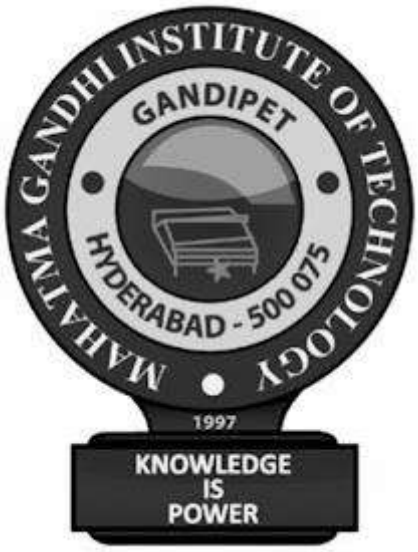
LABORATORY MANUAL

BIG DATA ANALYTICS

III B.Tech – II Semester [Branch: CSE (DS)]



Prepared by

DEPARTMENT OF EMERGING TECHNOLOGIES

Mahatma Gandhi Institute Of Technology

Gandipet, Hyderabad – 500 075.

Course Objectives

- The purpose of this course is to provide the students with the knowledge of Big data Analytics principles and techniques.
- This course is also designed to give an exposure of the frontiers of Big data Analytics

Course Outcomes

Students would be able to

- Use Excel as an Analytical tool and visualization tool.
- Ability to program using HADOOP and Map reduce.
- Ability to perform data analytics using ML in R.
- Use cassandra to perform social media analytics.

List of Experiments:

1. Implement a simple map-reduce job that builds an inverted index on the set of input documents (Hadoop)
2. Process big data in HBase
3. Store and retrieve data in Pig
4. Perform Social media analysis using cassandra
5. Buyer event analytics using Cassandra on suitable product sales data.
6. Using Power Pivot (Excel) Perform the following on any dataset

a) Big Data Analytics

b) Big Data Charting

7. Use R-Project to carry out statistical analysis of big data
8. Use R-Project for data visualization of social media data

1.Implement a simple map-reduce job that builds an inverted index on the set of input Docs(Hadoop)

```
// Mapper class
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class InvertedIndexMapper extends Mapper<LongWritable, Text, Text, Text> {

   private Text word = new Text();
   private Text docId = new Text();

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      String[] parts = line.split("\t"); // Assuming tab-separated input

      if (parts.length >= 2) {
         String DocId = parts[0];
         docId.set(DocId);

         String[] words = parts[1].split(" "); // Assuming words are separated by space
         for (String w : words) {
            word.set(w);
            context.write(word, docId);
         }
      }
   }
}

// Reducer class
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {

   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
      StringBuilder docList = new StringBuilder();

      for (Text docId : values) {
```

```
            if (docList.length() > 0) {
                docList.append(", ");
            }
            docList.append(docId.toString());
        }

        context.write(key, new Text(docList.toString()));
    }
}


// Main class
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 InvertedIndex {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: InvertedIndex <input path> <output path>");
            System.exit(-1);
        }

        Job job = new Job();
        job.setJarByClass(InvertedIndex.class);
        job.setJobName("Inverted Index");

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

        job.setMapperClass(InvertedIndexMapper.class);
        job.setReducerClass(InvertedIndexReducer.class);

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

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

Compile your code and create a jar file. Then, you can run your map-reduce job using Hadoop with the following command:

hadoop jar InvertedIndex.jar InvertedIndex <input-path> <output-path>

Doc1.txt
hello world
hello hadoop

Doc2.txt
hadoop is a framework
hello mapreduce
Output:
framework Doc2.txt
hello Doc1.txt,Doc2.txt
hadoop Doc1.txt,Doc2.txt
is Doc2.txt
mapreduce Doc2.txt
world Doc1.txt

2.Process big data in HBase

A list of HBase commands are given below.

- **Create:** Creates a new table identified by 'table1' and Column Family identified by 'colf'.
- **Put:** Inserts a new record into the table with row identified by 'row..'
- **Scan:** returns the data stored in table
- **Get:** Returns the records matching the row identifier provided in the table
- **Help:** Get a list of commands

Example 1

1. create 'table1', 'colf' list 'table1'

   put 'table1', 'row1', 'colf:a', 'value1'

   put 'table1', 'row1', 'colf:b', 'value2'

   put 'table1', 'row2', 'colf:a', 'value3'

   scan 'table1'

   get 'table1', 'row1'

Example 2

1. **Set up HBase**: Ensure HBase is installed and running on top of HDFS (Hadoop Distributed File System).
2. **Create a Table**: Use the HBase shell to create a table with the required column families.

```
create 'myTable', 'myColumnFamily'
```

3. **Insert Data**: Put data into the table you've created.

```
put 'myTable', 'row1', 'myColumnFamily:column1', 'value1'

put 'myTable', 'row2', 'myColumnFamily:column2', 'value2'
```

4. **Retrieve Data**: Fetch data from the table using the `get` command.

```
get 'myTable', 'row1'
```

5. **Scan Data**: Use the `scan` command to retrieve multiple rows of data.

```
scan 'myTable'
```

6. **Delete Data**: Remove data from the table if needed.

```
delete 'myTable', 'row1', 'myColumnFamily:column1'
```

7. **Close the Table**: Once done with the operations, close the table to release resources.

```
disable 'myTable'

drop 'myTable'
```

3. Store and retrieve data in Pig

Example

student_data.txt in the HDFS directory named /data/ with the following content.

001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata

003,Rajesh,Khanna,9848022339,Delhi

004,Preethi,Agarwal,9848022330,Pune

005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

006,Archana,Mishra,9848022335,Chennai.

We can load the data using the PigStorage function as shown below.

```
grunt> student = LOAD 'hdfs://localhost:9000/pig_data/student_data.txt' USING PigStorage(',') as ( id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray );
```

In the above example, we have seen that we have used comma (‘,’) delimiter. In the same way, we can use the PigStorage() function to store the data into HDFS directory as shown below.

```
grunt> STORE student INTO ' hdfs://localhost:9000/pig_Output/ ' USING PigStorage (',');
```

This will store the data into the given directory.

Output

You can verify the stored data as shown below. First of all, list out the files in the directory named pig_output using ls command as shown below.

$ hdfs dfs -ls 'hdfs://localhost:9000/pig_Output/'

Found 2 items

rw-r--r- 1 Hadoop supergroup 0 2020-10-05 13:03
hdfs://localhost:9000/pig_Output/_SUCCESS

rw-r--r- 1 Hadoop supergroup 224 2020-10-05 13:03
hdfs://localhost:9000/pig_Output/part-m-00000

$ hdfs dfs -cat 'hdfs://localhost:9000/pig_Output/part-m-00000'

1,Rajiv,Reddy,9848022337,Hyderabad

2,siddarth,Battacharya,9848022338,Kolkata

3,Rajesh,Khanna,9848022339,Delhi

4,Preethi,Agarwal,9848022330,Pune

5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar

6,Archana,Mishra,9848022335,Chennai

4. Perform Social media analysis using cassandra

Create a table for storing user posts

```
CREATE TABLE social_media.posts (
    post_id uuid PRIMARY KEY,
    user_id uuid,
    post_text text,
    post_time timestamp,
    likes int,
    shares int
);
```

Insert a post into the table

```
INSERT INTO social_media.posts (post_id, user_id,
Post_text,
 Post_time,
 Likes,
 shares)
VALUES (uuid(), uuid(), 'Excited to be learning about Cassandra!', toTimestamp(now()), 0, 0);
```

To find posts with more than 100 likes

```
SELECT * FROM social_media.posts WHERE likes > 100;
```

5. Buyer event analytics using Cassandra on suitable product sales data.

Create a table for storing product sales events

```
CREATE TABLE sales.product_events (
    event_id uuid PRIMARY KEY,
    product_id uuid,
    buyer_id uuid,
    event_time timestamp,
    event_type text,
    quantity int,
    price decimal
);
```

Insert a sales event into the table

```
INSERT INTO sales.product_events (event_id,
product_id,
buyer_id,
Event_time,
 event_type,
quantity, price)
VALUES (uuid(), uuid(), uuid(), toTimestamp(now()), 'purchase', 1, 19.99);
```

Output:

```
SELECT * FROM sales.product_events
WHERE product_id = <specific_product_id>
AND event_type = 'purchase';
```

7.Use R-Project to carry out statistical analysis of big data

Sample Data for `big_data.csv`

```
Product_ID,Product_Category,Sales_Amount,Date
1,Electronics,150,2022-03-15
2,Clothing,80,2022-07-22
3,Books,120,2022-05-10
4,Home Decor,90,2022-08-05
5,Electronics,200,2022-01-28
6,Clothing,50,2022-11-14
7,Books,110,2022-09-19
8,Home Decor,70,2022-04-03
9,Electronics,180,2022-06-30
10,Clothing,70,2022-10-17
```

#First, you'll need to install the required packages if you haven't already:

```
install.packages("dplyr")
install.packages("ggplot2")
```

#Then, you can use the following syntax to load, manipulate, and analyze the data:

```
# Load required libraries
library(dplyr)
library(ggplot2)

# Read the big data from CSV
big_data <- read.csv("big_data.csv")
# View the structure of the dataset
str(big_data)

# Summary statistics
summary(big_data)
# Perform some data manipulation (e.g., filtering, grouping, summarizing)

# Example: Calculate total sales by product category
total_sales <- big_data %>%
  group_by(Product_Category) %>%
  summarise(Total_Sales = sum(Sales))

# View the resulting data frame
total_sales

# Visualize the total sales by product category using a bar plot
```

```
ggplot(total_sales, aes(x = Product_Category, y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Sales by Product Category",
          x = "Product Category",
          y = "Total Sales") +
  theme_minimal()
```

Output:

```
Product_Category Total_Sales
1        Books         230
2   Clothing         200
3 Electronics         530
4 Home Decor         160
```

8. Use R-Project for data visualization of social media data

Social_media_data.csv

```
Text,Hashtags

"Excited to announce the launch of our new product! #newproduct
#launch #excited",#newproduct #launch #excited

"Check out our latest blog post about sustainability! #sustainability
#environment #blog",#sustainability #environment #blog

"We're hosting a webinar next week on digital marketing strategies.
Join us! #webinar #digitalmarketing",#webinar #digitalmarketing

"Happy Friday everyone! Have a great weekend! #friday #weekend
#happy",#friday #weekend #happy

"Throwback to our team outing last summer. #throwbackthursday
#teambuilding",#throwbackthursday #teambuilding

"Excited to attend the conference next month! #conference
#excited",#conference #excited
```

```
# Load required library

library(ggplot2)



# Assuming you have a dataset named social_media_data.csv with columns
including "Text" and "Hashtags"

# Read the data

social_media_data <- read.csv("social_media_data.csv")



# Assuming "Hashtags" column contains the hashtags used in the posts

# Filter data for posts containing a particular hashtag
```

```
particular_hashtag <- "your_hashtag"

hashtag_data <- subset(social_media_data, grepl(paste0("\\b",
particular_hashtag, "\\b"), Hashtags, ignore.case = TRUE))



# Count the frequency of the particular hashtag

hashtag_frequency <- nrow(hashtag_data)



# Visualize the frequency using a bar plot

ggplot() +

  geom_bar(data = NULL, aes(x = "", y = hashtag_frequency), fill =
"skyblue", stat = "identity") +

  geom_text(aes(x = "", y = hashtag_frequency, label =
hashtag_frequency), vjust = -0.5) +

  labs(title = paste("Frequency of Hashtag", particular_hashtag),

      x = NULL, y = "Frequency") +

  theme_minimal()
```

Output: