TAYLOR' S
UNIVERSITY
Wisdom • Integrity • Excellence

Bachelor of Computer Science (Hons)
Bachelor of Software Engineering (Hons)
Bachelor of Information Technology (Hons)

Module Code : ITS66904 (March 2023)

Module Name : Big Data Technologies

Assignment No./Title	**Assignment - Individual (15%)**
Course Tutor/Lecturer	Mr. Sham Shul Shukri Mat
Submission Date	19 February 2023 1159pm
Student ID 0354208	**Student Name** Ishihara Satoaki

Declaration *(need to be signed by students. Otherwise, the assessment will not be evaluated)*
Certify that this assignment is entirely my own work, except where I have given fully documented references to the work of others, and that the material contained in this assignment has not previously been submitted for assessment in any other formal course of study.

Signature of Students:

Marks/Grade:	**Evaluated by:**
Evaluator's Comments:	

Table of Contents

Introduction

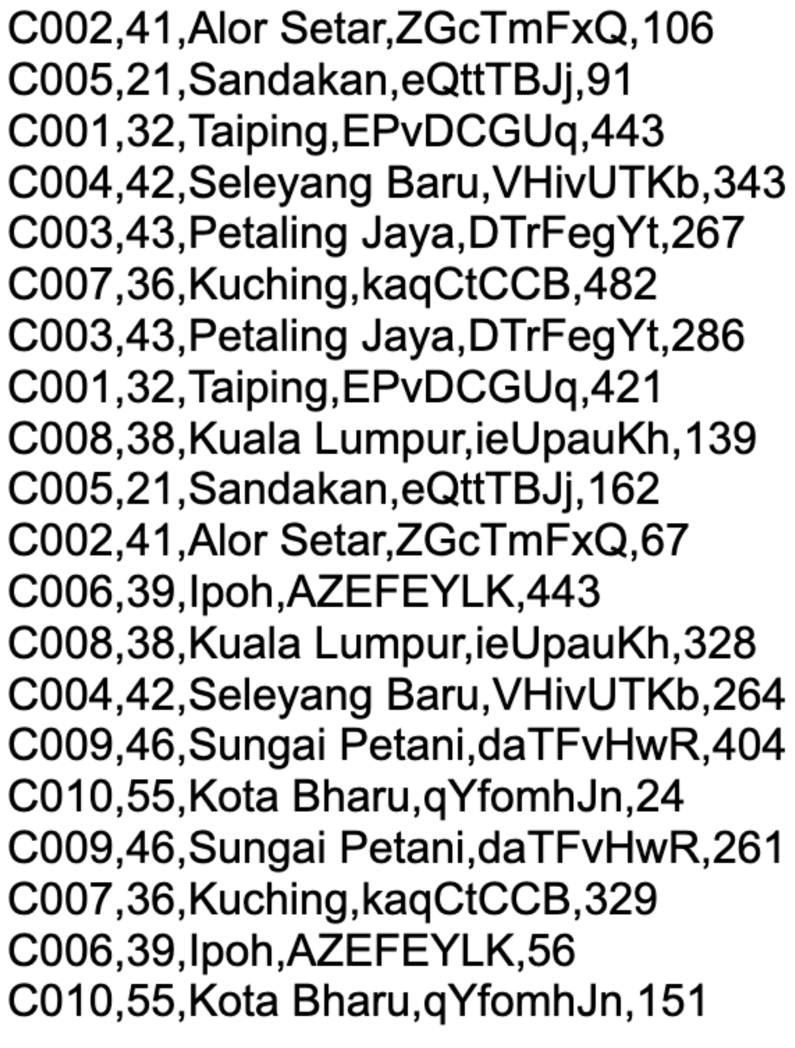
This is the individual assignment from ITS66904 (January 2023) Big Data Technologies. This assignment is expecting an accurately organized and suitably arranged scholarly manuscript that comprises your commercial proposal's aim and portrayal of your unprocessed data. Moreover, your manuscript ought to incorporate the undertakings listed in the aforementioned Duties. As a result of these scheduled outputs, I will explicate business objectives and employ unprocessed facts comprehensively, portray each phase of Hadoop HDFS and MapReduce in a tangible manner.

Business Case and Task Objective

Noir is a mobile super-app that offers diverse features such as payment and monetary transaction handling. Ever since its introduction a couple of years back, the application has consistently expanded its user base to the extent that it has been downloaded by a million individuals. In honor of the one millionth subscriber, we resolved to extend a unique coupon to clients who have been utilizing the application for a while and have carried out the highest sum of payment via the app. In fulfilling the objectives of this business case, the goal of this assignment is to maximize the use of Hadoop HDFS and MapReduce operations for data analysis, extraction of results, and business decisions based on those results, and to deploy the operations appropriately.

Raw Dataset Description

The raw dataset for this assignment is saved in text file names “satoaki_input.txt” in /user/satoaki_input folder. Dataset is constructed with 20 rows, 5 columns which are indicating customers’ ids, ages, cities customers are living, customers’ usernames and how much did each customer consume each payment. Two customers used the e-pay service for each day, and the campaign was operated for 10 days, so (2 x 10 =) 20 following customers data have gathered:



```
C002,41,Alor Setar,ZGcTmFxQ,106
C005,21,Sandakan,eQttTBJj,91
C001,32,Taiping,EPvDCGUq,443
C004,42,Seleyang Baru,VHivUTKb,343
C003,43,Petaling Jaya,DTrFegYt,267
C007,36,Kuching,kaqCtCCB,482
C003,43,Petaling Jaya,DTrFegYt,286
C001,32,Taiping,EPvDCGUq,421
C008,38,Kuala Lumpur,ieUpauKh,139
C005,21,Sandakan,eQttTBJj,162
C002,41,Alor Setar,ZGcTmFxQ,67
C006,39,Ipoh,AZEFEYLK,443
C008,38,Kuala Lumpur,ieUpauKh,328
C004,42,Seleyang Baru,VHivUTKb,264
C009,46,Sungai Petani,daTFvHwR,404
C010,55,Kota Bharu,qYfomhJn,24
C009,46,Sungai Petani,daTFvHwR,261
C007,36,Kuching,kaqCtCCB,329
C006,39,Ipoh,AZEFEYLK,56
C010,55,Kota Bharu,qYfomhJn,151
```

The customer paid most would be determined by rightmost column’s data.

HDFS and MapReduce Operations

The entire operations would begin from starting Namenode, Datanode, YARN resource and Node managers. Commands for these operations are as below:

```
start-dfs.sh
start-yarn.sh
```

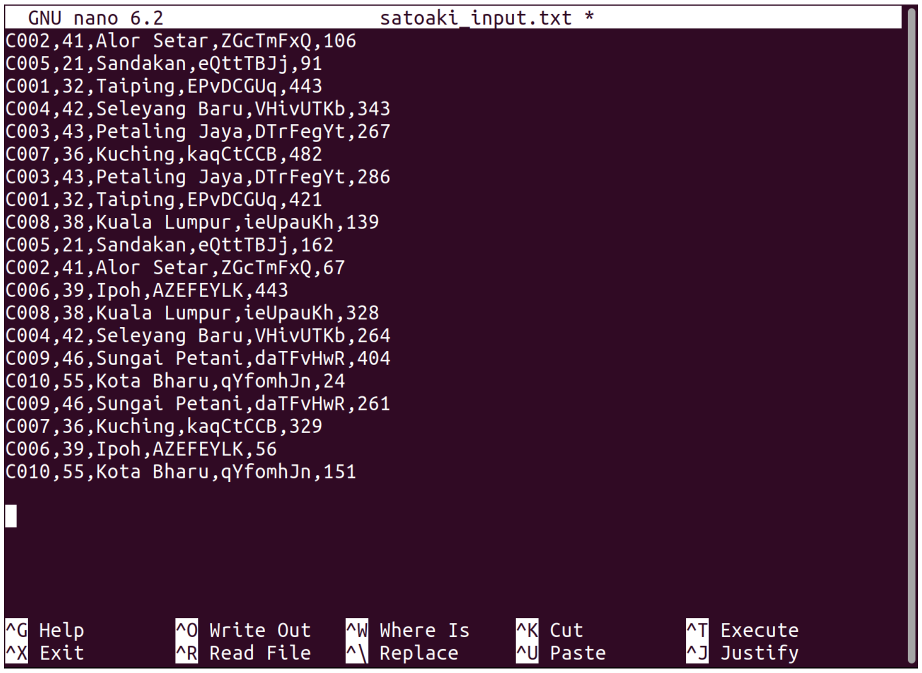
Now we can use wide variety of Linux-based HSFS commands.

1. HDFS Data Management

First, we need to manage data appropriately in suitable location. We create the "satoaki_input.txt" file and write down the raw dataset in the text file, and save with following commands:

```
nano satoaki_input.txt
```

So, we can see the "satoaki_input" text file is opened and set up for edition. We write down raw datasets in this file as below:



```
  GNU nano 6.2                  satoaki_input.txt *
C002,41,Alor Setar,ZGcTmFxQ,106
C005,21,Sandakan,eQttTBJj,91
C001,32,Taiping,EPvDCGUq,443
C004,42,Seleyang Baru,VHivUTKb,343
C003,43,Petaling Jaya,DTrFegYt,267
C007,36,Kuching,kaqCtCCB,482
C003,43,Petaling Jaya,DTrFegYt,286
C001,32,Taiping,EPvDCGUq,421
C008,38,Kuala Lumpur,ieUpauKh,139
C005,21,Sandakan,eQttTBJj,162
C002,41,Alor Setar,ZGcTmFxQ,67
C006,39,Ipoh,AZEFEYLK,443
C008,38,Kuala Lumpur,ieUpauKh,328
C004,42,Seleyang Baru,VHivUTKb,264
C009,46,Sungai Petani,daTFvHwR,404
C010,55,Kota Bharu,qYfomhJn,24
C009,46,Sungai Petani,daTFvHwR,261
C007,36,Kuching,kaqCtCCB,329
C006,39,Ipoh,AZEFEYLK,56
C010,55,Kota Bharu,qYfomhJn,151

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

And save the file by following shortcuts: [ctrl + o -> Enter key -> ctrl + x].

Not only this but also, we must save this data file in /user/satoaki_input. Create new directories with the following command:

```
$HADOOP_HOME/bin/hadoop fs -mkdir /user/satoaki_input
```

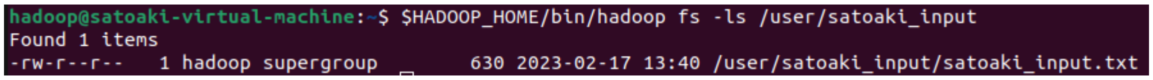
and export the data file in this directory with the following command:

```
$HADOOP_HOME/bin/hadoop fs -put satoaki_input.txt /user/satoaki_input
```

If we want to check whether the "satoaki_input.txt" is successfully saved in /user/satoaki_input, use the following command to verify:

```
$HADOOP_HOME/bin/hadoop fs -ls /user/satoaki_input
```

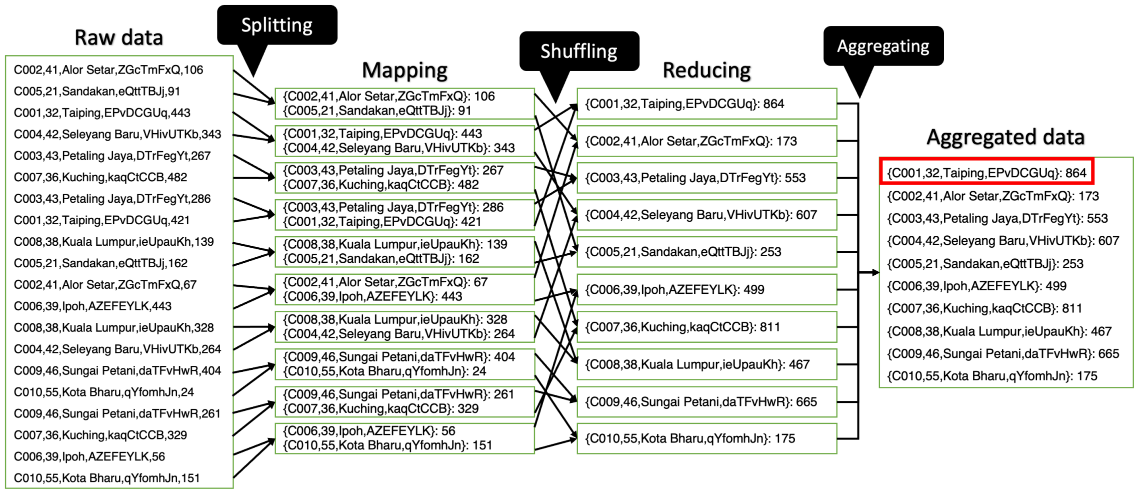
And if the following outcome is witnessed, the data file is successfully exported:



```
hadoop@satoaki-virtual-machine:~$ $HADOOP_HOME/bin/hadoop fs -ls /user/satoaki_input
Found 1 items
-rw-r--r--   1 hadoop supergroup        630 2023-02-17 13:40 /user/satoaki_input/satoaki_input.txt
```

2. MapReduce Processing

Second step is the MapReduce Processing. In this stage, we would implement the MapReduce operations with saved input data. The input data would be proceeded as following figure:



Raw data
Splitting
Mapping
Shuffling
Reducing
Aggregating
Aggregated data
C002,41,Alor Setar,ZGcTmFxQ,106
C005,21,Sandakan,eQttTBJj,91
C001,32,Taiping,EPvDCGUq,443
C004,42,Seleyang Baru,VHivUTKb,343
C003,43,Petaling Jaya,DTrFegYt,267
C007,36,Kuching,kaqCtCCB,482
C003,43,Petaling Jaya,DTrFegYt,286
C001,32,Taiping,EPvDCGUq,421
C008,38,Kuala Lumpur,ieUpauKh,139
C005,21,Sandakan,eQttTBJj,162
C002,41,Alor Setar,ZGcTmFxQ,67
C006,39,Ipoh,AZEFEYLK,443
C008,38,Kuala Lumpur,ieUpauKh,328
C004,42,Seleyang Baru,VHivUTKb,264
C009,46,Sungai Petani,daTFvHwR,404
C010,55,Kota Bharu,qYfomhJn,24
C009,46,Sungai Petani,daTFvHwR,261
C007,36,Kuching,kaqCtCCB,329
C006,39,Ipoh,AZEFEYLK,56
C010,55,Kota Bharu,qYfomhJn,151
{C002,41,Alor Setar,ZGcTmFxQ}: 106
{C005,21,Sandakan,eQttTBJj}: 91
{C001,32,Taiping,EPvDCGUq}: 443
{C004,42,Seleyang Baru,VHivUTKb}: 343
{C003,43,Petaling Jaya,DTrFegYt}: 267
{C007,36,Kuching,kaqCtCCB}: 482
{C003,43,Petaling Jaya,DTrFegYt}: 286
{C001,32,Taiping,EPvDCGUq}: 421
{C008,38,Kuala Lumpur,ieUpauKh}: 139
{C005,21,Sandakan,eQttTBJj}: 162
{C002,41,Alor Setar,ZGcTmFxQ}: 67
{C006,39,Ipoh,AZEFEYLK}: 443
{C008,38,Kuala Lumpur,ieUpauKh}: 328
{C004,42,Seleyang Baru,VHivUTKb}: 264
{C009,46,Sungai Petani,daTFvHwR}: 404
{C010,55,Kota Bharu,qYfomhJn}: 24
{C009,46,Sungai Petani,daTFvHwR}: 261
{C007,36,Kuching,kaqCtCCB}: 329
{C006,39,Ipoh,AZEFEYLK}: 56
{C010,55,Kota Bharu,qYfomhJn}: 151
{C001,32,Taiping,EPvDCGUq}: 864
{C002,41,Alor Setar,ZGcTmFxQ}: 173
{C003,43,Petaling Jaya,DTrFegYt}: 553
{C004,42,Seleyang Baru,VHivUTKb}: 607
{C005,21,Sandakan,eQttTBJj}: 253
{C006,39,Ipoh,AZEFEYLK}: 499
{C007,36,Kuching,kaqCtCCB}: 811
{C008,38,Kuala Lumpur,ieUpauKh}: 467
{C009,46,Sungai Petani,daTFvHwR}: 665
{C010,55,Kota Bharu,qYfomhJn}: 175
{C001,32,Taiping,EPvDCGUq}: 864
{C002,41,Alor Setar,ZGcTmFxQ}: 173
{C003,43,Petaling Jaya,DTrFegYt}: 553
{C004,42,Seleyang Baru,VHivUTKb}: 607
{C005,21,Sandakan,eQttTBJj}: 253
{C006,39,Ipoh,AZEFEYLK}: 499
{C007,36,Kuching,kaqCtCCB}: 811
{C008,38,Kuala Lumpur,ieUpauKh}: 467
{C009,46,Sungai Petani,daTFvHwR}: 665
{C010,55,Kota Bharu,qYfomhJn}: 175

According to the aggregated data, customer C001 has paid the most, 864.

There are 20 datasets in the initial raw data file. First, the raw datasets are split into 10 splits in splitting phase and mapped in each split. Those maps are shuffled by related keys and reduced into 10 maps by summing each shuffled split's paid amount. Finally, reduced 10 rows are aggregated into single data file and aggregated data is generated.

To demonstrate these processes, we create the MostPaidPerson.java file:

```
nano MostPaidPerson.java
```

Below is the Java source code for MapReduce.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
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 MostPaidPerson {
  public static class MostPaidMapper extends Mapper<Object, Text, Text, DoubleWritable> {
    private Text person = new Text(); // set Text type key
    private DoubleWritable amount = new DoubleWritable(); // set DoubleWritable type value

    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      String[] line = value.toString().split(",");
      person.set(line[0] + "," + line[1] + "," + line[2] + "," + line[3]); // set first 4
columns as key
      amount.set(Double.parseDouble(line[4])); // set value as rightmost column which is
equivalent to paid amount
      context.write(person, amount); // emit key-value pair
    }
  }

  public static class MostPaidReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private Text maxPerson = new Text(); // set Text type maxPerson
    private DoubleWritable maxAmount = new DoubleWritable(Double.NEGATIVE_INFINITY); // set
DoubleWritable type maxAmount

    public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws
IOException, InterruptedException {
      double sum = 0.0; // initialize sum of values
      for (DoubleWritable value : values) {
        sum += value.get(); // calculate total amount for each person
      }
      if (sum > maxAmount.get()) { // if current person has the highest amount
        maxPerson.set(key); // replace the maxPerson’s key with the person’s key
        maxAmount.set(sum); // replace the maxPerson’s value with the person’s value
      }
    }

    protected void cleanup(Context context) throws IOException, InterruptedException {
      context.write(maxPerson, maxAmount); // emit the most paid person's whole 5 columns'
dataset
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "most paid person");
    job.setJarByClass(MostPaidPerson.class);
    job.setMapperClass(MostPaidMapper.class);
    job.setCombinerClass(MostPaidReducer.class); // use reducer as combiner
    job.setReducerClass(MostPaidReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(DoubleWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Save this Java file with [ctrl + o -> Enter key -> ctrl + x], then compile Java file and create Java JAR programming file with following commands:

```
javac -classpath hadoop-core-1.2.1.jar MostPaidPerson.java
jar cf most-paid-person.jar MostPaidPerson*.class
```

Finally, Run MapReduce JAR file with the following command:

```
$HADOOP_HOME/bin/hadoop jar most-paid-person.jar MostPaidPerson
/user/satoaki_input/satoaki_input.txt /user/satoaki_output
```

3. Expected Output

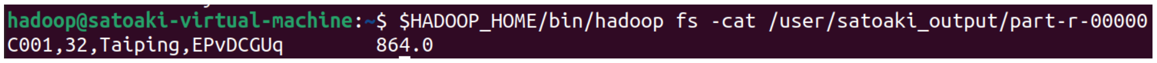
According to the raw dataset above, the 10 customers' total paid amounts are,

{C001,32,Taiping,EPvDCGUq}	864
{C002,41,Alor Setar,ZGcTmFxQ}	173
{C003,43,Petaling Jaya,DTrFegYt}	553
{C004,42,Seleyang Baru,VHivUTKb}	607
{C005,21,Sandakan,eQttTBJj}	253
{C006,39,Ipoh,AZEFEYLK}	499
{C007,36,Kuching,kaqCtCCB}	811
{C008,38,Kuala Lumpur,ieUpauKh}	467
{C009,46,Sungai Petani,daTFvHwR}	675
{C010,55,Kota Bharu,qYfomhJn}	175

Ought to the table, the customer C001 has paid the most which amount is "864", the expected output by the hypothesis. We would verify whether the calculated output is same with the expected one or not. Type in the following commands to check:

```
$HADOOP_HOME/bin/hadoop fs -cat /user/satoaki_output/part-r-00000
```

We get the following output.



```
hadoop@satoaki-virtual-machine:~$ $HADOOP_HOME/bin/hadoop fs -cat /user/satoaki_output/part-r-00000
C001,32,Taiping,EPvDCGUq        864.0
```

The calculated output is "C001 864.0", same with the expected output "C001,864". Ought to this result, we found that the MapReduce processing was successful.

Alternative tools in Hadoop

Hadoop ecosystem needs many other tools aside from MapReduce to effectively process a wide variety of big data for five reasons below:

1) One major point to discuss is that MapReduce is a batch processing system and is not suitable for all use cases. Many big data use cases require real-time or near-real-time processing, which cannot be achieved with MapReduce alone.

2) Furthermore, many big data use cases require data processing at scale, which is beyond the capabilities of a single MapReduce job. To process data at scale, distributed data processing frameworks like Apache Spark, Apache Flink, and Apache Storm are used.

3) It is also worthy to note that Hadoop ecosystem also requires tools for data storage and retrieval. Apache Hadoop provides the Hadoop Distributed File System (HDFS), which is a distributed file system for storing and processing large data sets. However, HDFS alone is not sufficient for all use cases, and additional tools such as Apache HBase, Apache Cassandra, and Apache Accumulo are needed to provide scalable, high-performance data storage and retrieval.

4) Additionally, Hadoop ecosystem also requires tools for data management, data governance, and data security. Apache Atlas, Apache Ranger, and Apache Sentry are examples of such tools that provide metadata management, access control, and data security.

5) Hadoop ecosystem also requires tools for data ingestion, data processing, and data analysis. Apache Flume, Apache Kafka, and Apache NiFi are examples of data ingestion tools that enable the collection and processing of large volumes of data. Apache Pig, Apache Hive, and Apache Impala are examples of data processing tools that provide SQL-like interfaces for data processing. Apache Mahout and Apache Spark MLlib are examples of data analysis tools that provide machine learning capabilities.

In summary, Hadoop ecosystem needs many other tools aside from MapReduce to effectively process a wide variety of big data because it requires a variety of tools for data storage, data processing, data management, and data analysis to address diverse use cases and requirements.