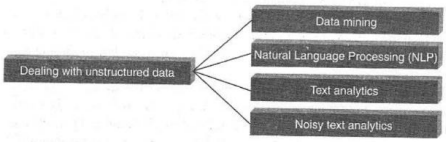


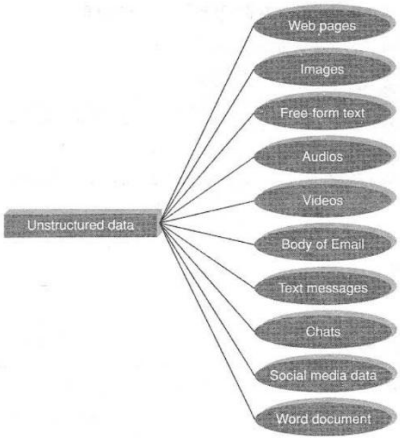


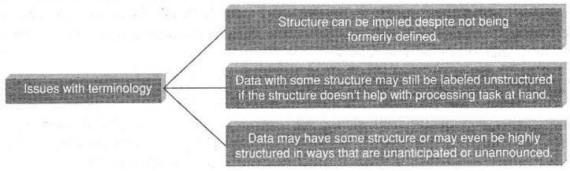












10

(OR)

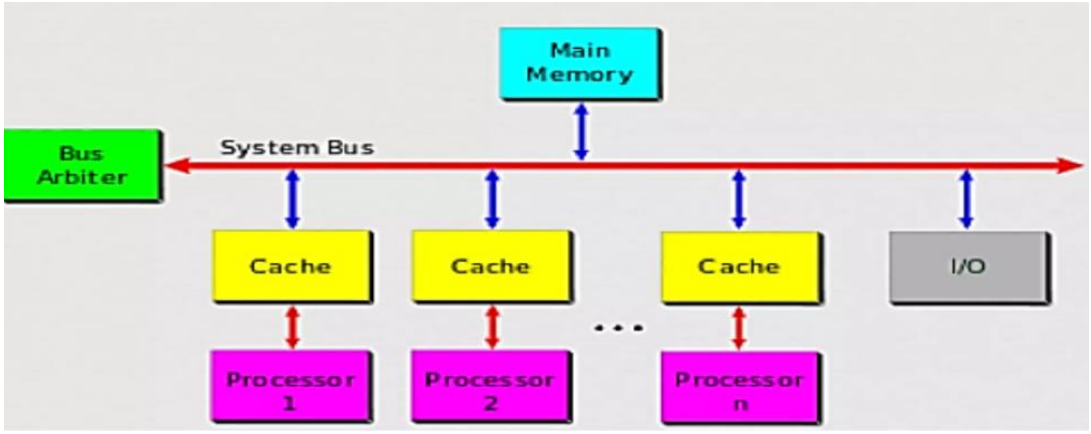
2.	**a. Discuss Big Data Analytics and explain the following terminologies:** **i) Symmetric Multiprocessor System** **ii) CAP Theorem** **Scheme : Definition + Explanation of each – 2+4+4 Marks** **Solution**	**10**	**1**	**L2**

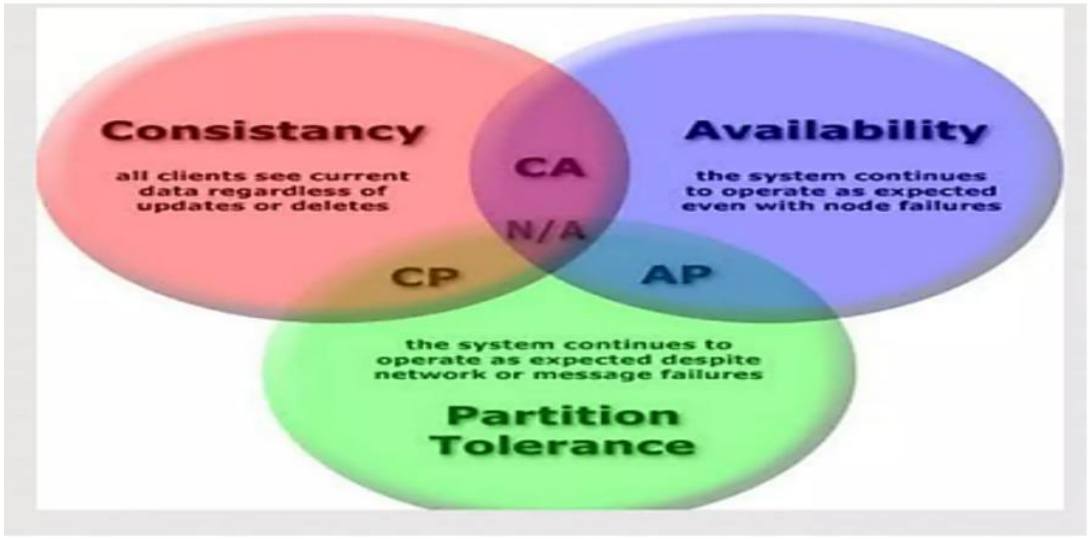
SMP is a single common main memory that is shared by two or more identical processors.

To all I/O devices and are controlled by a single operating system instance.

SMP are tightly coupled multiprocessor systems.

Its own high-speed memory, called cache memory and are connected using a system bus.





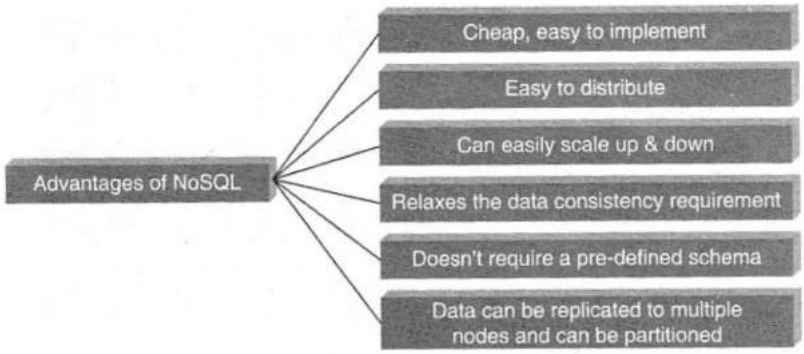
b. Explain the **features and advantages of NoSQL**. Discuss the **types of NoSQL databases**. 10

Scheme : Features and Advantages + Explanation of each data base– 4+6 marks

Solution:

Few features of NoSQL databases are as follows:

1. They are open source.
2. They are non-relational.
3. They are distributed.
4. They are schema-less.
5. They are cluster friendly.
6. They are born out of 21st century web applications.



1. Key–value or the big hash table.
2. Schema-less.

Refer Figure 4.3. Let us take a closer look at key–value and few other types of schema-less databases:

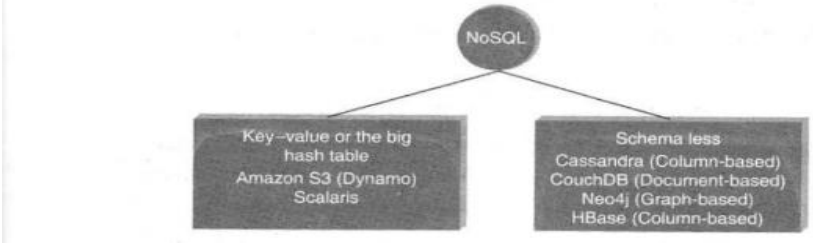
1. **Key–value:** It maintains a big hash table of keys and values. For example, Dynamo, Redis, Riak, etc.
   Sample Key-Value Pair in Key-Value Database

Key	Value
First Name	Simmonds
Last Name	David

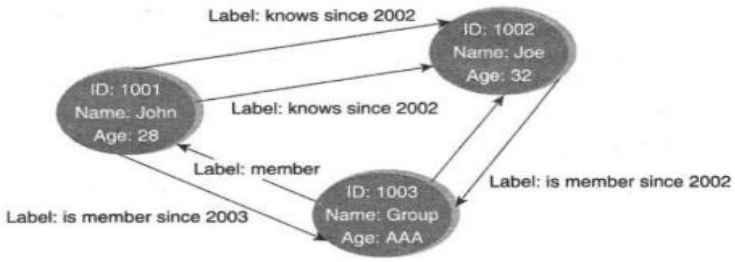
2. **Document:** It maintains data in collections constituted of documents. For example, MongoDB, Apache CouchDB, Couchbase, MarkLogic, etc.
   Sample Document in Document Database

```
{
"Book Name":    "Fundamentals of Business Analytics",
"Publisher":    "Wiley India",
"Year of Publication":    "2011"
}
```

3. **Column:** Each storage block has data from only one column. For example: Cassandra, HBase, etc.

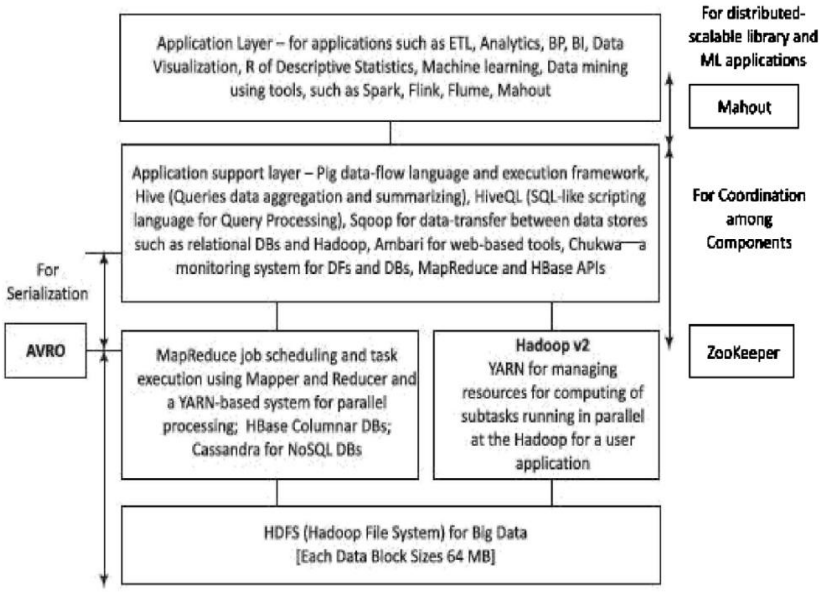


4. **Graph:** They are also called network database. A graph stores data in nodes. For example, Neo4j, HyperGraphDB, etc.
   Sample Graph in Graph Database



MODULE -2

3. **a. Discuss the need for Hadoop and its high-level architecture**
   Scheme: Definition + Explanation + Architecture 2+6+2 marks
   Solution:
   1. **Handling Big Data:** Traditional databases cannot efficiently store and process huge volumes of structured and unstructured data. Hadoop is designed to handle *Big Data* (large volume, high velocity, and variety).
   2. **Distributed Storage:** Hadoop uses **HDFS (Hadoop Distributed File**

	System) to store data across multiple machines, ensuring scalability and fault tolerance. 3. **Parallel Processing:** Hadoop processes data in parallel using the **MapReduce** programming model, which significantly reduces processing time for large datasets. 4. **Cost Effective and Fault Tolerant:** Hadoop runs on low-cost commodity hardware and automatically handles node failures, making it reliable and economical 	**10**	**2**	**L2**
	b. Illustrate the MapReduce process with a word count example. **Scheme : Definition + Explanation + Example – 2+3+5 marks** **Solution:** **MapReduce** is a programming model used by Hadoop to process large datasets in a distributed and parallel manner. It consists of two main phases: **Map** and **Reduce**. **Problem Statement: Word Count** Count the number of occurrences of each word in a given input file. **Input Data (Text File):** `Hadoop is big data` `Big data is Hadoop` **1. Input Splitting** • The input file is divided into smaller chunks called **input splits**. • Each split is processed by a separate **Mapper**. **2. Map Phase** • The **Mapper** reads input data line by line.	**10**	**2**	**L3**

- Each word is treated as a key and assigned a value **1**.

Mapper Output (Key, Value pairs):

```
(Hadoop, 1)
(is, 1)
(big, 1)
(data, 1)
(Big, 1)
(data, 1)
(is, 1)
(Hadoop, 1)
```

3. Shuffling and Sorting

- The MapReduce framework automatically:
  - Groups all identical keys together.
  - Sorts and transfers them to the Reducer.

After Shuffling:

```
(Hadoop, [1,1])
(is, [1,1])
(big, [1])
(Big, [1])
(data, [1,1])
```

4. Reduce Phase

- The **Reducer** takes each key and its list of values.
- It sums the values to produce the final count.

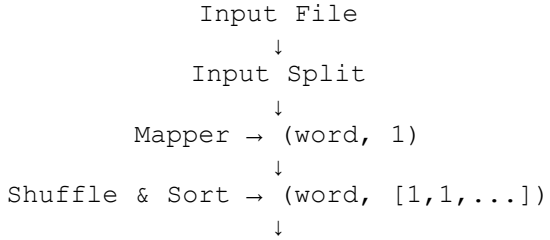
Reducer Output:

```
(Hadoop, 2)
(is, 2)
(big, 1)
(Big, 1)
(data, 2)
```

5. Output

- The final result is stored in **HDFS** as output files.
- Each line contains a word and its total count.

Diagrammatic Representation (Explanation)



```
Reducer → (word, total count)
                ↓
           Output File
```

Advantages of MapReduce

- Processes large data efficiently.
- Parallel and distributed execution.
- Fault tolerant.
- Scalable for big data applications.

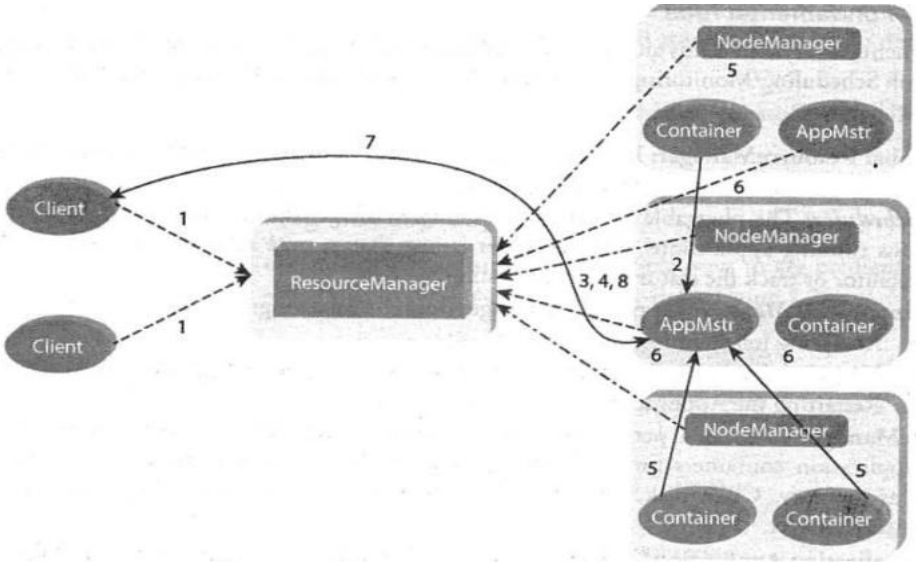
(OR)

4. **a. Discuss the limitations of HDFS and its solution. Explain the YARN architecture.**

Scheme : Limitations + Explanation + YARN architecture – 2+3+5 marks

Solution:

1. **Not suitable for small files:**
   HDFS is inefficient in handling a large number of small files as it overloads the NameNode memory.
   Solution: Use **HDFS Archive (HAR)**, **HBase**, or combine small files using **SequenceFiles**.
2. **High latency access:**
   HDFS is optimized for batch processing and not for real-time or low-latency data access.
   Solution: Use **HBase** or **NoSQL databases** for real-time data access.
3. **Write-once, read-many model:**
   HDFS does not support random write or update of files.
   Solution: Use **HBase** or redesign applications to use append-only operations.
4. **Single point of failure (NameNode):**
   Failure of the NameNode can make the HDFS cluster unavailable.
   Solution: Use **High Availability (HA)** with **Active and Standby NameNodes**.



		10	**2**	**L3**

	b. Implement a MapReduce program in Java / Python / R to implement matrix multiplication. **Scheme: Definition + Program – 2+8 marks** **Solution:**			
	import java.io.IOException; import java.util.*; import java.util.AbstractMap.SimpleEntry; import java.util.Map.Entry; import org.apache.hadoop.fs.Path; import org.apache.hadoop.conf.*; import org.apache.hadoop.io.*; import org.apache.hadoop.mapreduce.*; import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.input.TextInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat; public class MatrixMultiplication { public static class Map extends Mapper<LongWritable, Text, Text, Text> { public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException { String line = value.toString().trim(); // Skip empty or malformed lines if (line.isEmpty()) return; String[] tokens = line.split(","); if (tokens.length != 4) return; String matrixName = tokens[0]; int row = Integer.parseInt(tokens[1]); int col = Integer.parseInt(tokens[2]); float val = Float.parseFloat(tokens[3]); Text outputKey = new Text(); Text outputValue = new Text(); if (matrixName.equals("A")) { // A[i][k], emit for all columns j in result matrix (since B is 2x2) for (int j = 0; j < 2; j++) { outputKey.set(row + "," + j); // key = i,j outputValue.set("A," + col + "," + val); // A,k,value context.write(outputKey, outputValue); } } else if (matrixName.equals("B")) { // B[k][j], emit for all rows i in result matrix (since A is 2x2) for (int i = 0; i < 2; i++) { outputKey.set(i + "," + col); // key = i,j outputValue.set("B," + row + "," + val); // B,k,value context.write(outputKey, outputValue);	**10**	**2**	**L4**

```
            }
        }
    }
}
public static class Reduce extends Reducer<Text, Text, Text, FloatWritable> {
    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {

        List<Entry<Integer, Float>> listA = new ArrayList<>();
        List<Entry<Integer, Float>> listB = new ArrayList<>();

        for (Text val : values) {
            String[] tokens = val.toString().split(",");
            if (tokens.length != 3) continue;

            String matrixName = tokens[0];
            int index = Integer.parseInt(tokens[1]);
            float value = Float.parseFloat(tokens[2]);

            if (matrixName.equals("A")) {
                listA.add(new SimpleEntry<>(index, value));
            } else if (matrixName.equals("B")) {
                listB.add(new SimpleEntry<>(index, value));
            }
        }

        float sum = 0;
        for (Entry<Integer, Float> a : listA) {
            for (Entry<Integer, Float> b : listB) {
                if (a.getKey().equals(b.getKey())) {
                    sum += a.getValue() * b.getValue();
                }
            }
        }

        context.write(key, new FloatWritable(sum));
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "MatrixMultiplication2x2");

    job.setJarByClass(MatrixMultiplication.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(FloatWritable.class);
```

```
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));     // Input path
        FileOutputFormat.setOutputPath(job, new Path(args[1]));   // Output path

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Input :
matrix.txt

```
A,0,0,1
A,0,1,2
A,1,0,3
A,1,1,4
B,0,0,5
B,0,1,6
B,1,0,7
B,1,1,8
```

MODULE -3

5. **a. Discuss replication and sharding in MongoDB.**

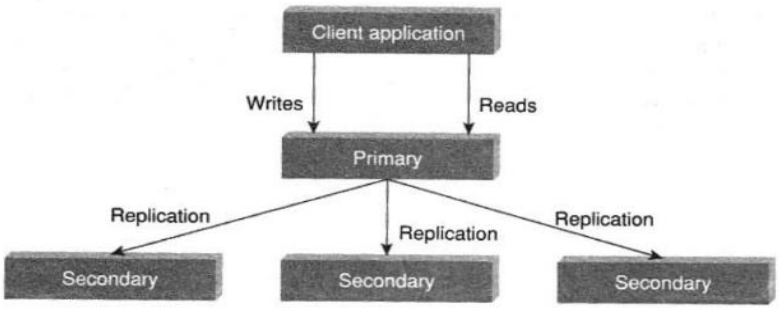
Scheme: Definition +Explanation+ Diagram – 2+6+2 marks

Solution:

MongoDB is

1. Cross-platform.
2. Open source.
3. Non-relational.
4. Distributed.
5. NoSQL.
6. Document-oriented data store.

Why replication? It provides data redundancy and high availability. It helps to recover from hardware failure and service interruptions. In MongoDB, the replica set has a single primary and several secondaries. Each write request from the client is directed to the primary. The primary logs all write requests into its Oplog (operations log). The Oplog is then used by the secondary replica members to synchronize their data. This way there is strict adherence to consistency. Refer Figure 6.3. The clients usually read from the primary. However, the client can also specify a read preference that will then direct the read operations to the secondary.



10 | **3** | **L2**

	b.Illustrate the CRUD operations using MongoDB query language with examples. **Scheme: Definition + CRUD operations + Example – 2+5+3 marks** **Solution:** **CRUD** stands for **Create, Read, Update, and Delete**. MongoDB provides a rich query language to perform these operations on documents stored in collections. # 1. Create Operation (Insert) Used to insert documents into a MongoDB collection. **Insert One Document** ``` db.student.insertOne({ sid: 101, name: "Ravi", dept: "CSE", marks: 85 }) ``` **Insert Multiple Documents** ``` db.student.insertMany([{ sid: 102, name: "Anu", dept: "ECE", marks: 78 }, { sid: 103, name: "Kiran", dept: "ME", marks: 82 }]) ``` # 2. Read Operation (Retrieve / Query) Used to retrieve documents from a collection. **Find All Documents** ``` db.student.find() ``` **Find with Condition** ``` db.student.find({ dept: "CSE" }) ``` **Find Specific Fields** ``` db.student.find({ marks: { $gt: 80 } }, { name: 1, marks: 1, _id: 0 }) ``` # 3. Update Operation Used to modify existing documents.	**10**	**3**	**L3**

Update One Document

```
db.student.updateOne(
  { sid: 101 },
  { $set: { marks: 90 } }
)
```

Update Multiple Documents

```
db.student.updateMany(
  { dept: "ECE" },
  { $set: { dept: "Electronics" } }
)
```

4. Delete Operation

Used to remove documents from a collection.

Delete One Document

```
db.student.deleteOne({ sid: 103 })
```

Delete Multiple Documents

```
db.student.deleteMany({ marks: { $lt: 80 } })
```

Advantages of MongoDB CRUD

- Schema-less and flexible document structure
- Easy-to-use query language
- Supports nested documents and arrays
- High performance and scalability

(OR)

6. **a. Demonstrate the following operations in MongoDB query language with examples:**
   i) Count
   ii) Limit
   iii) Sort
   iv) Skip

10 | **3** | **L2**

Scheme: Explanation of each operation– 2.5+2.5+2.5+2.5 marks
Solution:

MongoDB provides several query modifiers to control and process query results. The following operations are commonly used: **Count, Limit, Sort, and Skip**.

i) Count Operation

The **count()** operation is used to find the number of documents in a collection or matching a condition.

Example: Count all documents

```
db.student.countDocuments()
```

Example: Count documents with a condition

```
db.student.countDocuments({ dept: "CSE" })
```

ii) Limit Operation

The **limit()** operation restricts the number of documents returned by a query.

Example: Display only first 3 documents

```
db.student.find().limit(3)
```

iii) Sort Operation

The **sort()** operation arranges documents in ascending or descending order.

- **1** → Ascending order
- **-1** → Descending order

Example: Sort by marks in ascending order

```
db.student.find().sort({ marks: 1 })
```

Example: Sort by marks in descending order

```
db.student.find().sort({ marks: -1 })
```

iv) Skip Operation

The **skip()** operation is used to skip a specified number of documents in the result set.

Example: Skip first 2 documents

```
db.student.find().skip(2)
```

Combined Example (Limit + Skip + Sort)

Display top 3 students based on marks after skipping the first 2 records:

```
db.student.find()
          .sort({ marks: -1 })
          .skip(2)
          .limit(3)
```

	b. Explain the application of the following in MongoDB: **i) Cursors** **ii) Indexes** **iii) Mongo Export** **iv) Aggregate function** **Scheme : Explanation of each application – 2.5+2.5+2.5+2.5 marks** **Solution:**	**10**	**3**	**L2**

i) Cursors

- A **cursor** is a pointer to the result set returned by a `find()` query.
- It allows MongoDB to return large result sets in batches instead of loading all documents into memory.

Application:

- Used to iterate through query results efficiently.
- Helps in processing large datasets.

Example:

```
var cur = db.student.find()
while (cur.hasNext()) {
   printjson(cur.next())
}
```

ii) Indexes

- **Indexes** improve the speed of query operations by reducing the number of documents scanned.
- MongoDB supports single-field, compound, and unique indexes.

Application:

- Faster search and retrieval of data.
- Efficient execution of sorting and filtering operations.

Example:

```
db.student.createIndex({ sid: 1 })
```

iii) Mongo Export

- **mongoexport** is a command-line tool used to export MongoDB data into external files such as **JSON** or **CSV**.
- Useful for backup, data migration, and data sharing.

Application:

- Export data from MongoDB for reporting and analysis.
- Transfer data to other databases or tools.

Example:

```
mongoexport --db college --collection student --out student.json
```

iv) Aggregate Function

- The **aggregation framework** processes data records and returns computed results.
- It performs operations like grouping, filtering, sorting, and calculating totals.

Application:

- Used for data analysis and summarization.
- Supports operations similar to SQL `GROUP BY`.

Example:

```
db.student.aggregate([
  { $group: { _id: "$dept", avgMarks: { $avg: "$marks" } } }
])
```

MODULE -4

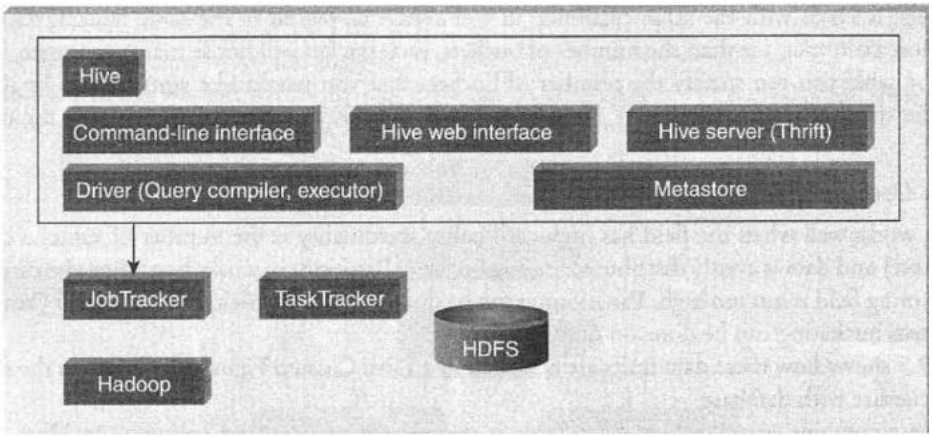
7. **a. Discuss the features of Hive. Explain the Hive architecture.** **10** **3** **L2**

Scheme : Definition + diagram with explanation for each – 2+8 marks

Solution:

Features of Hive

1. It is similar to SQL.
2. HQL is easy to code.
3. Hive supports rich data types such as structs, lists and maps.
4. Hive supports SQL filters, group-by and order-by clauses.
5. Custom Types, Custom Functions can be defined.



1. **Hive Command-Line Interface (Hive CLI):** The most commonly used interface to interact with Hive.
2. **Hive Web Interface:** It is a simple Graphic User Interface to interact with Hive and to execute query.
3. **Hive Server:** This is an optional server. This can be used to submit Hive Jobs from a remote client.

4. **JDBC/ODBC:** Jobs can be submitted from a JDBC Client. One can write a Java code to connect to Hive and submit jobs on it.
5. **Driver:** Hive queries are sent to the driver for compilation, optimization and execution.
6. **Metastore:** Hive table definitions and mappings to the data are stored in a Metastore. A Metastore consists of the following:
   - **Metastore service:** Offers interface to the Hive.
   - **Database:** Stores data definitions, mappings to the data and others.

b. Explain the DDL and DML commands in Hive. | **10** | **3** | **L2**

Scheme : Definition with explanation and example for each – 5+5 marks

Solution:

1. Create/Drop/Alter Database
2. Create/Drop/Truncate Table
3. Alter Table/Partition/Column
4. Create/Drop/Alter View
5. Create/Drop/Alter Index
6. Show
7. Describe

1. Loading files into table.
2. Inserting data into Hive Tables from queries.

DDL Commands in Hive

DDL commands are used to define and manage database objects.

1. **Create Table**

```
CREATE TABLE student (
  sid INT,
  name STRING,
  dept STRING
);
```

2. **Alter Table**

```
ALTER TABLE student ADD COLUMNS (marks INT);
```

3. **Drop Table**

```
DROP TABLE student;
```

DML Commands in Hive

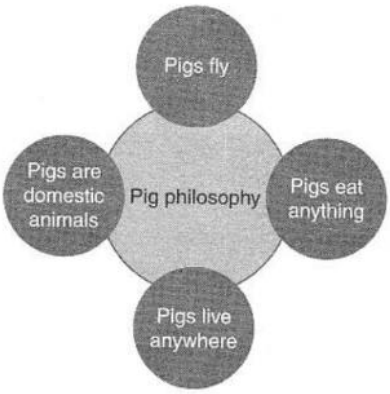
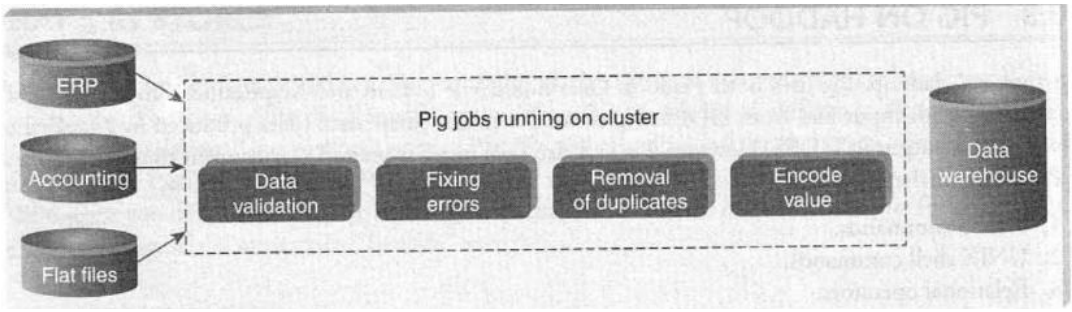
DML commands are used to manipulate data in tables.

1. **Load Data**

```
LOAD DATA INPATH '/user/hive/student.txt'
INTO TABLE student;
```

2. **Insert Data**

```
INSERT INTO TABLE student
VALUES (101, 'Ravi', 'CSE', 85);
```

	(OR)			
8.	**a.** Express the **features and philosophy of Pig**. Discuss **ETL processing**. **Scheme : Definition with explanation of features and philosophy and ETL– 5+5 marks** **Solution:** 1. It provides an **engine** for executing **data flows** (how your data should flow). Pig processes data in parallel on the Hadoop cluster. 2. It provides a language called "**Pig Latin**" to express data flows. 3. Pig Latin contains operators for many of the traditional data operations such as join, filter, sort, etc. 4. It allows users to develop their own functions (User Defined Functions) for reading, processing, and writing data. 1. **Pigs Eat Anything:** Pig can process different kinds of data such as structured and unstructured data. 2. **Pigs Live Anywhere:** Pig not only processes files in HDFS, it also processes files in other sources such as files in the local file system. 3. **Pigs are Domestic Animals:** Pig allows you to develop user-defined functions and the same can be included in the script for complex operations. 4. **Pigs Fly:** Pig processes data quickly.  	**10**	**3**	**L2**
	b. Discuss the following in Pig: **i) Relational operators – Foreach and Limit** **ii) Complex data types – Tuple and Map** **Scheme : Definition with explanation and example for each – 5+5 marks** **Solution:** Apache Pig uses **Pig Latin**, a high-level data flow language, to process large datasets in Hadoop. # i) Relational Operators – FOREACH and LIMIT **FOREACH Operator**	**10**	**3**	**L2**

- The **FOREACH** operator is used to process each tuple (row) of a relation.
- Commonly used with **GENERATE** to project fields or apply expressions.

Applications:

- Selecting specific fields
- Performing transformations on data

Example:

```
student = LOAD 'student.txt' USING PigStorage(',')
          AS (sid:int, name:chararray, marks:int);

result = FOREACH student GENERATE sid, name, marks+5 AS new_marks;
```

LIMIT Operator

- The **LIMIT** operator restricts the number of output tuples.
- Useful for sampling data or testing scripts.

Applications:

- Displaying a subset of data
- Reducing output size

Example:

```
top5 = LIMIT student 5;
```

ii) Complex Data Types – TUPLE and MAP

Tuple

- A **tuple** is an ordered set of fields.
- Represented using parentheses `()`.

Applications:

- Represents a single record
- Can be nested inside relations

Example:

```
A = LOAD 'data.txt' AS (t:(id:int, name:chararray, marks:int));
```

Map

- A **map** is a set of key–value pairs.
- Keys must be **chararray**, and values can be any data type.
- Represented using square brackets `[]`.

Applications:

- Storing semi-structured data
- Handling dynamic attributes

Example:

```
B = LOAD 'info.txt' AS (details:map[]);
```

Accessing map values:

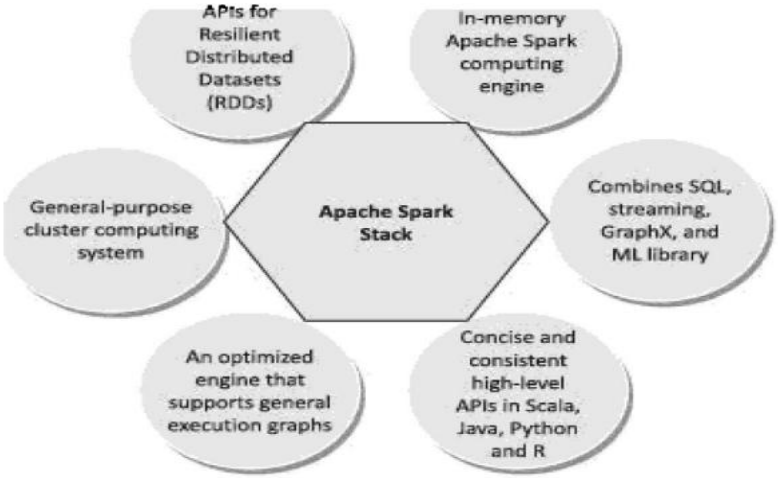
```
C = FOREACH B GENERATE details#'city', details#'state';
```

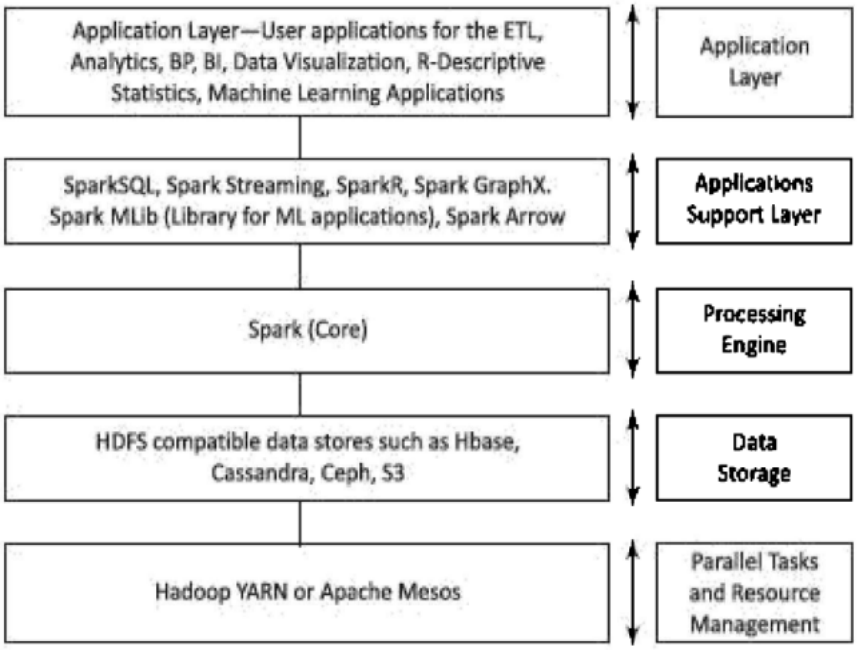
MODULE -5

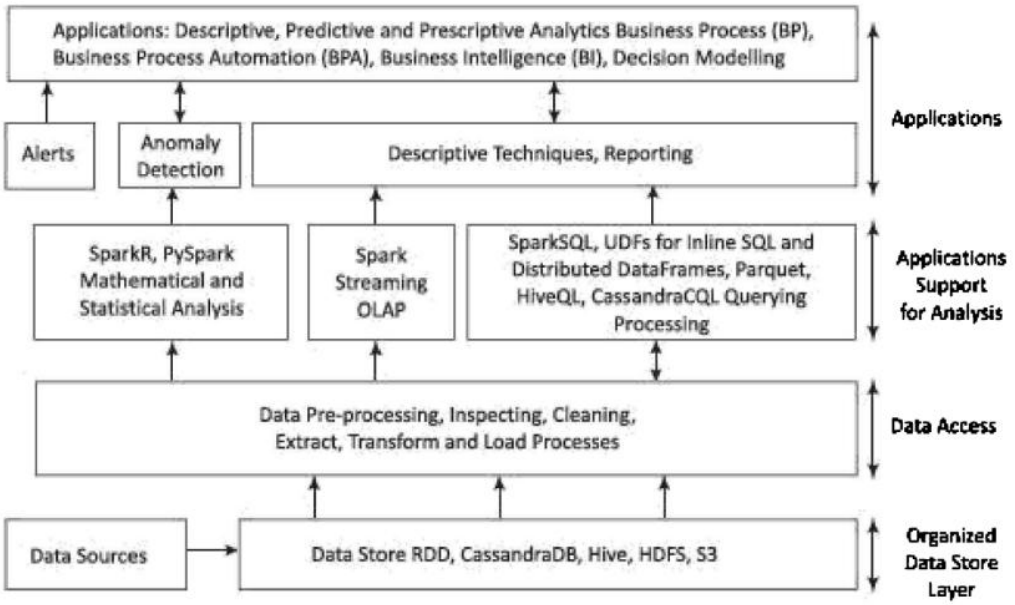
9. **a. Discuss the features of Spark. Explain the Spark software stack.** **10** **4** **L2**

Scheme : Definition with explanation + Spark software stack – 5+5 marks

Solution:





	b.Explain the steps involved between acquisition of data from multiple sources and its application in Spark. **Scheme : Definition + explanation + Steps + Application – 2+3+3+2 marks** **Solution:**  1. Data Storage: Store of data from the multiple sources after acquisition. The Big Data storage may be in HDFS compatible files, Cassandra, Hive, HDFS or S3. 2. Data pre-processing: This step requires: (a) dropping out of range, inconsistent and outlier values, (b) filtering unreliable, irrelevant and redundant information, (c) data cleaning, editing, reduction and/or wrangling, (d) data-validation, transformation or transcoding. 3. Extract, transform and Load (ETL)) for the analysis 4. Mathematical and statistical analysis of the data obtained after querying relevant data needing the analysis, or OLAP, 5. Applications of analyzed data, for example, descriptive, predictive and prescriptive analytics, business processes (BPs), business process automation (BPA), business intelligence (BI), decision modelling and knowledge discovery.	**10**	**4**	**L2**
	(OR)			
10.	**a.** Discuss **text mining** and its applications. Explain the **process of text mining**. **Scheme: Definition with explanation + Process with Diagram – 5+5 marks** **Solution:** 1. "Text mining refers to the process of deriving high-quality information from text." (Wikipedia) 2. "Text mining is the process of discovering and extracting knowledge from unstructured data." (National Center of Text Mining —The University of Manchester[1]) 3. "Text mining is the process of analyzing collections of textual contents in order to capture key concepts themes, uncover hidden relationships, and discover the trends without requiring that you know the precise words or terms that authors have used to express those concepts." (IBM[2]) 4. "Text mining is a technique which helps in revealing the patterns and relationships in large volumes of textual content that are not visible to the naked eye, leading to new business opportunities and improvements in processes." (Amazon BigData Official Blog[3])	**10**	**4**	**L2**

Application of text mining:

Natural Language Processing (NLP) is a technique for analyzing, understanding and deriving meaning from human language. NLP involves the computer's understanding and manipulation of human language. NLP algorithms are typically based on ML algorithms. They automatically learn the rules. First, they analyze set of examples from a large collection of sentences in a book. Then, they make the statistical inferences.

Information Retrieval (IR) is a process of searching and retrieving a subset of documents from the abundant collection of documents. IR can also be defined as extraction of information required by a user. IR is an area derived fundamentally from database technology. One of the most popular applications of IR is searching the information on the web. Search engines provide IR using various advance techniques. For example, the crawler program is capable of retrieving information from a wide variety of data sources. Search methods use metadata or full-text indexing.

Information Extraction (IE) is a process in which the software extracts structured information from unstructured and/or semi-structured documents. IE finds the relationship within text or desired contents from text. IE ideally derives from machine learning, more specifically from the NLP domain. Content extraction from the images, audio or video is an example of information extraction.

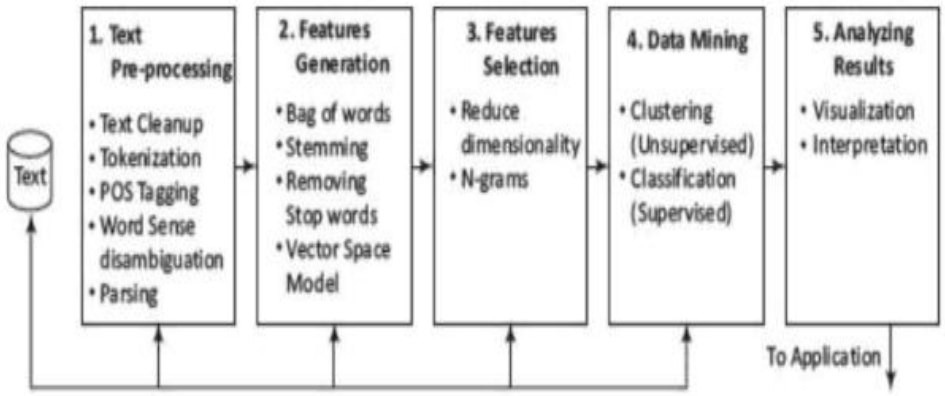
IE requires a dictionary of extraction patterns (For example, "Citizen of <x>, or "Located in <x>") and a semantic lexicon (dictionary of words with semantic category labels).

Document Clustering is an application which groups text documents into clusters. Automating document organization, topic extraction and fast information retrieval or filtering use the document clustering method. For example, web document clustering facilitates easy search by users.

Document Classification is an application to classify text documents into classes or categories. The application is useful for publishers, news sites, blogs or areas where lot of contents are present.

Web Mining is an application of data mining techniques. They discover patterns from the web Data Store. The patterns facilitate understanding. They improve the services of web-based applications. Data mining of web usage provides the browsing behavior of a website.

Concept Extraction is an application that deals with the extraction of concept from textual data. Concept extraction is an area of text classification in which words and phrases are classified into a semantically similar group.



b. Implement a **word count program** in **Hadoop and Spark** using **Java / Python / R**.

10 | **4** | **L4**

Scheme : Definition + Program – 2+8 marks

Solution:

Hadoop Program:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
```

```
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

   // Mapper Class
   public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
         StringTokenizer itr = new StringTokenizer(value.toString());
         while (itr.hasMoreTokens()) {
            word.set(itr.nextToken().replaceAll("[^a-zA-Z0-9]", "").toLowerCase());
            if (!word.toString().isEmpty()) {
               context.write(word, one);
            }
         }
      }
   }

   // Reducer Class
   public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
      private IntWritable result = new IntWritable();

      public void reduce(Text key, Iterable<IntWritable> values,
                         Context context
                         ) throws IOException, InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         result.set(sum);
         context.write(key, result);
      }
   }

   // Driver Code
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");

      job.setJarByClass(WordCount.class);
      job.setMapperClass(TokenizerMapper.class);
      job.setCombinerClass(IntSumReducer.class); // Optional combiner
```

```
        job.setReducerClass(IntSumReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));  // HDFS input
        FileOutputFormat.setOutputPath(job, new Path(args[1])); // HDFS output

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

word.txt (Input)

Hello world
Hello Hadoop
Hadoop is great
hello all
hadoop ecosystem
hi big data

Spark Program:

```
val textFile = sc.textFile("file:///home/bdalab/input.txt")
val counts = textFile.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_ + _)
counts.collect().foreach(println)
```

SAMPLE INPUT:

Spark is amazing
It makes data processing fast

Output:

(Spark,2)
(makes,1)
(data,1)
(processing,1)
(fast.,1)
(is,1)
(amazing,1)

Faculty

Prof. Arvind R
Assistant Professor
Department of ISE