

1a. Jumlah data menurut KETR (LUNAS, TARIKAN) dengan Hadoop mapReduce

Tautan ke project java yang digunakan:

https://drive.google.com/drive/folders/1oUGv6a4swcdqoPnEa1tuGE_TeFUFr8uq?usp=sharing

Pertama-tama, saya menyalakan Hadoop dengan command start-all lalu membuat directory /input pada hdfs dengan script:

```
hadoop fs -mkdir /input
```

Kemudian, saya memasukan file kredit.csv ke hdfs pada directory /input dengan script:

```
hadoop fs -put Users/ghait/Documents/PDB/Tugas2PDB/input/kredit.csv /input
```

Setelah itu, untuk melakukan map dan reduce, saya perlu membuat project java yang berisi kode mapping dan reducing menggunakan library hadoop. Berikut kodenya:

WC_Mapper.java

```
public class WC_Mapper extends MapReduceBase implements
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,OutputCollector<Text,IntWritable> output,
                    Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line,",");
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

WC_Reducer.java

```
public class WC_Reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterator<IntWritable>
values,OutputCollector<Text,IntWritable> output,
                    Reporter reporter) throws IOException {
        int sum=0;
        while (values.hasNext()) {
            sum+=values.next().get();
        }
        if(key.toString().equals("LUNAS") ||
key.toString().equals("TARIKAN")){
            output.collect(key,new IntWritable(sum));
        }
    }
}
```

```

    }
}
}

```

WC_Runner.java

```

public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.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);
    }
}

```

Jangan lupa untuk menambahkan dependencies dan import yang sesuai (saya tidak mencantumkan import di kode atas karena terlalu panjang). File WC_Runner.java berfungsi sebagai kelas yang dipanggil pada saat script run JAR dijalankan. Setelah itu, buat file JAR dengan command.

```

PS C:\Users\ghait\Documents\PDB\Tugas2PDB\WordCountMapReducer> mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----< groupId:MapReduceExample >-----
[INFO] Building MapReduceExample 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]

```

Setelah build JAR berhasil, saya bisa mulai menjalankan program map reducer dengan hadoop. Scriptnya adalah:

```

hadoop jar MapReduceExample-1.0-SNAPSHOT.jar demo.WC_Runner /input
/output

```

Ghaitsa Maulidina Shofa - 2006597014

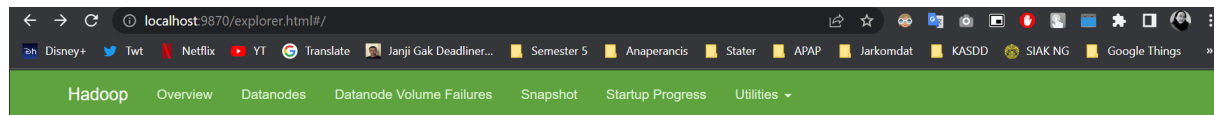
Kelas PDB

Tugas 2

Jalankan script tersebut pada directory tempat menyimpan file JAR. Berikut adalah screenshot saat menjalankan script.

```
C:\Users\ghait\Documents\PDB\Tugas2PDB\WordCountMapReducer\target>hadoop jar MapReduceExample-1.0-SNAPSHOT.jar demo.WC_Runner /input /output
2022-12-03 00:20:40,953 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-12-03 00:20:41,142 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-12-03 00:20:41,630 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2022-12-03 00:20:41,654 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ghait/.staging/job_1669999923943_0006
2022-12-03 00:20:41,974 INFO mapred.FileInputFormat: Total input files to process : 2
2022-12-03 00:20:42,073 INFO mapreduce.JobSubmitter: number of splits:3
2022-12-03 00:20:42,198 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1669999923943_0006
2022-12-03 00:20:42,198 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-12-03 00:20:42,384 INFO conf.Configuration: resource-types.xml not found
2022-12-03 00:20:42,384 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-12-03 00:20:42,463 INFO impl.YarnClientImpl: Submitted application application_1669999923943_0006
2022-12-03 00:20:42,584 INFO mapreduce.Job: The url to track the job: http://LAPTOP-O4Q72834:8088/proxy/application_1669999923943_0006/
2022-12-03 00:20:42,586 INFO mapreduce.Job: Running job: job_1669999923943_0006
2022-12-03 00:20:50,656 INFO mapreduce.Job: Job job_1669999923943_0006 running in uber mode : false
2022-12-03 00:20:50,657 INFO mapreduce.Job: map 0% reduce 0%
2022-12-03 00:20:57,801 INFO mapreduce.Job: map 33% reduce 0%
2022-12-03 00:20:58,818 INFO mapreduce.Job: map 100% reduce 0%
2022-12-03 00:21:03,890 INFO mapreduce.Job: map 100% reduce 100%
2022-12-03 00:21:03,899 INFO mapreduce.Job: Job job_1669999923943_0006 completed successfully
2022-12-03 00:21:04,013 INFO mapreduce.Job: Counters: 55
  File System Counters
    FILE: Number of bytes read=58
    FILE: Number of bytes written=1061945
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=80679
```

Setelah script selesai dijalankan tanpa error, saya bisa cek file output hasil map reduce pada <http://localhost:9870/explorer.html#/> kemudian ke file output.



Browse Directory

/

Go!

Show

25

entries

Search:

<input type="checkbox"/>	<div><div></div></div> Permission	<div><div></div></div> Owner	<div><div></div></div> Group	<div><div></div></div> Size	<div><div></div></div> Last Modified	<div><div></div></div> Replication	<div><div></div></div> Block Size	<div><div></div></div> Name	<div><div></div></div>
<input type="checkbox"/>	drwxr-xr-x	ghait	supergroup	0 B	Dec 02 23:28	0	0 B	code	<div><div></div></div>
<input type="checkbox"/>	drwxr-xr-x	ghait	supergroup	0 B	Dec 02 21:44	0	0 B	input	<div><div></div></div>
<input type="checkbox"/>	drwxr-xr-x	ghait	supergroup	0 B	Dec 03 00:21	0	0 B	output	<div><div></div></div>
<input type="checkbox"/>	drwx-----	ghait	supergroup	0 B	Dec 02 01:37	0	0 B	tmp	<div><div></div></div>

Showing 1 to 4 of 4 entries

Previous

1

Next

Browse Directory

/output

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	ghait	supergroup	0 B	Dec 03 00:21	1	128 MB	_SUCCESS
-rw-r--r--	ghait	supergroup	23 B	Dec 03 00:21	1	128 MB	part-00000

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2020.

Dengan menekan “Head the file (first 32K)”, saya bisa langsung melihat isi dari output.

File information - part-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information — Block 0

Block ID: 1073741892
Block Pool ID: BP-164868171-169.254.5.11-1669878121641
Generation Stamp: 1068
Size: 23
Availability:
• LAPTOP-O4Q72834.mshome.net

File contents

LUNAS 1876
TARIKAN 113

Terlihat bahwa data kredit “Lunas” berjumlah 1876 dan “Tarikan” berjumlah 113.

1b. Jumlah data menurut KETR (LUNAS, TARIKAN) dengan Pyspark

Pertama, panggil file yang ingin diakses dan simpan di variabel df. Setelah itu, lakukan pengelompokan dengan groupBy(“STATUS”) dan hitung jumlah data tiap barisnya dengan count(). Terakhir, tampilkan hasil dengan show(). Berikut adalah script yang sudah berhasil dijalankan.

```
scala> df.groupBy(" STATUS").count().show()
+-----+-----+
| STATUS|count|
+-----+-----+
|  LUNAS| 1876|
|TARIKAN|  113|
+-----+-----+
```

2a. Rata-rata SALARY terkelompok menurut KETR dengan Hadoop

Tautan ke file project java yang digunakan:

https://drive.google.com/drive/folders/1K1apNWnzGm_MNQyHpvgPMEueHK2H3nYI?usp=share_link

Untuk dapat menghitung rata-rata berdasarkan STATUS (lunas, tarikan), kita perlu memisahkan header dari file csv agar tidak terjadi error pada program penghitungan rata-ratanya. Oleh karena itu, saya mengedit file kredit.csv dengan menghapus baris pertamanya dan menyimpannya dengan nama kreditNoHeader.csv. Kemudian, saya memasukan file tersebut ke hdfs directory /input dengan script:

```
hadoop fs -put
Users/ghait/Documents/PDB/Tugas2PDB/input/kreditNoHeader.csv /input
```

Lalu, buat program menghitung rata-ratanya. Berikut adalah codenya:

```
public class SalaryAverage {

    //Driver Class
    public static void main(String[] args) throws Exception {
        //set up configurations
        Configuration c = new Configuration();
        String[] files = new GenericOptionsParser(c,
args).getRemainingArgs();
        Path input = new Path(files[0]);
        Path output = new Path(files[1]);
        Job j = new Job(c, "SalaryAverage");
        j.setJarByClass(SalaryAverage.class);
        j.setMapperClass(MapForAverage.class);
        j.setReducerClass(ReduceForAverage.class);
        j.setOutputKeyClass(Text.class);
        j.setOutputValueClass(FloatWritable.class);

        //get input paths from arguments
        FileInputFormat.addInputPath(j, input);
        FileOutputFormat.setOutputPath(j, output);
```

```
//start measuring the start time.
long startTime = System.currentTimeMillis();
j.waitForCompletion(true);
long estimatedTime = System.currentTimeMillis() - startTime;
System.out.println("Time Elapsed : " + estimatedTime);
System.exit(0);
}

//Mapper
public static class MapForAverage extends Mapper<LongWritable, Text,
Text, FloatWritable> {

    public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String line = value.toString();
        String[] words = line.split(",");
        Text outputKey = new Text(words[6].toUpperCase().trim());
        FloatWritable outputValue = new
FloatWritable(Float.parseFloat(words[1]));
        context.write(outputKey, outputValue);
    }
}

//Reducer
public static class ReduceForAverage extends Reducer<Text,
FloatWritable, Text, FloatWritable> {

    public void reduce(Text word, Iterable<FloatWritable> values,
Context context) throws IOException, InterruptedException {
        float sum = 0;
        float count = 0;
        for (FloatWritable value : values) {
            sum += value.get();
            count = count + 1;
        }

        float average = sum / count;
        context.write(word, new FloatWritable(average));
    }
}
}
```

Tidak lupa untuk menambahkan dependency yang sesuai dan import yang dibutuhkan. Setelah itu, build JAR dengan command:

```
PS C:\Users\ghait\Documents\PDB\Tugas2PDB\SalaryAverage> mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----< groupId:SalaryAverage >-----
[INFO] Building SalaryAverage 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
```

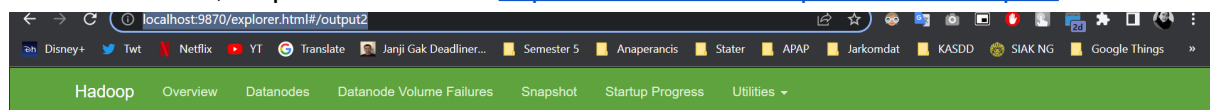
Setelah JAR terbuat, run program melalui hadoop dengan script:

```
hadoop jar SalaryAverage-1.0-SNAPSHOT.jar demo.SalaryAverage
/input/kreditNoHeader.csv /output2
```

Jalankan script tersebut pada directory tempat menyimpan file JAR. Berikut adalah screenshot saat menjalankan script.

```
C:\Users\ghait\Documents\PDB\Tugas2PDB\SalaryAverage\target>hadoop jar SalaryAverage-1.0-SNAPSHOT.jar demo.SalaryAverage
/input/kreditNoHeader.csv /output2
2022-12-03 11:29:01,327 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2022-12-03 11:29:02,515 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/
ghait/.staging/job_1670041324844_0001
2022-12-03 11:29:03,584 INFO input.FileInputFormat: Total input files to process : 1
2022-12-03 11:29:03,778 INFO mapreduce.JobSubmitter: number of splits:1
2022-12-03 11:29:03,983 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1670041324844_0001
2022-12-03 11:29:03,983 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-12-03 11:29:04,370 INFO conf.Configuration: resource-types.xml not found
2022-12-03 11:29:04,371 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-12-03 11:29:04,970 INFO impl.YarnClientImpl: Submitted application application_1670041324844_0001
2022-12-03 11:29:05,076 INFO mapreduce.Job: The url to track the job: http://LAPTOP-04Q72834:8088/proxy/application_1670
041324844_0001/
2022-12-03 11:29:05,078 INFO mapreduce.Job: Running job: job_1670041324844_0001
2022-12-03 11:29:17,377 INFO mapreduce.Job: Job job_1670041324844_0001 running in uber mode : false
2022-12-03 11:29:17,384 INFO mapreduce.Job: map 0% reduce 0%
2022-12-03 11:29:25,578 INFO mapreduce.Job: map 100% reduce 0%
2022-12-03 11:29:33,701 INFO mapreduce.Job: map 100% reduce 100%
2022-12-03 11:29:34,727 INFO mapreduce.Job: Job job_1670041324844_0001 completed successfully
2022-12-03 11:29:34,913 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=24100
  FILE: Number of bytes written=578461
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=76354
  HDFS: Number of bytes written=34
  HDFS: Number of read operations=8
```





Setelah berhasil, output bisa dilihat di <http://localhost:9870/explorer.html#/output2>.



Browse Directory

/output2

Go!














Show

25

 entries

Search:

<input type="checkbox"/>	 Permission	 Owner	 Group	 Size	 Last Modified	 Replication	 Block Size	 Name	
<input type="checkbox"/>	-rw-r--r--	ghait	supergroup	0 B	Dec 03 11:29	1	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	ghait	supergroup	34 B	Dec 03 11:29	1	128 MB	part-r-00000	

Showing 1 to 2 of 2 entries

Previous

1

Next

Showing 1 to 2 of 2 entries

Previous 1 Next

Untuk mengakses isi dari output, klik yang ada pada kotak merah di atas, lalu pilih "Head the file (first 32K)"

File information - part-r-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 0 ▾

Block ID: 1073742023

Block Pool ID: BP-164868171-169.254.5.11-1669878121641

Generation Stamp: 1199

Size: 34

Availability:

- LAPTOP-O4Q72834.mshome.net

File contents

```
LUNAS 1739077.6
TARIKAN 1667492.2
```

Terlihat pada file contents bahwa kita sudah berhasil menghitung rata-rata SALARY terkelompok menurut STATUS lunas dan tarikan. Rata-rata SALARY dengan STATUS lunas adalah 1739077,6 dan dengan STATUS tarikan adalah 1667492,2.

2b. Rata-rata SALARY terkelompok menurut KETR dengan Pyspark

Pertama, panggil file yang ingin diakses dan simpan di variabel df. Lalu, setelah dicek tipe datanya, ternyata kolom SALARY terbaca sebagai string yang mana seharusnya adalah numerik (integer, long, dsb). Oleh karena itu, saya ubah tipe datanya dengan selectExpr dan mengcast tipe data SALARY sebagai integer sekaligus mengambil kolom STATUS.

Perubahan direassign ke variabel yang sama yaitu df.

Setelah itu, lakukan pengelompokan dengan groupBy("STATUS") dan hitung rata-rata SALARY dengan avg("SALARY"). Terakhir, tampilkan hasil dengan show(). Berikut adalah script yang sudah berhasil dijalankan.


```
scala> var df = spark.read.option("header", "true").csv("C://Users/ghait/Documents/PDB/Tugas2PDB/input/kredit.csv")
df: org.apache.spark.sql.DataFrame = [OCCUPATION: string, SALARY: string ... 5 more fields]

scala> df = df.selectExpr("cast(SALARY as int) SALARY", "STATUS")
df: org.apache.spark.sql.DataFrame = [SALARY: int, STATUS: string]

scala> df.groupBy("STATUS").avg("SALARY").show()
+-----+-----+
| STATUS|      avg(SALARY)|
+-----+-----+
|  LUNAS|1739076.7611940298|
|TARIKAN|1667492.5309734512|
+-----+-----+
```

Rata-rata SALARY dengan STATUS lunas adalah 1739077,76 dan dengan STATUS tarikan adalah 1667492,53.

3a. Lakukanlah klasifikasi Naive Bayes dengan menggunakan pyspark.ml

Berikut adalah hasil klasifikasi menggunakan Naive Bayes dengan pyspark.ml. Klasifikasi menggunakan label STATUS dengan fitur OCCUPATION dan MEREK. Tautan di bawah adalah kode dan jawaban soal ini.

https://colab.research.google.com/drive/1YiCdDn05x0aOoQG_2ultKKjL04ufvPh0?authuser=1

Accuracy model Naive Bayes nomor ini adalah sekitar 0.94

3b. Lakukanlah klasifikasi Linear Support Vector Machine dengan menggunakan pyspark.ml

Berikut adalah hasil klasifikasi menggunakan Linear Support Vector Machine dengan pyspark.ml. Klasifikasi menggunakan label STATUS dengan fitur OCCUPATION dan MEREK. Tautan di bawah adalah kode dan jawaban soal ini.

https://colab.research.google.com/drive/1pURa61PXv_kvngVdW27BwUQl89qYAvNt?usp=sharing

Accuracy model Linear Support Vector Machine nomor ini adalah sekitar 0.94