Practical 1

Aim: To study real-world applications and use cases of Big Data in different industries.

Code / Procedure:

```
1. Healthcare: Analyze patient records, predict diseases, improve treatment.
2. Banking: Fraud detection, risk analysis, personalized offers.
3. E-commerce: Product recommendations, customer behavior analysis.
4. Transportation: Route optimization, traffic prediction.
5. Social Media: Sentiment analysis, targeted advertising.
```

Practical 2

Aim: To understand and use basic Hadoop Distributed File System (HDFS) commands.

Code / Procedure:

```
$ hadoop fs -mkdir /user/hadoop/input
$ hadoop fs -put localfile.txt /user/hadoop/input
$ hadoop fs -ls /user/hadoop/input
$ hadoop fs -cat /user/hadoop/input/localfile.txt
$ hadoop fs -rm /user/hadoop/input/localfile.txt
```

Practical 3

Aim: To understand the programming architecture of Hadoop using MapReduce API.

Code / Procedure:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
  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, InterruptedExceptio
      String[] tokens = value.toString().split(" ");
      for (String t : tokens) {
        word.set(t);
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, I
      int sum = 0;
      for (IntWritable val : values) sum += val.get();
      context.write(key, new IntWritable(sum));
    }
  }
  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);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Practical 4

Aim: To implement a word count application using Mapper and Reducer on a single node cluster.

Code / Procedure:

```
1. Write the MapReduce program (WordCount.java).
2. Compile: hadoop com.sun.tools.javac.Main WordCount.java
3. Create JAR: jar cf wc.jar WordCount*.class
4. Run: hadoop jar wc.jar WordCount /input /output
5. View result: hadoop fs -cat /output/part-r-00000
```

Practical 5

Aim: To analyze the weather dataset using Mapper Reducer on a single node cluster.

Code / Procedure:

```
1. Download weather dataset.
2. Mapper extracts year and temperature.
3. Reducer finds maximum temperature per year.
4. Run job: hadoop jar weather.jar Weather /input /output
5. Display results using: hadoop fs -cat /output/*
```

Practical 6

Aim: To set up a MongoDB database and perform basic CRUD operations.

Code / Procedure:

```
> use studentdb
> db.createCollection("students")
> db.students.insert({name: "Rahul", age: 22, course: "BCA"})
> db.students.find()
> db.students.update({name: "Rahul"}, {$set: {age: 23}})
> db.students.remove({name: "Rahul"})
```

Practical 7

Aim: To prepare a case study on Big Data Streaming.

Code / Procedure:

```
1. Study Apache Kafka, Flink, and Spark Streaming.
2. Use cases: Twitter analytics, stock price monitoring.
3. Kafka for ingestion, Spark Streaming for processing.
4. Visualize results using dashboards (Grafana).
```

Practical 8

Aim: To perform basic Pig operations and commands.

Code / Procedure:

```
grunt> A = LOAD 'data.txt' USING PigStorage(',') AS (id:int, name:chararray, age:int);
grunt> DUMP A;
grunt> B = FILTER A BY age > 25;
grunt> DUMP B;
grunt> C = GROUP A BY age;
grunt> DUMP C;
```

Practical 9

Aim: To perform daily show analysis using Pig.

Code / Procedure:

```
grunt> shows = LOAD 'shows.csv' USING PigStorage(',') AS (channel:chararray, viewers:int);
grunt> grouped = GROUP shows BY channel;
grunt> result = FOREACH grouped GENERATE group, SUM(shows.viewers);
grunt> DUMP result;
```

Practical 10

Aim: To implement a word count application using Pig.

Code / Procedure:

```
grunt> A = LOAD 'input.txt' USING PigStorage(' ') AS (word:chararray);
grunt> B = GROUP A BY word;
grunt> C = FOREACH B GENERATE group, COUNT(A);
grunt> DUMP C;
```

Practical 11

Aim: To perform basic queries to retrieve and analyze information using Hive.

Code / Procedure:

```
hive> CREATE TABLE students (id INT, name STRING, marks INT) ROW FORMAT DELIMITED FIELDS TERMINATED
hive> LOAD DATA LOCAL INPATH 'students.csv' INTO TABLE students;
hive> SELECT * FROM students;
hive> SELECT AVG(marks) FROM students;
hive> SELECT name FROM students WHERE marks > 70;
```

Practical 12

Aim: To create HDFS tables, load them in Hive, and perform joining of tables.

Code / Procedure:

```
hive> CREATE TABLE dept (id INT, name STRING);
hive> CREATE TABLE emp (id INT, name STRING, dept_id INT);
hive> LOAD DATA LOCAL INPATH 'dept.csv' INTO TABLE dept;
hive> LOAD DATA LOCAL INPATH 'emp.csv' INTO TABLE emp;
hive> SELECT e.name, d.name FROM emp e JOIN dept d ON (e.dept_id = d.id);
```