HADOOP

Steps to run

1. Create a New File named Bash.sh
3. Execute the bash.sh File using following command **source Bash.sh.**
4. Verify **JAVA_HOME**
echo $JAVA_HOME
Echo $PATH
5. **hadoop**
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files **javac -d . *.java**
9. **echo Main-Class: oddeven.driver > Manifest.txt**
10. Create an executable jar file: **jar cfm oddeven.jar Manifest.txt oddeven/*.class**
11. oe.txt is input file for Oddeven create Input File
echo 1 2 3 4 5 6 7 8 9 10 > input.txt
12. Run the jar file
hadoop jar oddeven.jar input.txt output
13. To see the Output
cat output/*

1. Write a MapReduce program to analyze the given natural numbers and generate statistics for the number as **Odd or Even** and print their sum.

driver.java

```
package oddeven;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
```

```
   public static void main(String args[]) throws IOException
   {
         JobConf conf=new JobConf(driver.class);
         conf.setMapperClass(mapper.class);
         conf.setReducerClass(reducer.class);
         conf.setOutputKeyClass(Text.class);
         conf.setOutputValueClass(IntWritable.class);
         FileInputFormat.addInputPath(conf, new Path(args[0]));
         FileOutputFormat.setOutputPath(conf,new Path(args[1]));
         JobClient.runJob(conf);
   }
}
```

mapper.java

```
package oddeven;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text ,
IntWritable>
{
   public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable>
output,Reporter r) throws IOException
   {
         String[] line=value.toString().split(" ");
         for(String num:line){
               int number=Integer.parseInt(num);
               if(number%2==0) {
                     output.collect(new Text("even"),new IntWritable(number));
               }
               else{
                     output.collect(new Text("odd"),new IntWritable(number));
               }
         }
   }
}
```

reducer.java

```
package oddeven;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
public class reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable>
{
   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable>
output ,Reporter r) throws IOException
   {
          int sum=0,count=0;
          while(value.hasNext()){
                sum+=value.next().get();
                count++;
          }
          output.collect(new Text("Sum of "+key+" Numbers"),new IntWritable(sum));
          output.collect(new Text(key+" Number count"),new IntWritable(count));
   }
}
```

oe.txt

1 2 3 4 5 6 7 8 9 10

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"}
{print
$1}')
export PATH=$(echo $PATH):$(pwd)/bin
export CLASSPATH=$(hadoop classpath)
```

3. Execute the bash.sh File using following command **source Bash.sh.**
4. Verify **JAVA_HOME** variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc File.

```
echo $JAVA_HOME
Echo $PATH
```

5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)
javac -d . *.java
9. Set driver class in manifest
echo Main-Class: oddeven.driver > Manifest.txt
10. Create an executable jar file
jar cfm oddeven.jar Manifest.txt oddeven/*.class
11. oe.txt is input file for Oddeven create Input File
echo 1 2 3 4 5 6 7 8 9 10 > oe.txt
12. Run the jar file
hadoop jar oddeven.jar oe.txt output
13. To see the Output
cat output/*

2. Write a MapReduce program to analyze the given **Weather Report** Data and to generate a report with cities having maximum and minimum temperature for a particular year.

driver.java

```
package weather;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
   public static void main(String args[]) throws IOException
   {
         JobConf conf=new JobConf(driver.class);
         conf.setMapperClass(mapper.class);
         conf.setReducerClass(reducer.class);
```

```
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(DoubleWritable.class);
        FileInputFormat.addInputPath(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

mapper.java

```
package weather;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class mapper extends MapReduceBase implements Mapper<LongWritable,
Text,Text,DoubleWritable>{
    public void map(LongWritable key , Text value , OutputCollector<Text,DoubleWritable>
output, Reporter r) throws IOException
    {
        String line=value.toString();
        String year=line.substring(15,19);
        Double temp=Double.parseDouble(line.substring(87,92));
        output.collect(new Text(year), new DoubleWritable(temp));
    }
}
```

reducer.java

```
package weather;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {
    public void reduce(Text key, Iterator<DoubleWritable> value,
OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException{
        Double max=-9999.0;
```

```
        Double min=9999.0;
        while(value.hasNext()){
                Double temp=value.next().get();
                max=Math.max(max,temp);
                min=Math.min(min,temp);
        }
        output.collect(new Text("Max temp at "+ key), new DoubleWritable(max));
        output.collect(new Text("Min temp at "+ key), new DoubleWritable(min));
   }
}
```

Input.txt

```
0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001
CN9999999N9+00001+99999999999
0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001
CN9999999N9+00221+99999999999
0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001
CN9999999N9-00111+99999999999
0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001
CN0500001N9+01111+99999999999
0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001
CN0500001N9+00781+99999999999
```

3. Write a MapReduce program to analyze the given **Earthquake Data** and generate statistics with region and magnitude/ region and depth/ region and latitude/ region and longitude.

driver.java

```
package earthquake;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
```

```
{
   public static void main(String args[]) throws IOException
   {
         JobConf conf=new JobConf(driver.class);
         conf.setMapperClass(mapper.class);
         conf.setReducerClass(reducer.class);
         conf.setOutputKeyClass(Text.class);
         conf.setOutputValueClass(DoubleWritable.class);
         FileInputFormat.addInputPath(conf, new Path(args[0]));
         FileOutputFormat.setOutputPath(conf,new Path(args[1]));
         JobClient.runJob(conf);
   }
```

mapper.java

```
package earthquake;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
public class mapper extends MapReduceBase implements Mapper<LongWritable,
Text,Text,DoubleWritable>
{
   public void map(LongWritable key , Text value , OutputCollector<Text,DoubleWritable>
output, Reporter r) throws IOException
   {
         String[] line=value.toString().split(",");
         Double longi=Double.parseDouble(line[7]);
         output.collect(new Text(line[11]), new DoubleWritable(longi));
   }
}
```

reducer.java

```
package earthquake;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
```

```
class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {

   public void reduce(Text key, Iterator<DoubleWritable> value,
OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException
   {
         Double max=-9999.0;
         while(value.hasNext())
         {
               Double temp=value.next().get();
               max=Math.max(max,temp);
         }
         output.collect(new Text(key), new DoubleWritable(max));
   }

}
```

4. Write a MapReduce program to analyze the given **Insurance Data** and generate a statistics report with the construction building name and the count of building/county name and its frequency**.**

driver.java

```
package insurance;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
   public static void main(String args[]) throws IOException
   {
         JobConf conf=new JobConf(driver.class);
         conf.setMapperClass(mapper.class);
         conf.setReducerClass(reducer.class);
         conf.setOutputKeyClass(Text.class);
```

```
        conf.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

mapper.java

```
package insurance;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text ,
IntWritable>
{
   public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable>
output,Reporter r) throws IOException
   {
        String[] line=value.toString().split(",");
        output.collect(new Text(line[2]),new IntWritable(1));
   }
}
```

reducer.java

```
package insurance;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
public class reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable>
{
   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable>
output ,Reporter r) throws IOException
   {
        int sum=0;
```

```
        while(value.hasNext())
        {
                sum+=value.next().get();
        }
        output.collect(key,new IntWritable(sum));
  }
}
```

5. Write a MapReduce program using Java, to analyze the given **Sales Records** over a period of time and generate data about the country's total sales, and the total number of the products. Country's total sales and the frequency of the payment mode.

driver.java

```
package sales;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
   public static void main(String args[]) throws IOException
   {
        JobConf conf=new JobConf(driver.class);
        conf.setMapperClass(mapper.class);
        conf.setReducerClass(reducer.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
   }
}
```

mapper.java

```
package sales;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text ,
IntWritable>
{
   public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable>
output,Reporter r) throws IOException
   {
         String[] line=value.toString().split(",");
         int price=Integer.parseInt(line[2]);
         String cardtype=line[3];
         String Country=line[7];
         output.collect(new Text("Country "+Country),new IntWritable(price));
         output.collect(new Text("CardType "+cardtype),new IntWritable(1));
   }
}
```

reducer.java

```
package sales;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable>
{
   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable>
output ,Reporter r) throws IOException
   {
```

```
        int sum=0;
        while(value.hasNext())
        {
                sum+=value.next().get();
        }
        output.collect(new Text(key),new IntWritable(sum));
    }
}
```

6. Write a MapReduce program using Java, to analyze the given employee record data and generate a statistics report with the total number of **Female and Male Employees** and their average salary.

driver.java

```
package employee;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
   public static void main(String args[]) throws IOException
   {
        JobConf conf=new JobConf(driver.class);
        conf.setMapperClass(mapper.class);
        conf.setReducerClass(reducer.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(DoubleWritable.class);
        FileInputFormat.addInputPath(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
   }
}
```

mapper.java

```
package employee;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text ,
DoubleWritable> {

   public void map(LongWritable key, Text value, OutputCollector<Text,DoubleWritable>
output ,Reporter r) throws IOException
   {
         String[] line=value.toString().split("\\t");
               salary=Double.parseDouble(line[8]);
         output.collect(new Text(line[3]), new DoubleWritable(salary));
   }

}
```

reducer.java

```
package employee;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {
public void reduce(Text key,Iterator<DoubleWritable> value ,
OutputCollector<Text,DoubleWritable> output ,Reporter r) throws IOException
   {
         int count=0;
         Double sum=0.0;
         while(value.hasNext()){
               sum+=value.next().get();
               count+=1;
         }
         output.collect(new Text(key+" Average"), new DoubleWritable(sum/count));
         output.collect(new Text(key+" Count"), new DoubleWritable(count));
   }
}
```

SPARK

1. Create a new Bash file
2. Copy the bash code into the Bash file
3. Run the Bash file **source bash.sh**
4. Verify the path variables by running the commands for Java Home and Path

   echo $JAVA_HOME

   Echo $PATH

5. Verify installation by executing **spark-shell**

Execution :

1. Create the program files with .py extension.
2. Execute the python files with **spark-submit <filename.py> <input file> output**
3. To display the output:

Check installation : spark-shell

Output : spark-submit prog.py data.txt output

1.Write a spark to analyze the given weather report data and to generate a report with cities having maximum temperature for a particular year

```
import sys
if(len(sys.argv)!=3):
        print("Provide Input File and Output Directory")
        sys.exit(0)
from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])
temp=f.map(lambda x: (int(x[15:19]),int(x[87:92])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

2.Write a spark program to analyze the given Earthquake data and generate statistics with region and magnitude

```
import sys
if(len(sys.argv)!=3):
        print("Provide Input File and Output Directory")
        sys.exit(0)
from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])
temp=f.map(lambda x: (x.split(',')[11],float(x.split(',')[8])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

3.Write a spark program to analyze the given Insurance data and generate a statistics report with the construction building name and the count of building.

```
        import sys
        if(len(sys.argv)!=3):
                print("Provide Input File and Output Directory")
                sys.exit(0)
        from pyspark import SparkContext
        sc =SparkContext()
        f = sc.textFile(sys.argv[1])
        temp=f.map(lambda x: (x.split(',')[16],1))
        data=temp.countByKey()
        dd=sc.parallelize(data.items())
        dd.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Insurance data and generate a statistics report with the county name and its frequency.

```
import sys
if(len(sys.argv)!=3):
        print("Provide Input File and Output Directory")
        sys.exit(0)
```

```
from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])
temp=f.map(lambda x: (x.split(',')[2],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

4. Write a map-reduce program to analyze the given sales records over a period and generate data about the country's total sales, and the total number of the products

```
import sys
if(len(sys.argv)!=3):
        print("Provide Input File and Output Directory")
        sys.exit(0)
from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])
temp=f.map(lambda x: (x.split(',')[7],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

.Write a map-reduce program to analyze the given sales records over a period of time and generate data about the country's total sales and the frequency of the payment mode.

```
import sys
if(len(sys.argv)!=3):
        print("Provide Input File and Output Directory")
        sys.exit(0)
from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])
temp=f.map(lambda x: (x.split(',')[3],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
```

dd.saveAsTextFile(sys.argv[2])

PIG

1.Filter

Create a new file with .pig extension

student_details = LOAD '/home/msrit/Downloads/pig/test/Filter/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

filter_data = FILTER student_details BY city == 'Chennai';

Dump filter_data;

Execution: pig <name.pig>

Student_details.txt
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai

2. **GROUPING**

```
student = LOAD 'student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, age:int, phone:chararray, city:chararray);

group_data = GROUP student by age;

Dump group_data;
```

3. **JOIN**

```
customers = LOAD 'customer.txt' USING PigStorage(',') as (id:int, name:chararray, age:int,
address:chararray, salary:int);

orders = LOAD 'order.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int,
amount:int);

join_result = JOIN customers BY id, orders BY customer_id;

Dump join_result
```

//Customer.txt

```
1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00
```

//Order.txt

```
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,2,1560
103,2008-05-20 00:00:00,4,2060
```

4. **Sorting**

```
student = LOAD 'student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, age:int, phone:chararray, city:chararray);

student_order = ORDER student BY age DESC;
```

```
student_limit = LIMIT student_order 4;

Dump student_limit;
```