Course: MScCS

Year: 2020-2022

Semester: IV

Program: Computer Science

Subject: Big Data Engineering Tools and Frameworks

Subject Code: PS-SCS-404

Seat Number: KSMSCCS012

Name: Tanay Kulkarni

University: HSNC University

College: KC College, Churchgate

Signature

____

Date

____

Practical No 1

Steps for Install Hadoop on Windows Based Platform

First, we need to make sure that the following prerequisites are installed:

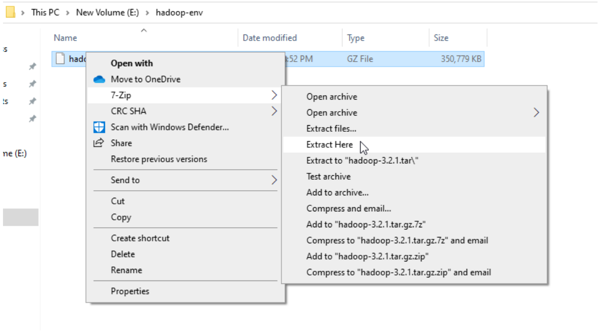
1. Java 8 runtime environment (JRE): Hadoop 3 requires a Java 8 installation. I prefer using the offline installer.

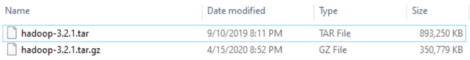
2. Java 8 development Kit (JDK)

3. To unzip downloaded Hadoop binaries, we should install 7zip.

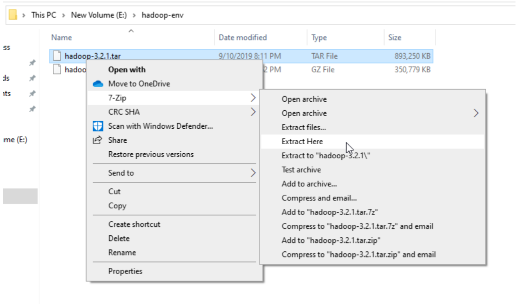
4. I will create a folder “E:\hadoop-env” on my local machine to store downloaded files.

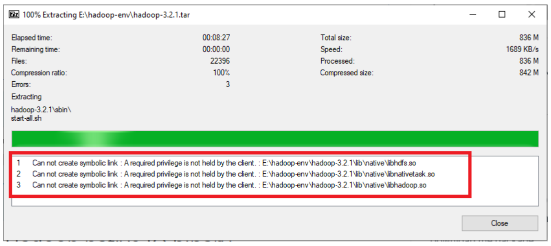
2. Download Hadoop binaries
The first step is to download Hadoop binaries from the official website.
https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz


This PC › New Volume (E:) › hadoop-env
Name
Date modified
Type
Size
hado
:52 PM
GZ File
350,779 KB
Open with
Move to OneDrive
7-Zip
CRC SHA
Scan with Windows Defender...
Share
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Open archive
Open archive
Extract files...
Extract Here
Extract to "hadoop-3.2.1.tar\"
Test archive
Add to archive...
Compress and email...
Add to "hadoop-3.2.1.tar.gz.7z"
Compress to "hadoop-3.2.1.tar.gz.7z" and email
Add to "hadoop-3.2.1.tar.gz.zip"
Compress to "hadoop-3.2.1.tar.gz.zip" and email



Name	Date modified	Type	Size
hadoop-3.2.1.tar	9/10/2019 8:11 PM	TAR File	893,250 KB
hadoop-3.2.1.tar.gz	4/15/2020 8:52 PM	GZ File	350,779 KB





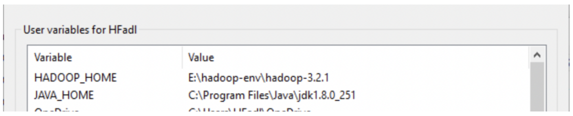
100% Extracting E:\hadoop-env\hadoop-3.2.1.tar

```
Elapsed time:          00:08:27     Total size:            836 M
Remaining time:        00:00:00     Speed:             1689 KB/s
Files:                    22396     Processed:             836 M
Compression ratio:         100%     Compressed size:       842 M
Errors:                       3
Extracting
hadoop-3.2.1\sbin\
start-all.sh

1   Can not create symbolic link : A required privilege is not held by the client. : E:\hadoop-env\hadoop-3.2.1\lib\native\libhdfs.so
2   Can not create symbolic link : A required privilege is not held by the client. : E:\hadoop-env\hadoop-3.2.1\lib\native\libnativetask.so
3   Can not create symbolic link : A required privilege is not held by the client. : E:\hadoop-env\hadoop-3.2.1\lib\native\libhadoop.so
```

After unpacking the package, add the Hadoop native IO libraries, which can be found in the following GitHub repository:
https://github.com/cdarlint/winutils

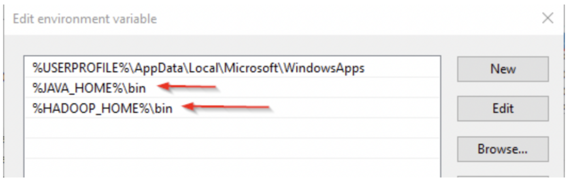
Since we are installing Hadoop 3.2.1, download the files located in https://github.com/cdarlint/winutils/tree/master/hadoop-3.2.1/bin and copy them into the “hadoop-3.2.1\bin” directory.

3. Setting up environment variables
After installing Hadoop and its prerequisites, we should configure the environment variables to define Hadoop and Java default paths.



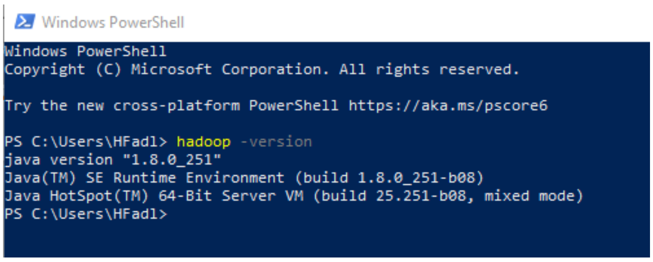
User variables for HFadl

Variable	Value
HADOOP_HOME	E:\hadoop-env\hadoop-3.2.1
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_251



Edit environment variable

```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
%JAVA_HOME%\bin
%HADOOP_HOME%\bin
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HFadl> hadoop -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.251-b08, mixed mode)
PS C:\Users\HFadl>
```

open “hdfs-site.xml” file located in “%HADOOP_HOME%\etc\hadoop” directory, and we should add the following properties within the <configuration></configuration> element:

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///E:/hadoop-env/hadoop-3.2.1/data/dfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:///E:/hadoop-env/hadoop-3.2.1/data/dfs/datanode</value>
</property>
```

configure the name node URL adding the following XML code into the <configuration></configuration> element within “core-site.xml”:

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9820</value>
</property>
```

add the following XML code into the <configuration></configuration> element within “mapred-site.xml”:

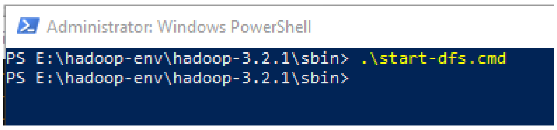
```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
<description>MapReduce framework name</description>
</property>
```

add the following XML code into the <configuration></configuration> element within “yarn-site.xml”:

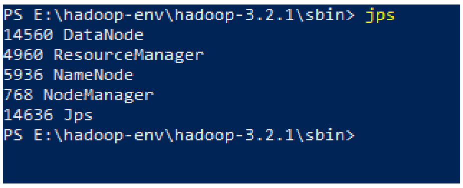
```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
```

<description>Yarn Node Manager Aux Service</description>
</property>

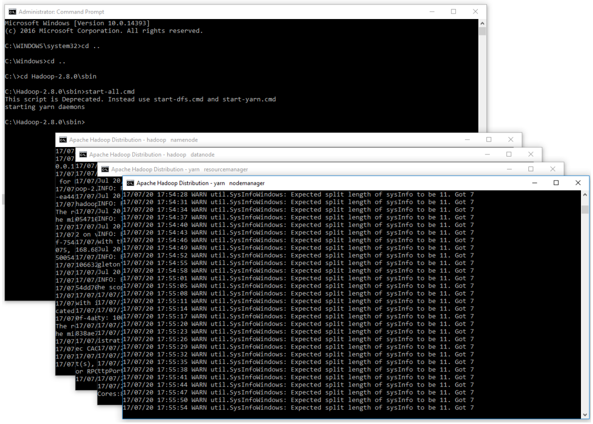
hdfs namenode -format



```
PS E:\hadoop-env\hadoop-3.2.1\sbin> .\start-dfs.cmd
PS E:\hadoop-env\hadoop-3.2.1\sbin>
```

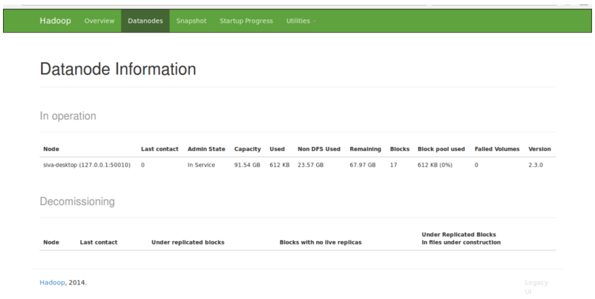


```
PS E:\hadoop-env\hadoop-3.2.1\sbin> jps
14560 DataNode
4960 ResourceManager
5936 NameNode
768 NodeManager
14636 Jps
PS E:\hadoop-env\hadoop-3.2.1\sbin>
```


Administrator: Command Prompt
Apache Hadoop Distribution - hadoop namenode
Apache Hadoop Distribution - hadoop datanode
Apache Hadoop Distribution - yarn resourcemanager
Apache Hadoop Distribution - yarn nodemanager

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd ..

C:\Windows>cd ..

C:\>cd Hadoop-2.8.0\sbin

C:\Hadoop-2.8.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\Hadoop-2.8.0\sbin>
```

```
17/07/20 17:54:28 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:31 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:34 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:37 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:40 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:43 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:46 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:49 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:52 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:55 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:54:58 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:01 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:05 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:08 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:11 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:14 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:17 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:20 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:23 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:26 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:29 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:32 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:35 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:38 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:41 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:44 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:47 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:50 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
17/07/20 17:55:54 WARN util.SysInfoWindows: Expected split length of sysInfo to be 11. Got 7
```



Datanode Information

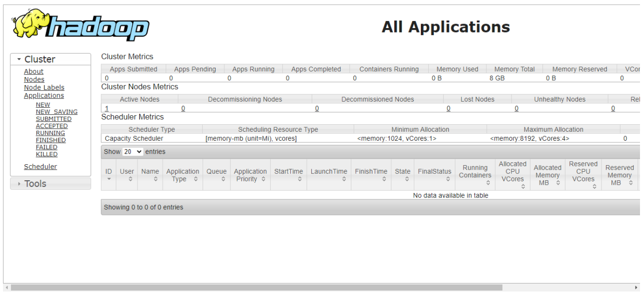
In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
siva-desktop (127.0.0.1:50010)	0	In Service	91.54 GB	612 KB	23.57 GB	67.97 GB	17	612 KB (0%)	0	2.3.0

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

Hadoop, 2014.



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCor
0	0	0	0	0	0 B	8 GB	0 B	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rel
1	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB

No data available in table

Showing 0 to 0 of 0 entries

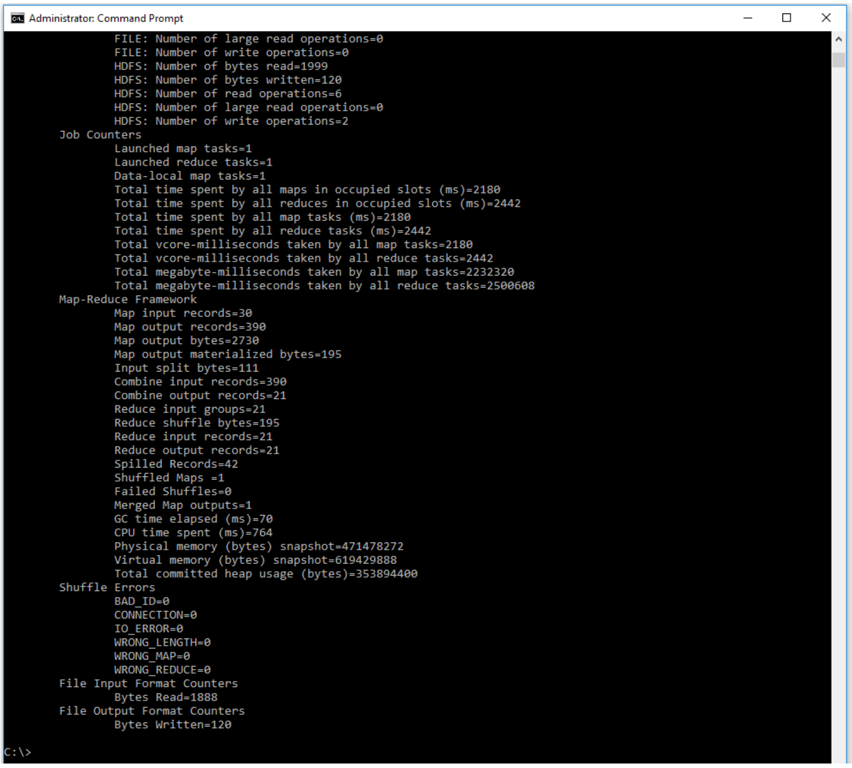
Practical No 2

Hadoop Word Count

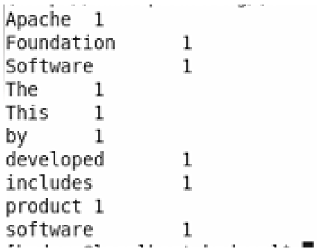
Code:

```
import java.io.IOException;
import java.util.*;
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 WordCount {
 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        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();
      conf.set("mapred.job.tracker", "hdfs://localhost:50001");
      conf.set("fs.default.name", "hdfs://localhost:50000");
        Job job = new Job(conf, "wordcount");

    job.setJarByClass(WordCount.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

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

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

    job.waitForCompletion(true);
  }
}
```

Output:



```
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1999
                HDFS: Number of bytes written=120
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2180
                Total time spent by all reduces in occupied slots (ms)=2442
                Total time spent by all map tasks (ms)=2180
                Total time spent by all reduce tasks (ms)=2442
                Total vcore-milliseconds taken by all map tasks=2180
                Total vcore-milliseconds taken by all reduce tasks=2442
                Total megabyte-milliseconds taken by all map tasks=2232320
                Total megabyte-milliseconds taken by all reduce tasks=2500608
        Map-Reduce Framework
                Map input records=30
                Map output records=390
                Map output bytes=2730
                Map output materialized bytes=195
                Input split bytes=111
                Combine input records=390
                Combine output records=21
                Reduce input groups=21
                Reduce shuffle bytes=195
                Reduce input records=21
                Reduce output records=21
                Spilled Records=42
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=70
                CPU time spent (ms)=764
                Physical memory (bytes) snapshot=471478272
                Virtual memory (bytes) snapshot=619429888
                Total committed heap usage (bytes)=353894400
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1888
        File Output Format Counters
                Bytes Written=120

C:\>
```



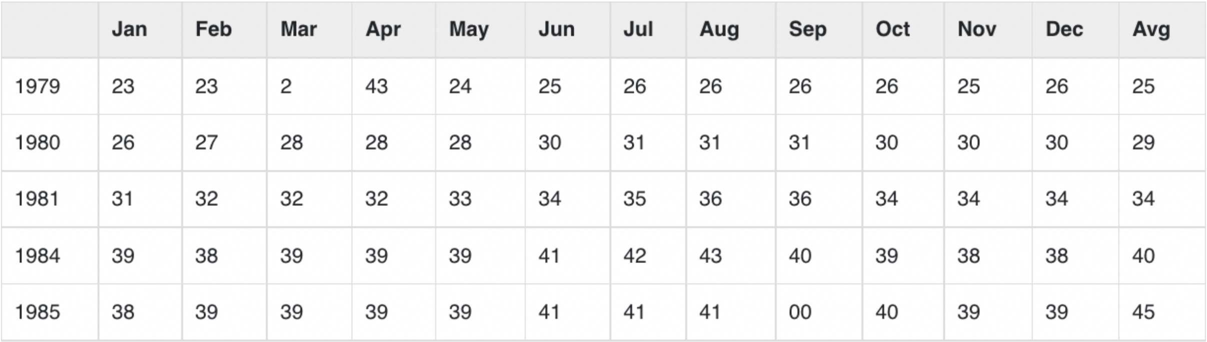
```
Apache  1
Foundation      1
Software        1
The     1
This    1
by      1
developed       1
includes        1
product 1
software        1
```

Practical No 3

Example working with Hadoop Map Reduce

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

If the above data is given as input, write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on.



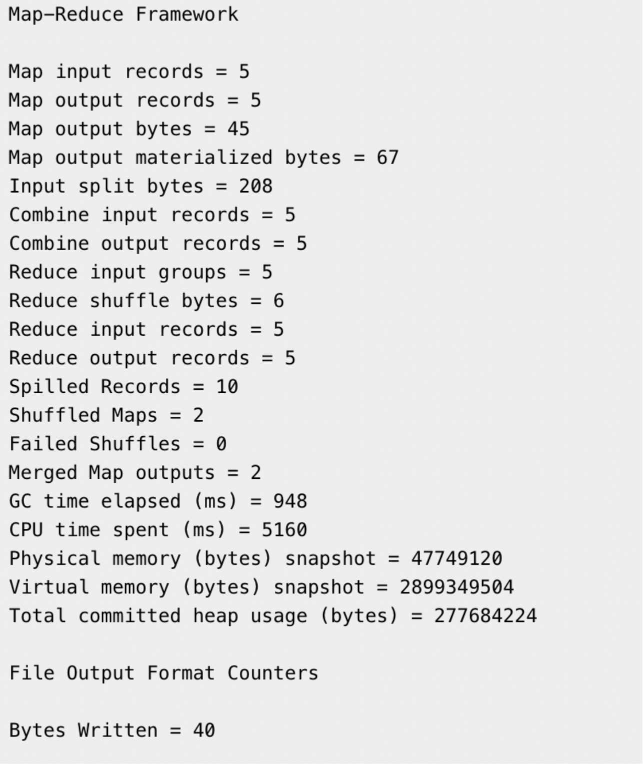
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

```
package hadoop;

import java.util.*;

import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits {
  //Mapper class
  public static class E_EMapper extends MapReduceBase implements
  Mapper<LongWritable ,/*Input key Type */
  Text,                /*Input value Type*/
  Text,                /*Output key Type*/
  IntWritable>         /*Output value Type*/
  {
    //Map function
    public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output,

    Reporter reporter) throws IOException {
      String line = value.toString();
      String lasttoken = null;
      StringTokenizer s = new StringTokenizer(line,"\t");
      String year = s.nextToken();

      while(s.hasMoreTokens()) {
        lasttoken = s.nextToken();
```

```
         }
         int avgprice = Integer.parseInt(lasttoken);
         output.collect(new Text(year), new IntWritable(avgprice));
      }
   }

   //Reducer class
   public static class E_EReduce extends MapReduceBase implements Reducer< Text, IntWritable, Text, IntWritable > {

      //Reduce function
      public void reduce( Text key, Iterator <IntWritable> values,
      OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
         int maxavg = 30;
         int val = Integer.MIN_VALUE;

         while (values.hasNext()) {
            if((val = values.next().get())>maxavg) {
               output.collect(key, new IntWritable(val));
            }
         }
      }
   }

   //Main function
   public static void main(String args[])throws Exception {
      JobConf conf = new JobConf(ProcessUnits.class);

      conf.setJobName("max_eletricityunits");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(E_EMapper.class);
      conf.setCombinerClass(E_EReduce.class);
      conf.setReducerClass(E_EReduce.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));

      JobClient.runJob(conf);
   }
}
```

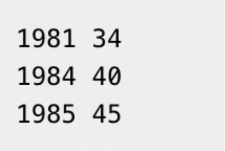
```
$ mkdir units
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
$ jar -cvf units.jar -C units/ .
```

```
$HADOOP_HOME/bin/hadoop fs -mkdir input_dir
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
$HADOOP_HOME/bin/hadoop fs -ls input_dir/
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir
```



```
Map-Reduce Framework

Map input records = 5
Map output records = 5
Map output bytes = 45
Map output materialized bytes = 67
Input split bytes = 208
Combine input records = 5
Combine output records = 5
Reduce input groups = 5
Reduce shuffle bytes = 6
Reduce input records = 5
Reduce output records = 5
Spilled Records = 10
Shuffled Maps = 2
Failed Shuffles = 0
Merged Map outputs = 2
GC time elapsed (ms) = 948
CPU time spent (ms) = 5160
Physical memory (bytes) snapshot = 47749120
Virtual memory (bytes) snapshot = 2899349504
Total committed heap usage (bytes) = 277684224

File Output Format Counters

Bytes Written = 40
```

$HADOOP_HOME/bin/hadoop fs -ls output_dir/
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000



```
1981 34
1984 40
1985 45
```

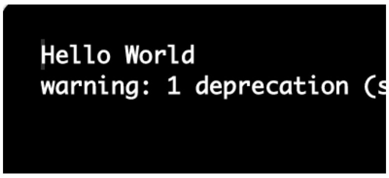
copy the output folder from HDFS to the local file system.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

Practical No 4

a. Write a Scala program to print "hello world"

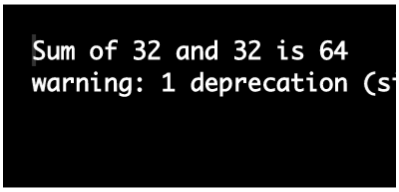
```
object MainObject
{
   def main(args:Array[String])
   {
      print("Hello World")
   }
}
```



```
Hello World
warning: 1 deprecation (s
```

b. Write a Scala program to compute the sum of the two given integer value, if the value are the same then return their sum.

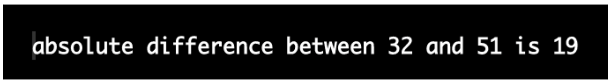
```
object MainObject
{
   def main(args:Array[String])
   {
       val a = scala.io.StdIn.readInt()
       val b = scala.io.StdIn.readInt()
       if(a==b){
          print(s"Sum of $a and $b is " + (a + b))
       }else {
          print(s"$a and $b are not same")
       }
   }
}
```



```
Sum of 32 and 32 is 64
warning: 1 deprecation (s
```

c. Write a Scala program to get the absolute difference between n to 51. If n is greater than 51 , error message should display.

```
object MainObject
{
   def main(args:Array[String])
   {
       val n = scala.io.StdIn.readInt()
       val n_abs = n.abs
       if(n_abs > 51) {
          println(s"Invalid Input")
       }else {
          val diff = 51 - n_abs
          println(s"absolute difference between $n and 51 is $diff")
       }
   }
}
```

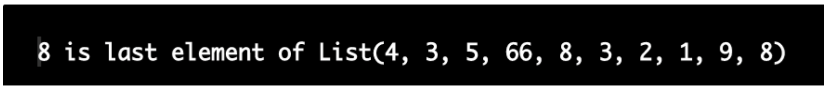


```
absolute difference between 32 and 51 is 19
```

Practical No 5

a.Write a Scala program to check if a given number is present in first or the last position of given array

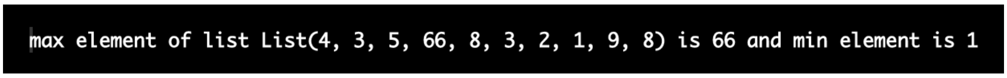
```
object MainObject
{
   def main(args:Array[String])
   {
        val list = List(4,3,5,66,8,3,2,1,9,8)
        val n = scala.io.StdIn.readInt()
        if(n == list.head){
           println(s"$n is first element of $list")
        }else if(n == list.last){
           println(s"$n is last element of $list")
        }else {
           println(s"$n is not first or last element of list")
        }
   }
}
```



```
8 is last element of List(4, 3, 5, 66, 8, 3, 2, 1, 9, 8)
```

b. Write a Scala program to find the maximum and minimum value of an array of integers.

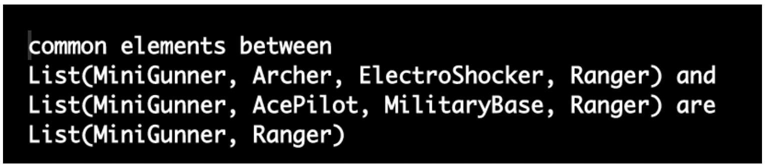
```
object MainObject
{
   def main(args:Array[String])
   {
        val list =
List(4,3,5,66,8,3,2,1,9,8)
        val mx = list.max
        val mn = list.min
        println(s"max element of list $list is $mx and min element is $mn")
   }
}
```



```
max element of list List(4, 3, 5, 66, 8, 3, 2, 1, 9, 8) is 66 and min element is 1
```

c. Write a Scala program to find the common element between two arrays of string.

```
object MainObject
{
   def main(args:Array[String])
   {
        val s1 = List("MiniGunner", "Archer", "ElectroShocker", "Ranger")
        val s2 = List("MiniGunner", "AcePilot", "MilitaryBase", "Ranger")
        val common = s1.intersect(s2)
        println(s"common elements between $s1 and $s2 are $common")
   }
}
```

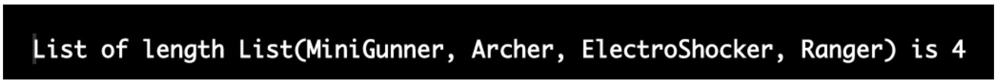


```
common elements between
List(MiniGunner, Archer, ElectroShocker, Ranger) and
List(MiniGunner, AcePilot, MilitaryBase, Ranger) are
List(MiniGunner, Ranger)
```

Practical No 6

a. Write a Scala program to calculate the length of a given list.

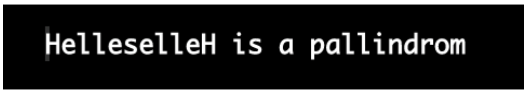
```
object MainObject
{
   def main(args:Array[String])
   {
       val s1 = List("MiniGunner", "Archer", "ElectroShocker", "Ranger")
       val len = s1.length
       println(s"List of length $s1 is $len")
   }
}
```



```
List of length List(MiniGunner, Archer, ElectroShocker, Ranger) is 4
```

b. Write a Scala program to check a given list is a palindrome or not.

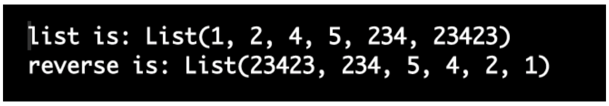
```
object MainObject
{
   def main(args:Array[String])
   {
       val str = scala.io.StdIn.readLine()
       val str_r = str.reverse
       if(str == str_r){
          println(s"$str is a pallindrom")
       }else {
          println(s"$str is not a pallindrome")
       }
   }
}
```



```
HelleselleH is a pallindrom
```

c. Write a Scala program to reverse a given list.

```
object MainObject
{
   def main(args:Array[String])
   {
       val str = scala.io.StdIn.readLine()
       val str_r = str.reverse
       if(str == str_r){
          println(s"$str is a pallindrom")
       }else {
          println(s"$str is not a pallindrome")
       }
   }
}
```



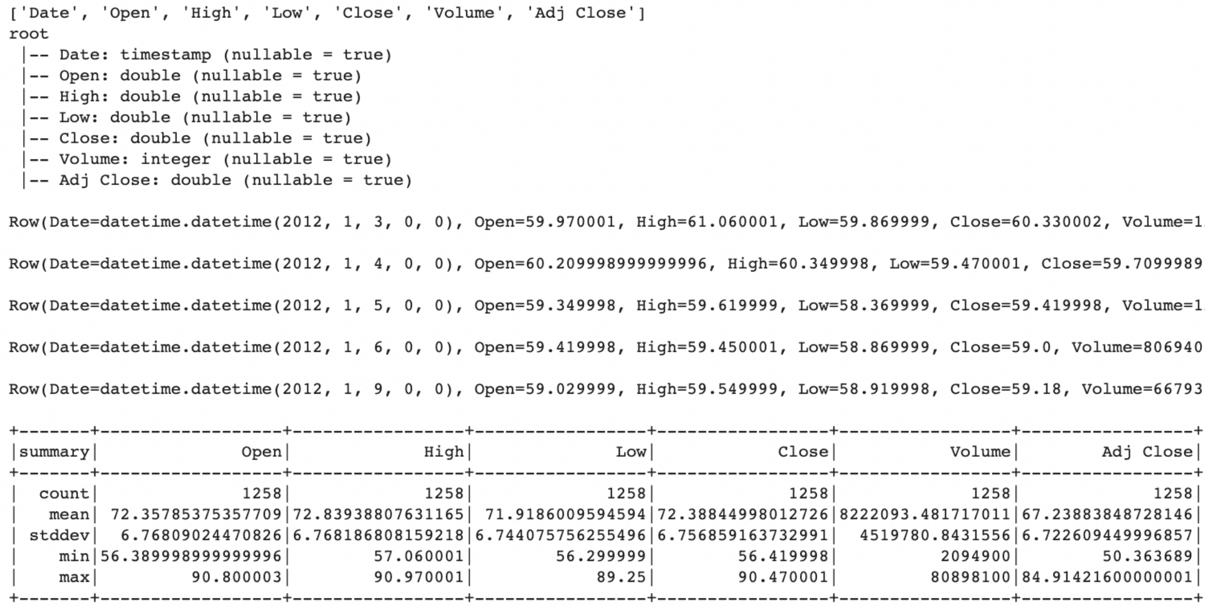
```
list is: List(1, 2, 4, 5, 234, 23423)
reverse is: List(23423, 234, 5, 4, 2, 1)
```

Practical No 7

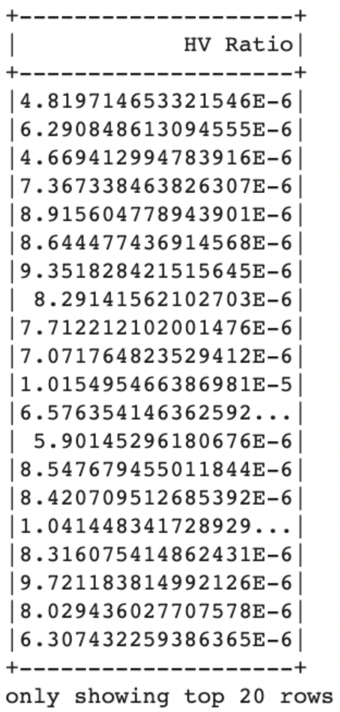
Working with Spark

1.What are the column name
2.What does the Schema look like
3.Print out the first 5 columns
4.Display mean, count, stdev, min, max
5. Create a new dataframe with a column called hvratio that is the ratio of the high price verus volume of stock traded for a day

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('walmart').getOrCreate()
df = spark.read.csv('walmart_stock.csv', inferSchema=True, header=True)
print(df.columns)
df.printSchema()
for line in df.head(5):
print(line, '\n')
df.describe().show()
df_hv = df.withColumn('HV Ratio', df['High']/df['Volume']).select(['HV Ratio'])
df_hv.show()
```



```
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)

Row(Date=datetime.datetime(2012, 1, 3, 0, 0), Open=59.970001, High=61.060001, Low=59.869999, Close=60.330002, Volume=1

Row(Date=datetime.datetime(2012, 1, 4, 0, 0), Open=60.209998999999996, High=60.349998, Low=59.470001, Close=59.7099989

Row(Date=datetime.datetime(2012, 1, 5, 0, 0), Open=59.349998, High=59.619999, Low=58.369999, Close=59.419998, Volume=1

Row(Date=datetime.datetime(2012, 1, 6, 0, 0), Open=59.419998, High=59.450001, Low=58.869999, Close=59.0, Volume=806940

Row(Date=datetime.datetime(2012, 1, 9, 0, 0), Open=59.029999, High=59.549999, Low=58.919998, Close=59.18, Volume=66793

+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|              Open|             High|              Low|            Close|           Volume|        Adj Close|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|              1258|             1258|             1258|             1258|             1258|             1258|
|   mean| 72.35785375357709|72.83938807631165| 71.9186009594594|72.38844998012726|8222093.481717011|67.23883848728146|
| stddev|  6.76809024470826|6.768186808159218|6.744075756255496|6.756859163732991|  4519780.8431556|6.722609449996857|
|    min|56.389998999999996|        57.060001|        56.299999|        56.419998|          2094900|        50.363689|
|    max|         90.800003|        90.970001|            89.25|        90.470001|         80898100|84.91421600000001|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

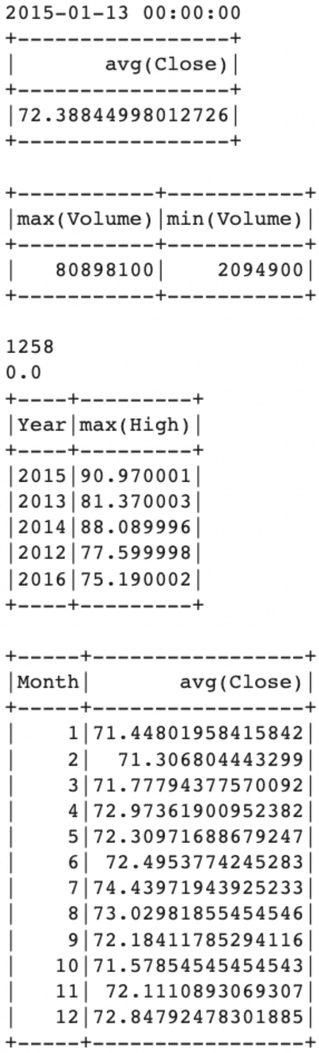


```
+--------------------+
|            HV Ratio|
+--------------------+
|4.819714653321546E-6|
|6.290848613094555E-6|
|4.669412994783916E-6|
|7.367338463826307E-6|
|8.915604778943901E-6|
|8.644477436914568E-6|
|9.351828421515645E-6|
| 8.29141562102703E-6|
|7.712212102001476E-6|
|7.071764823529412E-6|
|1.015495466386981E-5|
|6.576354146362592...|
| 5.90145296180676E-6|
|8.547679455011844E-6|
|8.420709512685392E-6|
|1.041448341728929...|
|8.316075414862431E-6|
|9.721183814992126E-6|
|8.029436027707578E-6|
|6.307432259386365E-6|
+--------------------+
only showing top 20 rows
```

Practical No 8

1. What day had the Peak high in price
2. What is the mean of the close column
3. What is the max and min of the volume column
4. How many days was the close lower than 220
5. What percentage of the time was the high greater than 250
6. What is the max high per year
7. What is the average close for each Calendar Month

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('walmart').getOrCreate()
df = spark.read.csv('walmart_stock.csv', inferSchema=True, header=True)
print(df.orderBy(df['High'].desc()).select(['Date']).head(1)[0]['Date'])
from pyspark.sql.functions import mean
df.select(mean('Close')).show()
from pyspark.sql.functions import min, max
df.select(max('Volume'),min('Volume')).show()
print(df.filter(df['Close'] < 220).count())
print(df.filter('High > 250').count() * 100/df.count())
from pyspark.sql.functions import (dayofmonth, hour,
dayofyear, month,
year, weekofyear,
format_number, date_format)
year_df = df.withColumn('Year', year(df['Date']))
year_df.groupBy('Year').max()['Year', 'max(High)'].show()
#Create a new column Month from existing Date column
month_df = df.withColumn('Month', month(df['Date']))
#Group by month and take average of all other columns
month_df = month_df.groupBy('Month').mean()
#Sort by month
month_df = month_df.orderBy('Month')
#Display only month and avg(Close), the desired columns
month_df['Month', 'avg(Close)'].show()
```

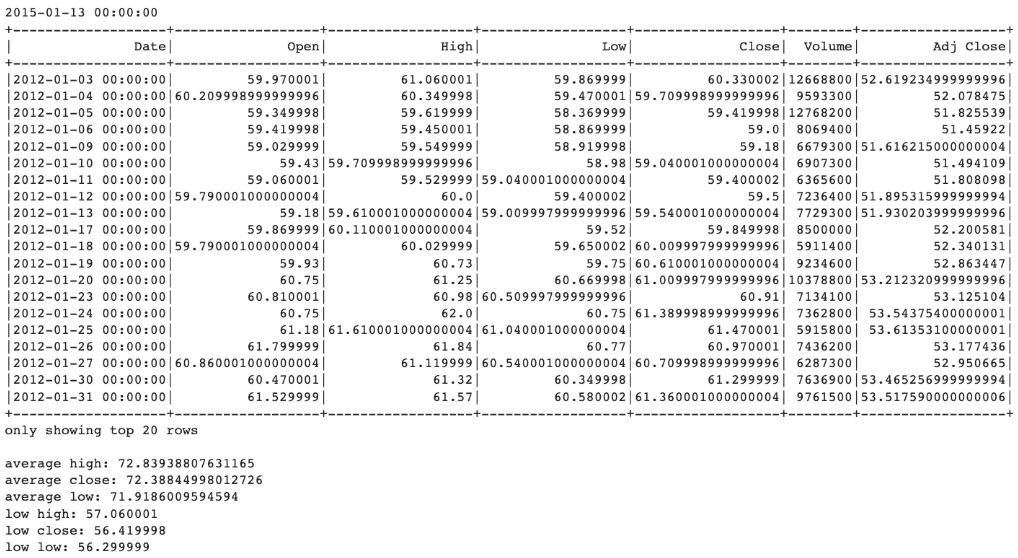


```
2015-01-13 00:00:00
+-----------------+
|       avg(Close)|
+-----------------+
|72.38844998012726|
+-----------------+

+-----------+-----------+
|max(Volume)|min(Volume)|
+-----------+-----------+
|   80898100|    2094900|
+-----------+-----------+

1258
0.0
+----+---------+
|Year|max(High)|
+----+---------+
|2015|90.970001|
|2013|81.370003|
|2014|88.089996|
|2012|77.599998|
|2016|75.190002|
+----+---------+

+-----+-----------------+
|Month|       avg(Close)|
+-----+-----------------+
|    1|71.44801958415842|
|    2|  71.306804443299|
|    3|71.77794377570092|
|    4|72.97361900952382|
|    5|72.30971688679247|
|    6| 72.4953774245283|
|    7|74.43971943925233|
|    8|73.02981855454546|
|    9|72.18411785294116|
|   10|71.57854545454543|
|   11| 72.1110893069307|
|   12|72.84792478301885|
+-----+-----------------+
```

Practical No 9:

Spark SQL connecting with Data Source : Display all the High and Closing price
What is the average high price , close price, low price
What is the lowest price in high, close and low price

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('walmart').getOrCreate()
df = spark.read.csv('walmart_stock.csv', inferSchema=True, header=True)
print(df.orderBy(df['High'].desc()).select(['Date']).head(1)[0]['Date'])
from pyspark.sql.functions import mean

avg_high = df.agg({"High": "avg"}).collect()[0][0]
print("average high: " + str(avg_high))

avg_close = df.agg({"Close": "avg"}).collect()[0][0]
print("average close: " + str(avg_close))

avg_low = df.agg({"Low": "avg"}).collect()[0][0]
print("average low: " + str(avg_low))

low_high = df.agg({"High": "min"}).collect()[0][0]
print("low high: " + str(low_high))

low_close = df.agg({"Close": "min"}).collect()[0][0]
print("low close: " + str(low_close))

low_low = df.agg({"Low": "min"}).collect()[0][0]
print("low low: " + str(low_low))
```



```
2015-01-13 00:00:00
+-------------------+------------------+------------------+------------------+------------------+--------+------------------+
|               Date|              Open|              High|               Low|             Close|  Volume|         Adj Close|
+-------------------+------------------+------------------+------------------+------------------+--------+------------------+
|2012-01-03 00:00:00|         59.970001|         61.060001|         59.869999|         60.330002|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.209998999999996|         60.349998|         59.470001|59.709998999999996| 9593300|         52.078475|
|2012-01-05 00:00:00|         59.349998|         59.619999|         58.369999|         59.419998|12768200|         51.825539|
|2012-01-06 00:00:00|         59.419998|         59.450001|         58.869999|              59.0| 8069400|          51.45922|
|2012-01-09 00:00:00|         59.029999|         59.549999|         58.919998|             59.18| 6679300|51.616215000000004|
|2012-01-10 00:00:00|             59.43|59.709998999999996|             58.98|59.040001000000004| 6907300|         51.494109|
|2012-01-11 00:00:00|         59.060001|         59.529999|59.040001000000004|         59.400002| 6365600|         51.808098|
|2012-01-12 00:00:00|59.790001000000004|              60.0|         59.400002|              59.5| 7236400|51.895315999999994|
|2012-01-13 00:00:00|             59.18|59.610001000000004|59.009997999999996|59.540001000000004| 7729300|51.930203999999996|
|2012-01-17 00:00:00|         59.869999|60.110001000000004|             59.52|         59.849998| 8500000|         52.200581|
|2012-01-18 00:00:00|59.790001000000004|         60.029999|         59.650002|60.009997999999996| 5911400|         52.340131|
|2012-01-19 00:00:00|             59.93|             60.73|             59.75|60.610001000000004| 9234600|         52.863447|
|2012-01-20 00:00:00|             60.75|             61.25|         60.669998|61.009997999999996|10378800|53.212320999999996|
|2012-01-23 00:00:00|         60.810001|             60.98|60.509997999999996|             60.91| 7134100|         53.125104|
|2012-01-24 00:00:00|             60.75|              62.0|             60.75|61.389998999999996| 7362800| 53.54375400000001|
|2012-01-25 00:00:00|             61.18|61.610001000000004|61.040001000000004|         61.470001| 5915800| 53.61353100000001|
|2012-01-26 00:00:00|         61.799999|             61.84|             60.77|         60.970001| 7436200|         53.177436|
|2012-01-27 00:00:00|60.860001000000004|         61.119999|60.540001000000004|60.709998999999996| 6287300|         52.950665|
|2012-01-30 00:00:00|         60.470001|             61.32|         60.349998|         61.299999| 7636900|53.465256999999994|
|2012-01-31 00:00:00|         61.529999|             61.57|         60.580002|61.360001000000004| 9761500|53.517590000000006|
+-------------------+------------------+------------------+------------------+------------------+--------+------------------+
only showing top 20 rows

average high: 72.83938807631165
average close: 72.38844998012726
average low: 71.9186009594594
low high: 57.060001
low close: 56.419998
low low: 56.299999
```

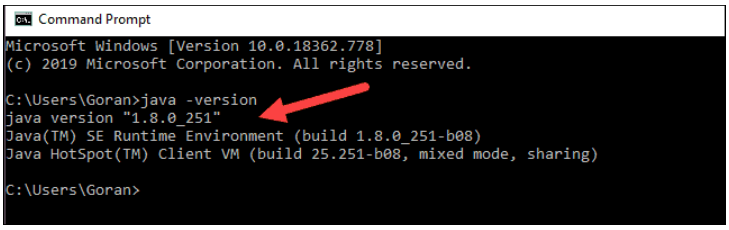
Practical No 10:

Installation SPark and Scala on window based application

Install Java 8
Apache Spark requires Java 8. You can check to see if Java is installed using the command prompt.

Open the command line by clicking Start > type cmd > click Command Prompt.

Type the following command in the command prompt:



```
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Goran>java -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) Client VM (build 25.251-b08, mixed mode, sharing)

C:\Users\Goran>
```

Install Apache Spark
Installing Apache Spark involves extracting the downloaded file to the desired location.

1. Create a new folder named Spark in the root of your C: drive. From a command line, enter the following:

cd \

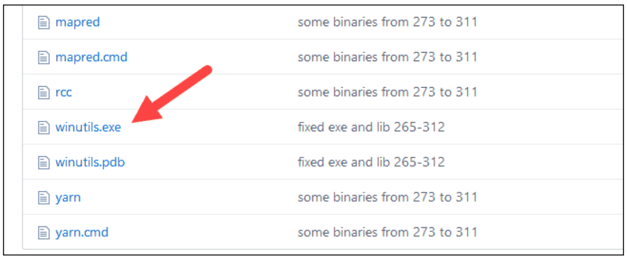
mkdir Spark
2. In Explorer, locate the Spark file you downloaded.

3. Right-click the file and extract it to C:\Spark using the tool you have on your system (e.g., 7-Zip).

4. Now, your C:\Spark folder has a new folder spark-2.4.5-bin-hadoop2.7 with the necessary files inside.

Add winutils.exe File
Download the winutils.exe file for the underlying Hadoop version for the Spark installation you downloaded.

1. Navigate to this URL https://github.com/cdarlint/winutils and inside the bin folder, locate winutils.exe, and click it.



mapred	some binaries from 273 to 311
mapred.cmd	some binaries from 273 to 311
rcc	some binaries from 273 to 311
winutils.exe	fixed exe and lib 265-312
winutils.pdb	fixed exe and lib 265-312
yarn	some binaries from 273 to 311
yarn.cmd	some binaries from 273 to 311

2. Find the Download button on the right side to download the file.

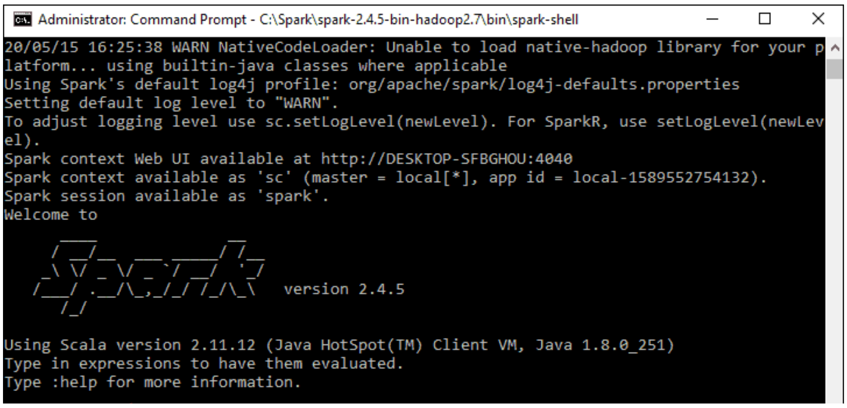
3. Now, create new folders Hadoop and bin on C: using Windows Explorer or the Command Prompt.

4. Copy the winutils.exe file from the Downloads folder to C:\hadoop\bin.

For Hadoop, the variable name is HADOOP_HOME and for the value use the path of the folder you created earlier: C:\hadoop. Add C:\hadoop\bin to the Path variable field, but we recommend using %HADOOP_HOME%\bin.
For Java, the variable name is JAVA_HOME and for the value use the path to your Java JDK directory (in our case it’s C:\Program Files\Java\jdk1.8.0_251).

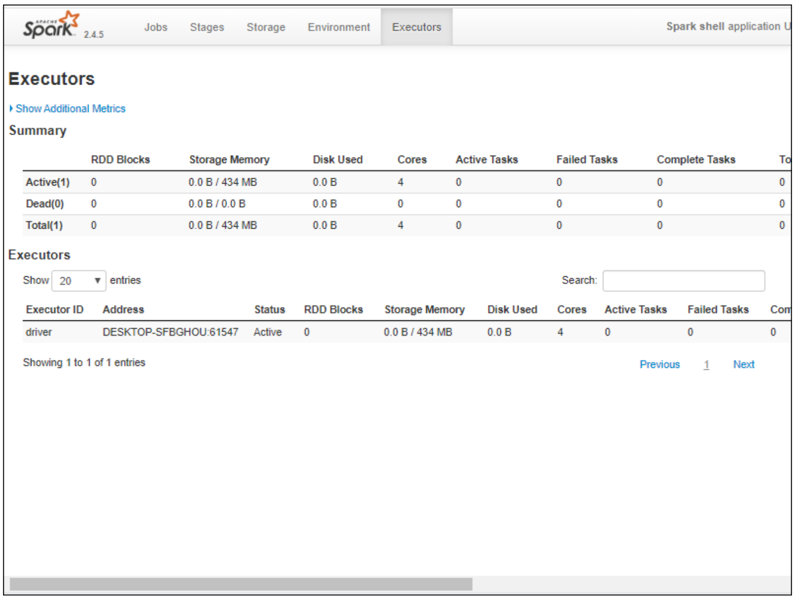
To start Spark, enter:

C:\Spark\spark-2.4.5-bin-hadoop2.7\bin\spark-shell



Administrator: Command Prompt - C:\Spark\spark-2.4.5-bin-hadoop2.7\bin\spark-shell

```
20/05/15 16:25:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your p
latform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLev
el).
Spark context Web UI available at http://DESKTOP-SFBGHOU:4040
Spark context available as 'sc' (master = local[*], app id = local-1589552754132).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) Client VM, Java 1.8.0_251)
Type in expressions to have them evaluated.
Type :help for more information.
```



Executors

▸ Show Additional Metrics

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	To
Active(1)	0	0.0 B / 434 MB	0.0 B	4	0	0	0	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0
Total(1)	0	0.0 B / 434 MB	0.0 B	4	0	0	0	0

Executors

Show 20 entries   Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Com
driver	DESKTOP-SFBGHOU:61547	Active	0	0.0 B / 434 MB	0.0 B	4	0	0	0

Showing 1 to 1 of 1 entries