

Author

LAB ID-8

1. Nikita Goyal Class ID -9
2. Maseerah Muradabadi Class ID- 20
3. Eric Chadwick Class ID - 2

Introduction

This assignment teaches us some practical problem which include the concept of MapReduce, Hive queries and Apache solr.

Objective

To learn and implement the problems provided in the assignment which improves our logical thinking and get familiar with the Big Data Programming Concepts.

Features

We have used the following features:

1. Eclipse

2. cloudera Terminal

3. Apache solr

Datasets used

1. Zomato Dataset
2. SuperHero Dataset

Questions

1. Hadoop MapReduce Algorithm

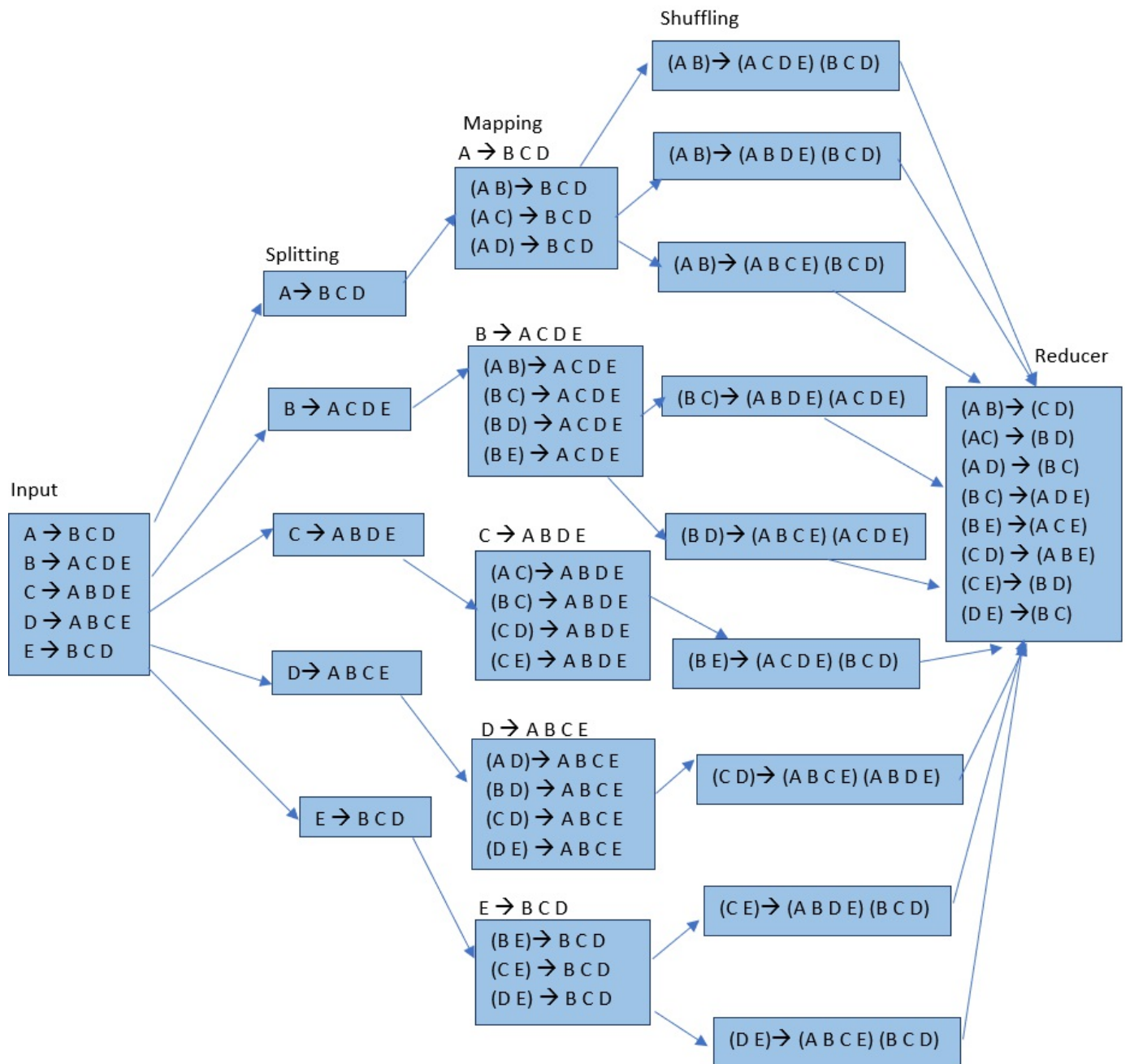
Implement MapReduce algorithm for finding Facebook common friends problem and run the MapReduce job on Apache Hadoop. Show your implementation through map-reduce diagram as shown in Lesson Plan 2:

<https://umkc.box.com/s/jhpgd8yeerlkurqwtgp5ej5izpt47lk>

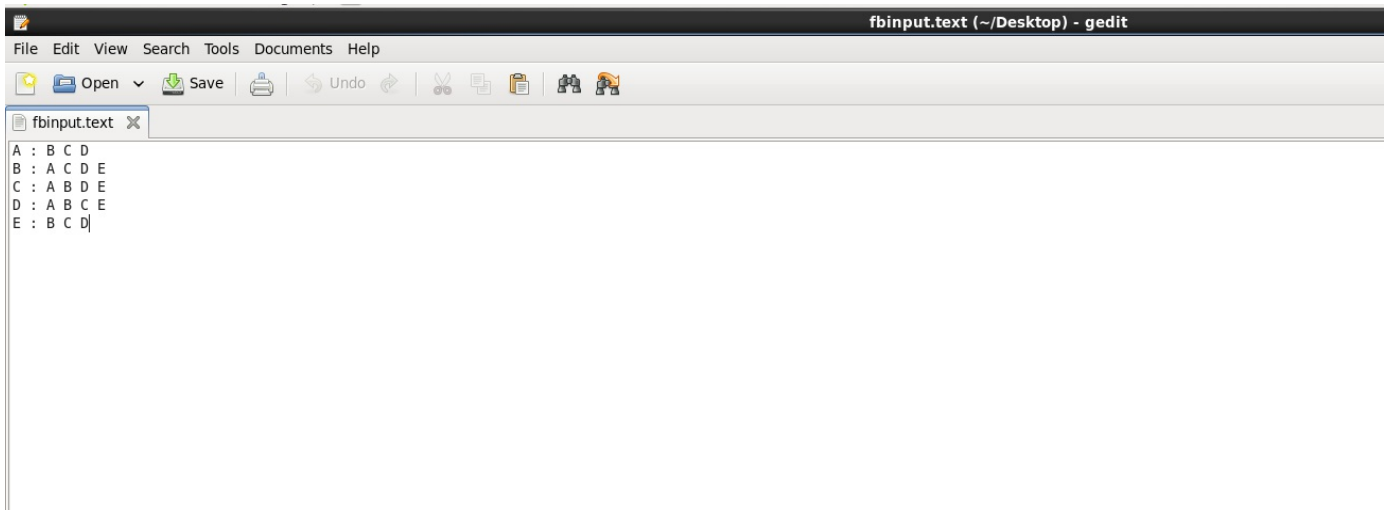
Write a report including your algorithm and result screenshots.

Solution-1

Workflow diagram



Input

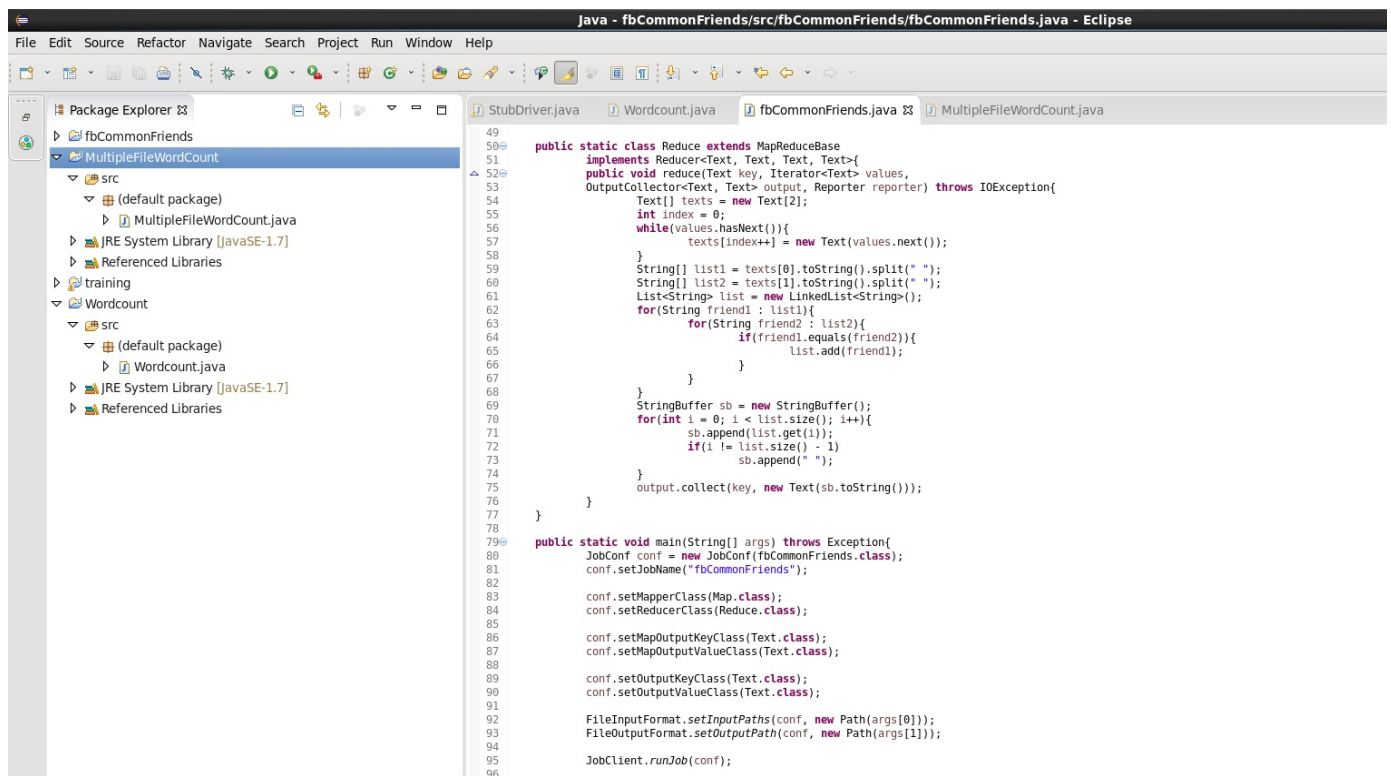


Map Algorithm

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class fbCommonFriends {

    public static class Map extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text>{
        public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter)
            throws IOException{
            StringTokenizer tokenizer = new StringTokenizer(value.toString(), "\n");
            String line = null;
            String[] lineArray = null;
            String[] friendArray = null;
            String[] tempArray = null;
            while(tokenizer.hasMoreTokens()){
                line = tokenizer.nextToken();
                lineArray = line.split(" : ");
                friendArray = lineArray[1].split(" ");
                tempArray = new String[2];
                for(int i = 0; i < friendArray.length; i++){
                    tempArray[0] = friendArray[i];
                    tempArray[1] = lineArray[0];
                    Arrays.sort(tempArray);
                    output.collect(new Text(tempArray[0] + " " + tempArray[1]), new Text(lineArray[1]));
                }
            }
        }
    }
}
```

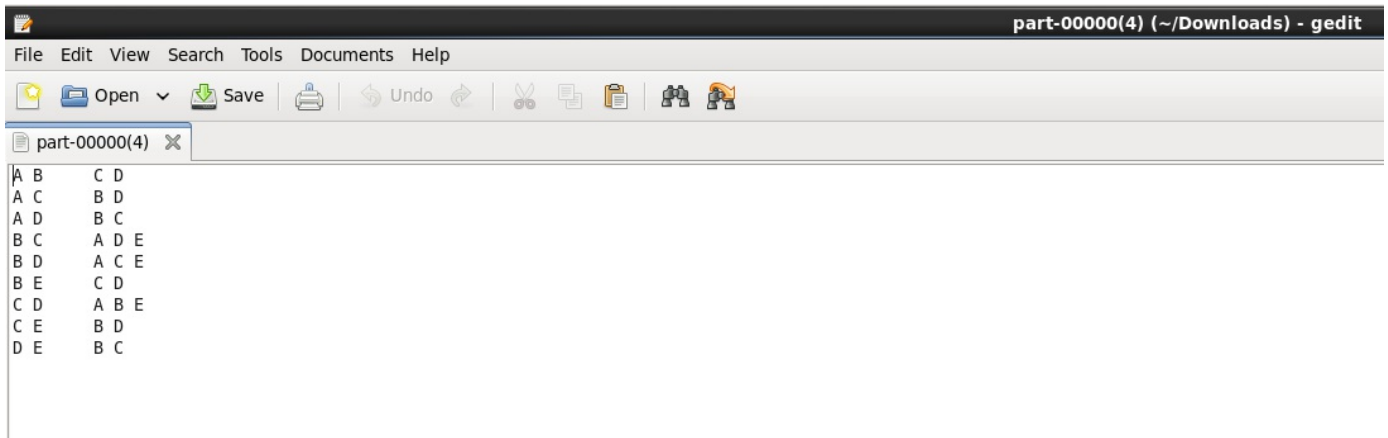
Reduce Algorithm



Running the Program

```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/fbCommonFriends.jar fbCommonFriends.fbCommonFriends /LABinput/ /output40
19/06/22 07:52:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/06/22 07:52:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/06/22 07:52:18 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
19/06/22 07:52:19 INFO mapred.FileInputFormat: Total input paths to process : 1
19/06/22 07:52:19 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1281)
    at java.lang.Thread.join(Thread.java:1355)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
19/06/22 07:52:19 INFO mapreduce.JobSubmitter: number of splits:2
19/06/22 07:52:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1561207392908_0006
19/06/22 07:52:20 INFO impl.YarnClientImpl: Submitted application application_1561207392908_0006
19/06/22 07:52:20 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1561207392908_0006/
19/06/22 07:52:20 INFO mapreduce.Job: Running job: job_1561207392908_0006
19/06/22 07:52:31 INFO mapreduce.Job: Job job_1561207392908_0006 running in uber mode : false
19/06/22 07:52:31 INFO mapreduce.Job: map 0% reduce 0%
19/06/22 07:52:49 INFO mapreduce.Job: map 50% reduce 0%
19/06/22 07:52:50 INFO mapreduce.Job: map 100% reduce 0%
19/06/22 07:52:59 INFO mapreduce.Job: map 100% reduce 100%
19/06/22 07:52:59 INFO mapreduce.Job: Job job_1561207392908_0006 completed successfully
19/06/22 07:52:59 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=246
  FILE: Number of bytes written=431126
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=270
  HDFS: Number of bytes written=78
  HDFS: Number of read operations=9
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=29280
  Total time spent by all reduces in occupied slots (ms)=8297
  Total time spent by all map tasks (ms)=29280
  Total time spent by all reduce tasks (ms)=8297
  Total vcore-milliseconds taken by all map tasks=29280
  Total vcore-milliseconds taken by all reduce tasks=8297
```

Output

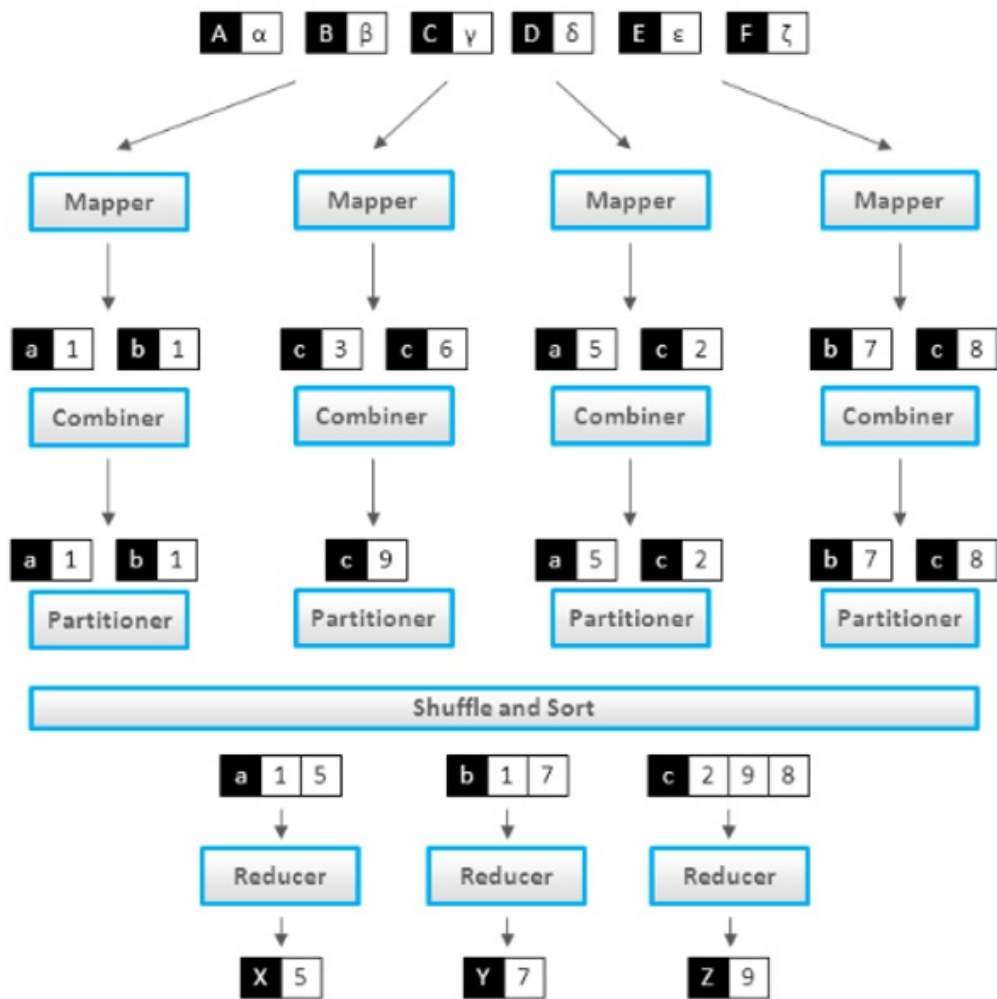


2.Use Case: Counting and Summing

There is a number of documents where each document is a set of terms. It is required to calculate a total number of occurrences of each term in all documents. Alternatively, it can be an arbitrary function of the terms. For instance, there is a log file where each record contains a response time and it is required to calculate an average response time.

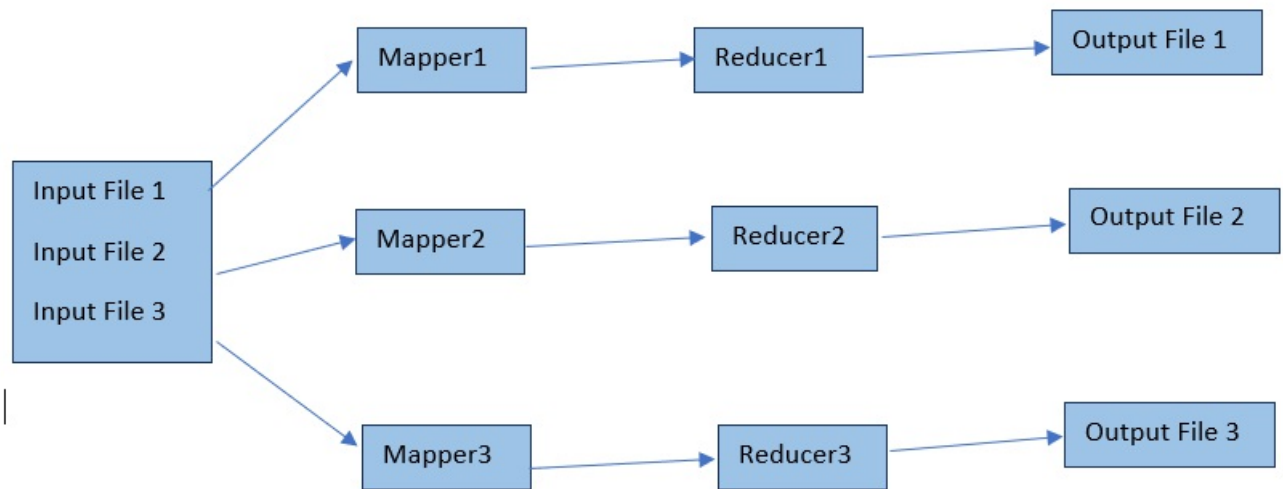
For reference, use the image above and show your implementation through map-reduce diagram as shown in Lesson Plan 2:

<https://umkc.box.com/s/jhpgd8yeerlkurqwjtgp5ej5izpt47lk>



Solution-2

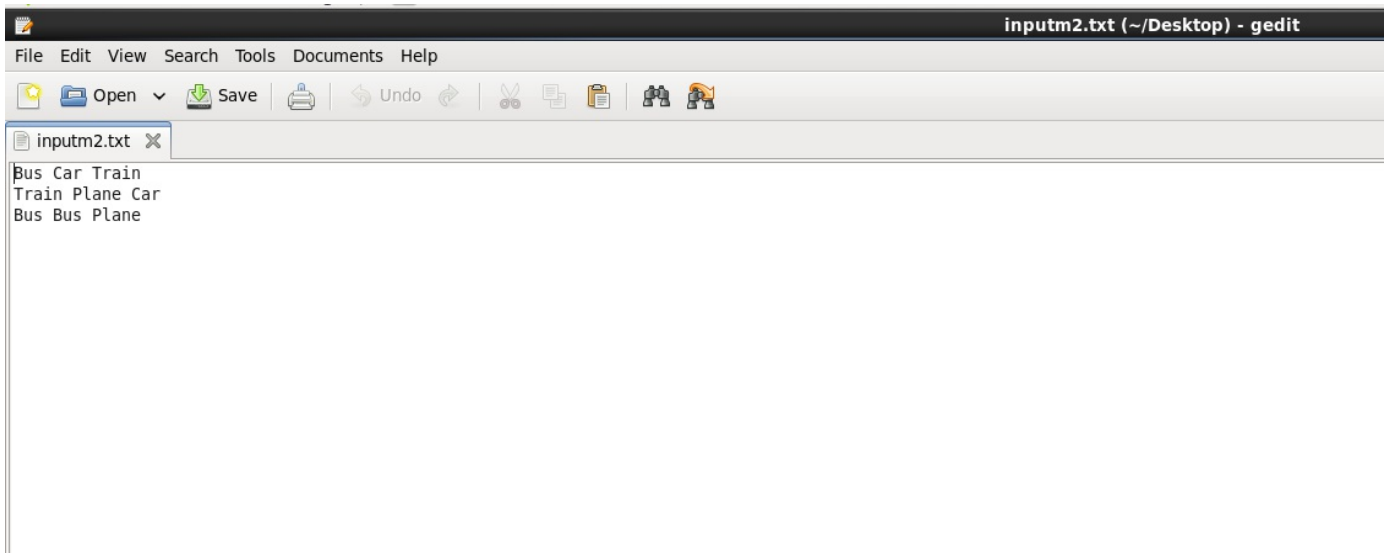
Workflow Diagram



Input File-1

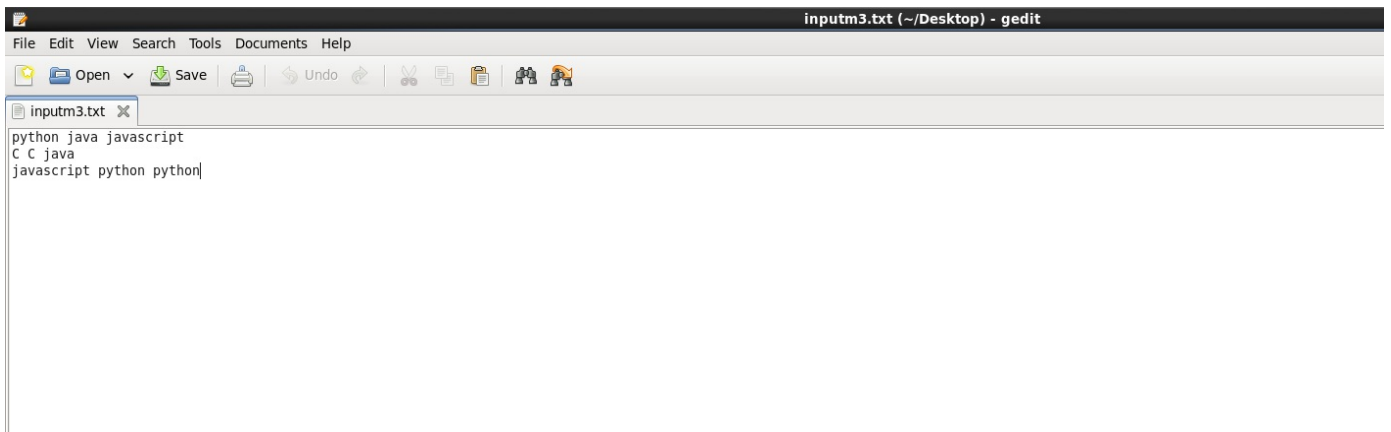
```
inputm1.txt (~/Desktop) - gedit
File Edit View Search Tools Documents Help
[Icons: Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, Help]
inputm1.txt x
Deer Bear River
Car Car River
Deer Car Bear|
```

Input File-2



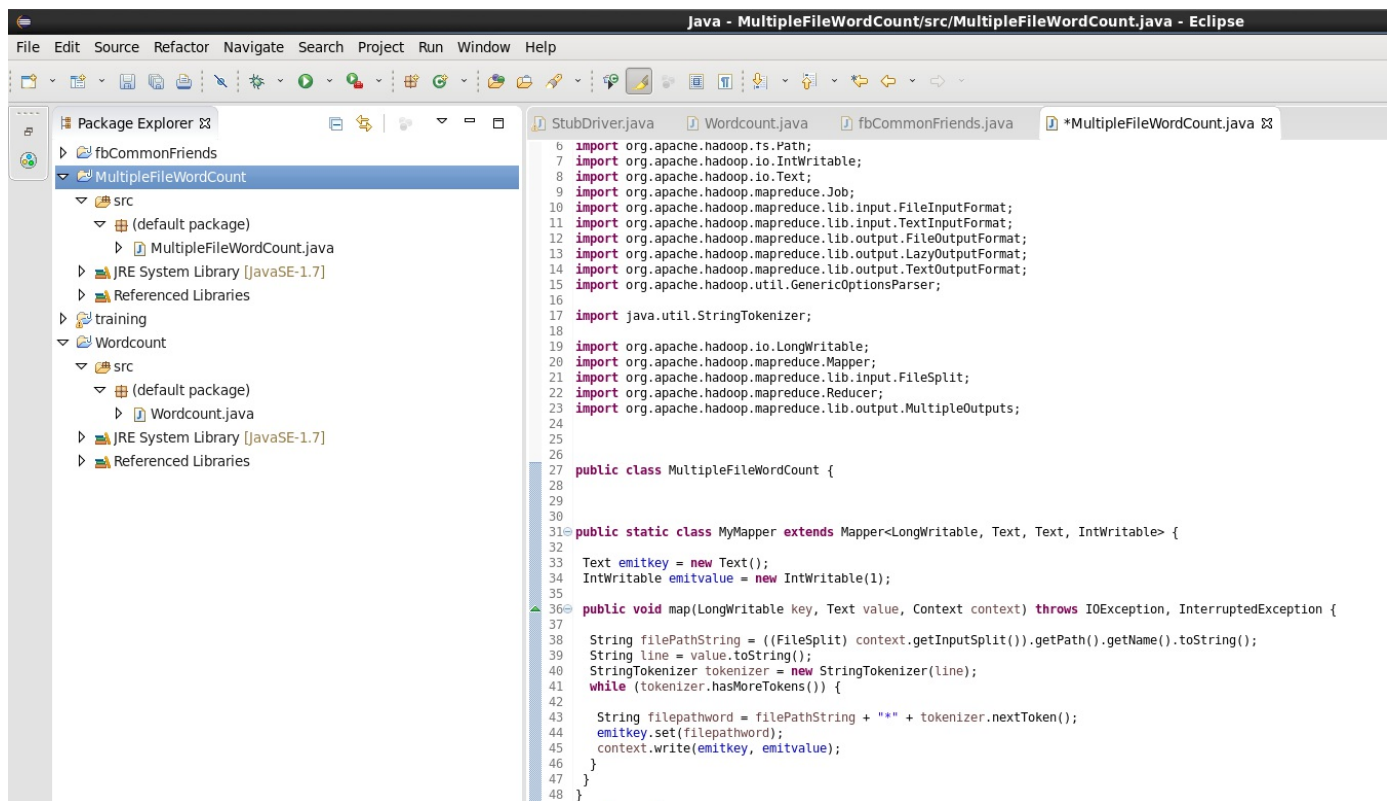
```
inputm2.txt X
Bus Car Train
Train Plane Car
Bus Bus Plane
```

Input File -3



```
inputm3.txt X
python java javascript
C C java
javascript python python
```

Mapper Algorithm



Reducer Algorithm

```

public static class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    Text emitkey = new Text();
    IntWritable emitvalue = new IntWritable();
    private MultipleOutputs<Text, IntWritable> multipleoutputs;

    public void setup(Context context) throws IOException, InterruptedException {
        multipleoutputs = new MultipleOutputs<Text, IntWritable>(context);
    }

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
        int sum = 0;

        for (IntWritable value : values) {
            sum = sum + value.get();
        }
        String pathandword = key.toString();
        String[] splitted = pathandword.split("\\*");
        String path = splitted[0];
        String word = splitted[1];
        emitkey.set(word);
        emitvalue.set(sum);
        System.out.println("word:" + word + "\t" + "sum:" + sum + "\t" + "path: " + path);
        multipleoutputs.write(emitkey, emitvalue, ("/NewOut/"+path));
    }

    public void cleanup(Context context) throws IOException, InterruptedException {
        multipleoutputs.close();
    }
}

```

```

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
    Configuration conf = new Configuration();
    @SuppressWarnings("deprecation")
    Job myJob = new Job(conf, "Multiwordcount");
    args = new GenericOptionsParser(conf, args).getRemainingArgs();
    FileSystem fs = FileSystem.get(new Configuration());
    fs.delete(new Path("/NewOut/"), true);

    myJob.setJarByClass(MultipleFileWordCount.class);
    myJob.setMapperClass(MyMapper.class);
    myJob.setReducerClass(MyReducer.class);
    myJob.setMapOutputKeyClass(Text.class);
    myJob.setMapOutputValueClass(IntWritable.class);
    // myJob.setNumReduceTasks(0);
    myJob.setOutputKeyClass(Text.class);
    myJob.setOutputValueClass(IntWritable.class);
    LazyOutputFormat.setOutputFormatClass(myJob, TextOutputFormat.class);

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

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

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

```

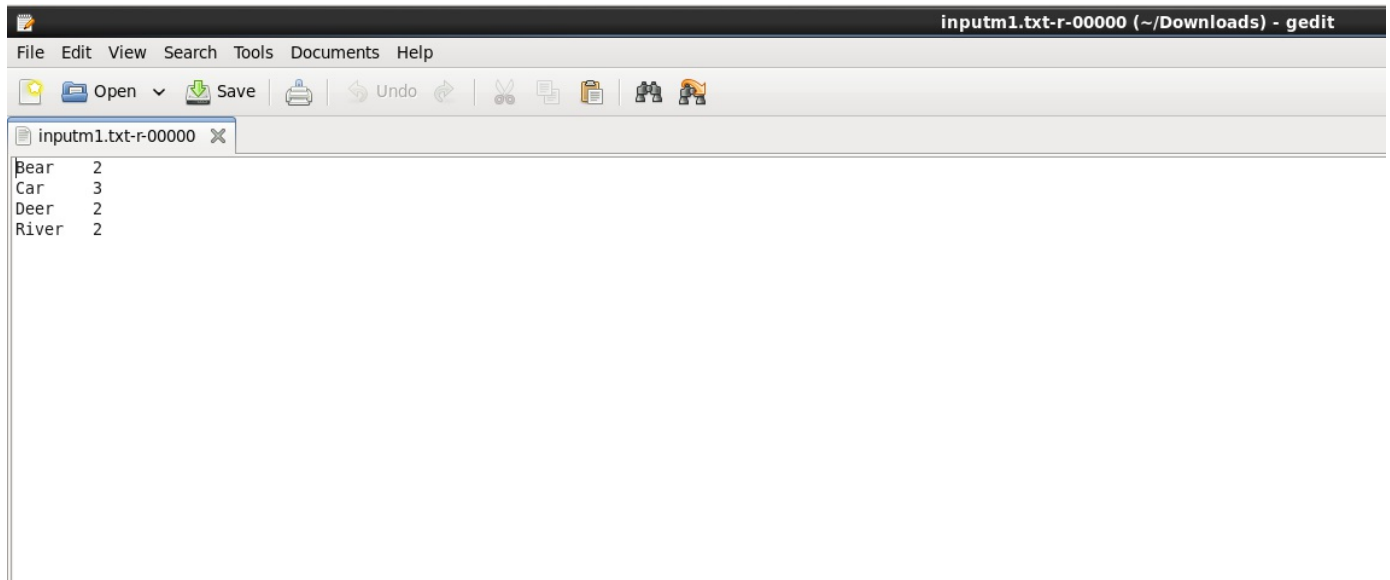
Running the Program

```

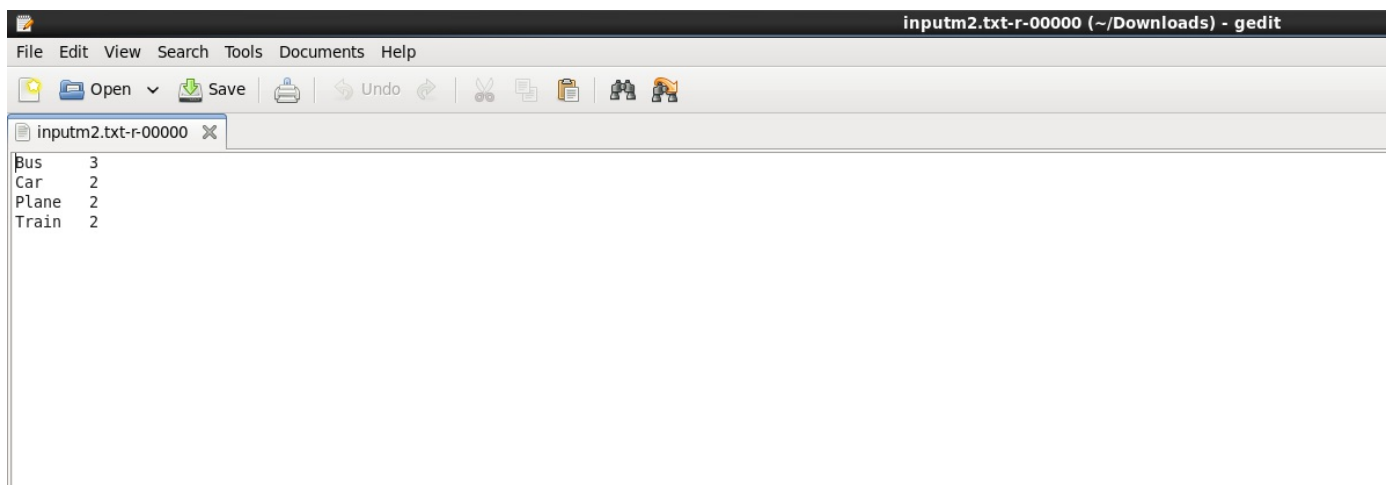
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/MultipleFileWordCount.jar MultipleFileWordCount /Input /output39
19/06/22 07:45:21 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/06/22 07:45:21 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
19/06/22 07:45:22 INFO input.FileInputFormat: Total input paths to process : 5
19/06/22 07:45:22 INFO mapreduce.JobSubmitter: number of splits:5
19/06/22 07:45:23 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1561207392908_0005
19/06/22 07:45:23 INFO impl.YarnClientImpl: Submitted application application_1561207392908_0005
19/06/22 07:45:23 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1561207392908_0005/
19/06/22 07:45:23 INFO mapreduce.Job: Running job: job_1561207392908_0005
19/06/22 07:45:36 INFO mapreduce.Job: Job job_1561207392908_0005 running in uber mode : false
19/06/22 07:45:36 INFO mapreduce.Job:  map 0% reduce 0%
19/06/22 07:46:15 INFO mapreduce.Job:  map 20% reduce 0%
19/06/22 07:46:18 INFO mapreduce.Job:  map 40% reduce 0%
19/06/22 07:46:20 INFO mapreduce.Job:  map 60% reduce 0%
19/06/22 07:46:21 INFO mapreduce.Job:  map 100% reduce 0%
19/06/22 07:46:30 INFO mapreduce.Job:  map 100% reduce 100%
19/06/22 07:46:30 INFO mapreduce.Job: Job job_1561207392908_0005 completed successfully
19/06/22 07:46:30 INFO mapreduce.Job: Counters: 51
File System Counters
  FILE: Number of bytes read=816
  FILE: Number of bytes written=866975
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=757
  HDFS: Number of bytes written=117
  HDFS: Number of read operations=18
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=6
Job Counters
  Killed map tasks=1
  Launched map tasks=5
  Launched reduce tasks=1
  Other local map tasks=1
  Data-local map tasks=4
  Total time spent by all maps in occupied slots (ms)=195043
  Total time spent by all reduces in occupied slots (ms)=12928
  Total time spent by all map tasks (ms)=195043
  Total time spent by all reduce tasks (ms)=12928
  Total vcore-milliseconds taken by all map tasks=195043
  Total vcore-milliseconds taken by all reduce tasks=12928
  Total megabyte-milliseconds taken by all map tasks=199724032
  Total megabyte-milliseconds taken by all reduce tasks=13238272
Map-Reduce Framework
  Map input records=12
  Map output records=36

```

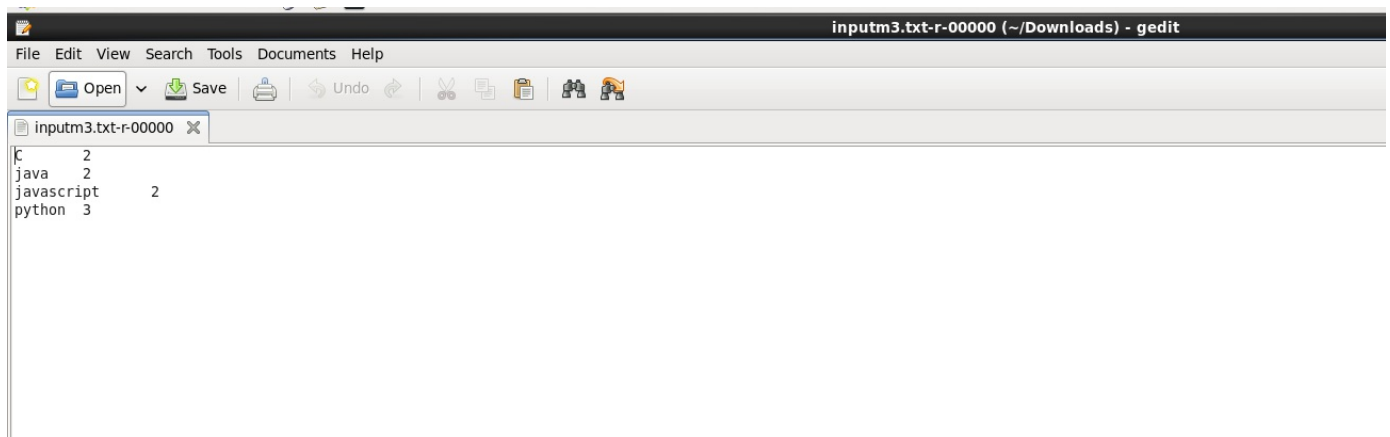
Ouptut Files



```
inputm1.txt-r-00000 X
Bear 2
Car 3
Deer 2
River 2
```



```
inputm2.txt-r-00000 X
Bus 3
Car 2
Plane 2
Train 2
```



```
inputm3.txt-r-00000 X
c 2
java 2
javascript 2
python 3
```

Hive Queries

Q3 Consider one of the following use cases,

A. Zomato Restaurants Data

<https://www.kaggle.com/shrutimehta/zomato-restaurants-data>

B. Super Heros Dataset

<https://www.kaggle.com/clauidodavi/superhero-set/data>

C. Google Job Skills Dataset

<https://www.kaggle.com/niyamatalmass/google-job-skills/data>

D. Seinfeld Chronicles Dataset

<https://www.kaggle.com/thec03u5/seinfeld-chronicles/data>

**** HIVE USECASE****

a. Create a Hive Table including Complex Data Types

b. Use built-in functions in your queries

c. Perform 10 intuitive questions in Dataset (e.g.: pattern recognition, topic discussion, most important terms, etc.). Use your innovation to think out of box

Solution-3

We have used a zomato restaurant data

Query -1 Display 6 records for online booking with restaurant id and address

Query-2 Display country code 14 restaurant id and name sort it by ascending order

Query-3 Display Country code for value Davenport

Query-4 To display the address for city

Des moines

Query -5 Display table information for country code 14

Query-6 Display all records for restaurant Shirley Display

Query-7 Show 5 records for price range greater than 3.

Query-8 New table

Query-8.1 Join operation

Query-9 Show restaurant name in city Boise

Query-10 Display records where average cost for 2 is more than 25 and

limit the output to 2

```

```

Q4 SOLR USECASE

- a. Create a Solr Collection including our own Field Types
- b. Perform 10 intuitive questions in Dataset (e.g.: pattern recognition, topic discussion, most important terms, etc.). Use your innovation to think out of box. Implement at least 5 nested queries among the 10.
- c. Record the time execution for the queries.

Write a report including your algorithm and result screenshots.

Solution-4

Configs:

Modify films schema:

1. gedit /tmp/films/conf/schema.xml

2. add these lines:

```
<field name="Gender" type="string" indexed="true"
stored="true" />
<field name="Eye color" type="string" indexed="true"
stored="true" />
<field name="Race" type="string" indexed="true"
stored="true" />
<field name="Hair color" type="string" indexed="true"
stored="true" />
<field name="Height" type="double" indexed="true"
stored="true" />
<field name="Publisher" type="string" indexed="true"
stored="true" />
<field name="Skin color" type="string" indexed="true"
stored="true" />
<field name="Alignment" type="string" indexed="true"
stored="true" />
<field name="Weight" type="double" indexed="true"
stored="true" />
```

3. Delete any fields which duplicate these names

4. Create instancedir:

```
solr instancedir --create superheros /tmp/films
```

5. Create collection:

solr collection --create superheros

6. Copy/Paste CSV in Solr UI

7. Query away

We have used superhero dataset

Non-nested Queries

Solr1 - returns name, Gender, Race and BMI (Body Mass Index) for each character. BMI is calculated as $\text{weight}/\text{height}^2$ for metric. The 4th param of fl computes that function.

Request-Handler (qt)

/select

common

q

fq

sort

start, rows

010

fl

e, div(Weight, div(product(Height,Height),100))

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

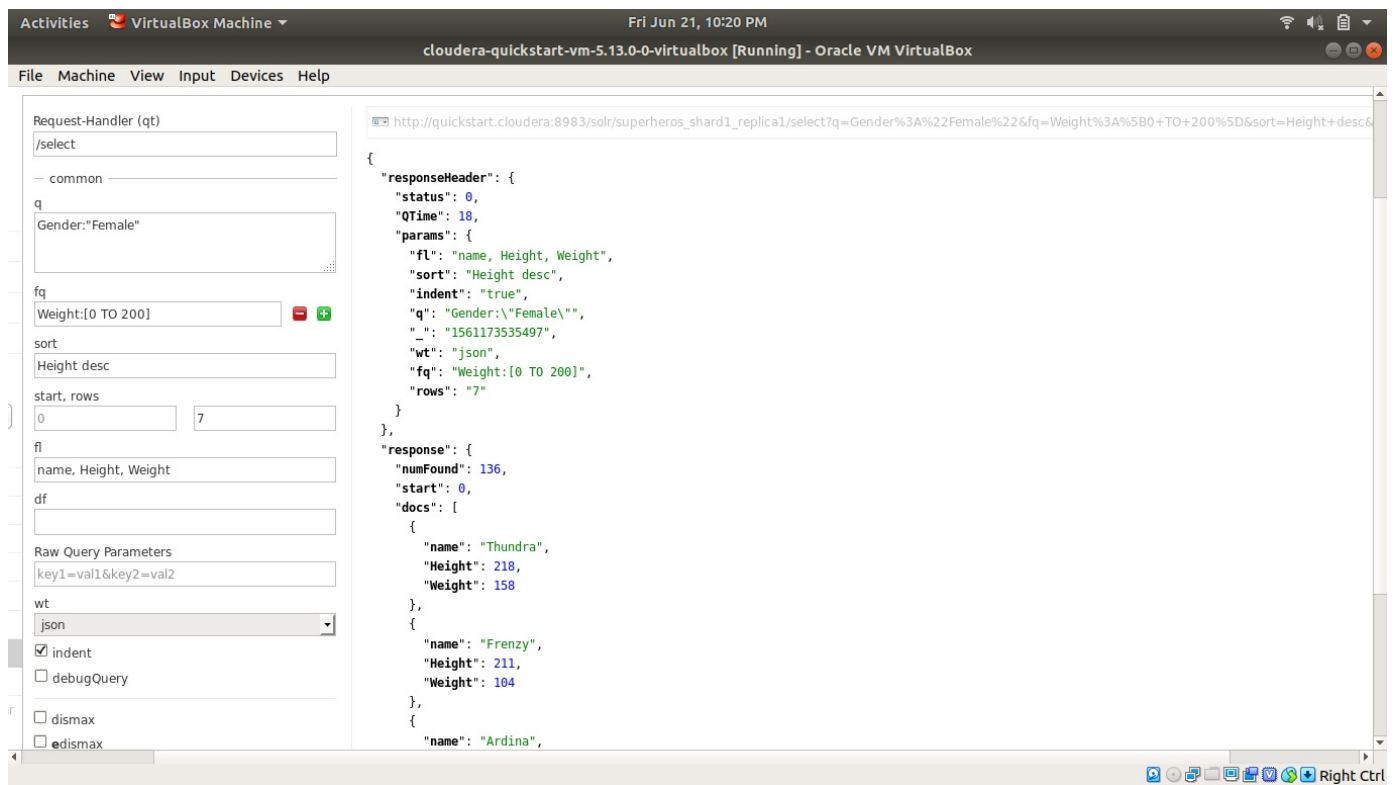
Execute Query

olr/#/superheros_shard1_replica1/quer

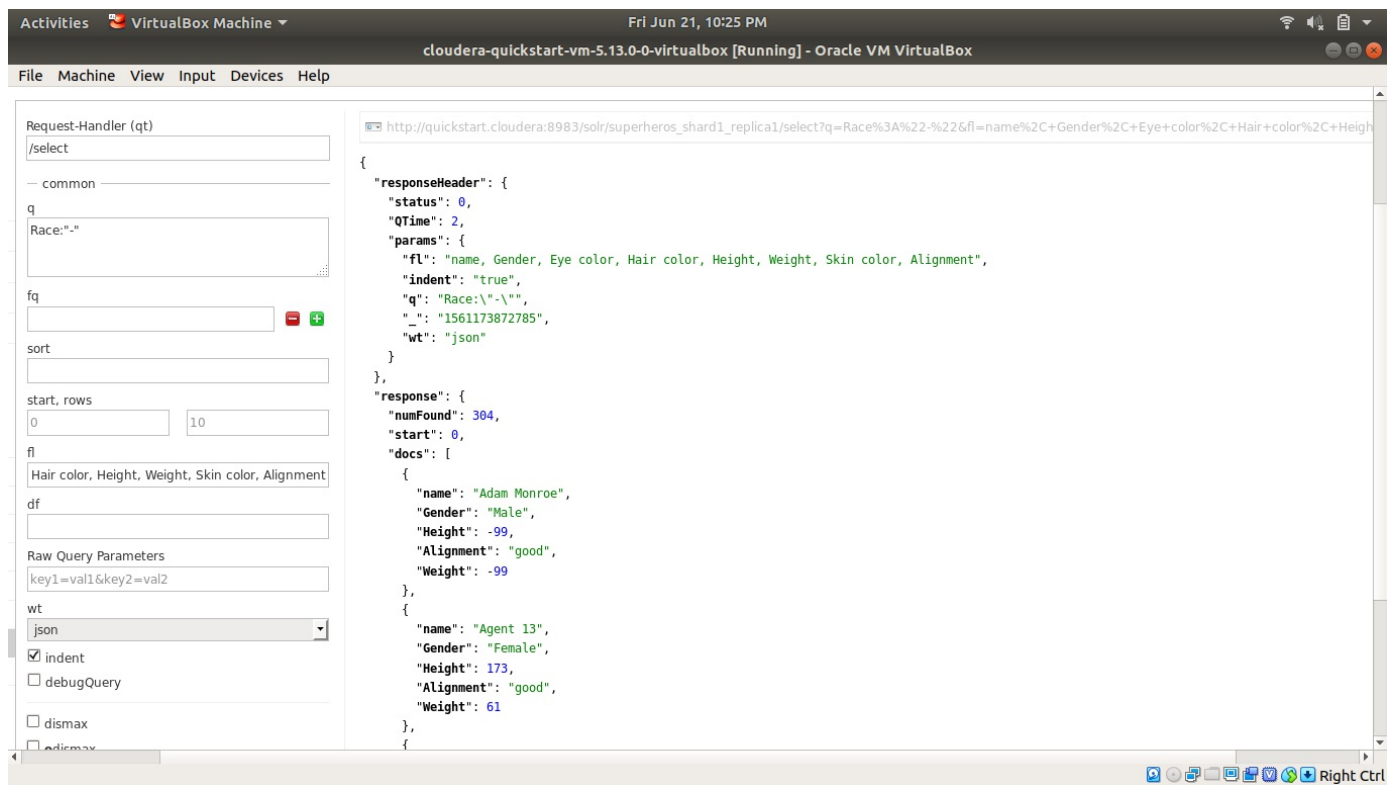
http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=*&fl=name%2C+Gender%2C+Race%2C+div(Weight%2C+div(product(Height%2CHeight)%2C100

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 3,
    "params": {
      "fl": "name, Gender, Race, div(Weight, div(product(Height,Height),100))",
      "indent": "true",
      "q": ":*:**",
      "_": "1561220954886",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 734,
    "start": 0,
    "docs": [
      {
        "name": "A-Bomb",
        "Gender": "Male",
        "Race": "Human",
        "div(Weight, div(product(Height,Height),100))": 1.0701545
      },
      {
        "name": "Abe Sapien",
        "Gender": "Male",
        "Race": "Ichthy Sapien",
        "div(Weight, div(product(Height,Height),100))": 0.17817494
      },
      {
        "name": "Abin Sur",
        "Gender": "Male",
        "Race": "Ungaran",
        "div(Weight, div(product(Height,Height),100))": 0.26296568
      },
      {
        "name": "Abomination",
        "Gender": "Male",
        "Race": "Human / Radiation",
        "div(Weight, div(product(Height,Height),100))": 1.0701545
      },
      {
        "name": "Ahrvase"
      }
    ]
  }
}
```

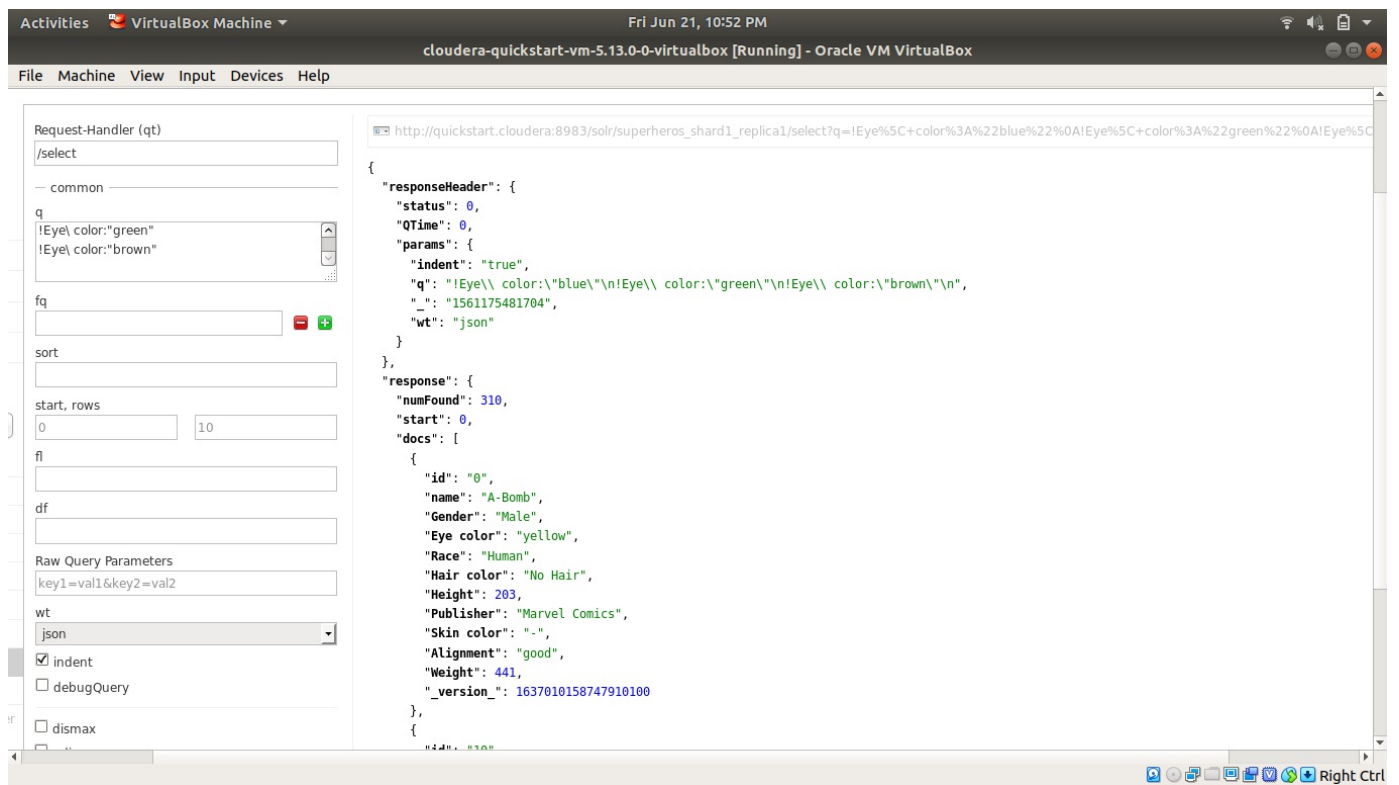
Solr2 - returns name, Height and Weight for all Female characters filtered by $0 < \text{weight} < 200$ (there are some negative heights and weights in the data). Could be thought of as making a Basketball team.



Solr3 - returns name, Gender, Height, Weight and Alignment for all characters of unknown race. A list of species for research.



Solr4 - return all fields for all characters whose Eye color is not blue, green or brown. Basically a list of all characters with abnormal Eye color.



Solr5 - return all fields for all characters whose Publisher is not Marvel or DC, sorted by publisher and then sorted by name. So basically a list of characters from the small studios.

Request-Handler (qt)

/select

— common

q

!Publisher:Marvel*

!Publisher:DC*

fq

sort

Publisher asc, name asc

start, rows

0

10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=!Publisher%3AMarvel%0A!Publisher%3A+DC*%0A&sort=Publisher+asc%2C+name+asc&wt=json&indent=2

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 9,
    "params": {
      "sort": "Publisher asc, name asc",
      "indent": "true",
      "q": "!Publisher:Marvel*\n!Publisher: DC*\n",
      " ": "1561219130224",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 131,
    "start": 0,
    "docs": [
      {
        "id": "199",
        "name": "Daphne Powell",
        "Gender": "Female",
        "Eye color": "-",
        "Race": "-",
        "Hair color": "-",
        "Height": -99,
        "Publisher": "ABC Studios",
        "Skin color": "-",
        "Alignment": "good",
        "Weight": -99,
        "_version_": 1637010158091565000
      },
      {
        "id": "363",
        "name": "Jim Powell",
        "Gender": "Male",
        "Eye color": "-",
        "Race": "-",
        "Hair color": "-",
        "Height": -99,
        "Publisher": "ABC Studios",

```

Nested Queries

*NOTE - the <AND *query*:“someField:someValue”> syntax denotes the nested query.

Solr6 - return all fields for all characters whose name contains “man” (outer) from all characters whose Gender is Male (inner). So it’s a list of characters with uncreative names.

Request-Handler (qt)

/select

common

q

name:"man" AND _query_:"Gender:Male"

fq

sort

start, rows

010

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=name%3A"man"&AND+_query_%3A%22Gender%3AMale%22&wt=json&indent=true

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 2,
    "params": {
      "indent": "true",
      "q": "name:'man' AND _query_:\"Gender:Male\"",
      "": "1561219610045",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 47,
    "start": 0,
    "docs": [
      {
        "id": "5",
        "name": "Absorbing Man",
        "Gender": "Male",
        "Eye color": "blue",
        "Race": "Human",
        "Hair color": "No Hair",
        "Height": 193,
        "Publisher": "Marvel Comics",
        "Skin color": "-",
        "Alignment": "bad",
        "Weight": 122,
        "_version_": 1637010158768881700
      },
      {
        "id": "27",
        "name": "Animal Man",
        "Gender": "Male",
        "Eye color": "blue",
        "Race": "Human",
        "Hair color": "Blond",
        "Height": 183,
        "Publisher": "DC Comics",
        "Skin color": "-",
        "Alignment": "good",
        "Weight": 83,
        "_version_": 1637010158787756000
      }
    ]
  }
}
```

Solr7 - return name, Gender, Race, Height and Weight of all characters who are approximately my size (yes, I fall in those ranges). Weight is inner query and Height is outer.

Request-Handler (qt)

/select

common

q

Height:[180 TO 185] AND _query_:"Weight:[87 TO 92]"

fq

sort

start, rows

010

fl

name, Gender, Race, Height, Weight

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

olr/#/~cores

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=Height%3A%5B180+TO+185%5D+AND+_query_%3A%22Weight%3A%5B87+TO+92%5D%22&fl=name

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "fl": "name, Gender, Race, Height, Weight",
      "indent": "true",
      "q": "Height:[180 TO 185] AND _query_:\"Weight:[87 TO 92]\"",
      "_": "1561220380207",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 23,
    "start": 0,
    "docs": [
      {
        "name": "Abin Sur",
        "Gender": "Male",
        "Race": "Ungaran",
        "Height": 185,
        "Weight": 90
      },
      {
        "name": "Adam Strange",
        "Gender": "Male",
        "Race": "Human",
        "Height": 185,
        "Weight": 88
      },
      {
        "name": "Alan Scott",
        "Gender": "Male",
        "Race": "-",
        "Height": 180,
        "Weight": 90
      },
      {
        "name": "Annihilus",
        "Gender": "Male",
        "Race": "-",

```

Solr8 - return all fields for all characters whose Race is Human (outer) and whose Weight is less than 0 (inner) in descending order by Height. So essentially a list of humans with unknown weight.

Request-Handler (qt)

/select

common

q

Race:"Human" AND _query_:"Weight:[* TO 0]"

fq

sort

Height desc

start, rows

010

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=Race%3A%22Human%22+AND+_query_%3A%22Weight%3A%5B*+TO+0%5D%22&sort=Height+desc

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "sort": "Height desc",
      "indent": "true",
      "q": "Race:\"Human\" AND _query_:\"Weight:[* TO 0]\"",
      "_": "1561221840405",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 47,
    "start": 0,
    "docs": [
      {
        "id": "557",
        "name": "Rey",
        "Gender": "Female",
        "Eye color": "hazel",
        "Race": "Human",
        "Hair color": "Brown",
        "Height": 297,
        "Publisher": "George Lucas",
        "Skin color": "-",
        "Alignment": "good",
        "Weight": -99,
        "_version_": 1637010159079260200
      },
      {
        "id": "307",
        "name": "Hancock",
        "Gender": "Male",
        "Eye color": "brown",
        "Race": "Human",
        "Hair color": "Black",
        "Height": 188,
        "Publisher": "Sony Pictures",
        "Skin color": "-",
        "Alignment": "good",
        "Weight": -99
      }
    ]
  }
}
```

Solr9 - return all fields for all characters whose publisher is Marvel (inner) and whose Alignment is bad (outer) sorted by Race descending and then sorted by name ascending. Basically a list of all the Marvel bad guys.

Request-Handler (qt)

/select

— common

q

Alignment:"bad" AND
query:"Publisher:*Marvel**

fq

sort

Race desc, name asc

start, rows

010

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Execute Query

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=Alignment%3A%22bad%22+AND+_query_%3A%22Publisher%3A*Marvel**%22&sort=Race+desc%2C+name+asc&wt=json&indent=true

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 5,
    "params": {
      "sort": "Race desc, name asc",
      "indent": "true",
      "q": "Alignment:\\"bad\\" AND _query_:\\"Publisher:*Marvel*\\"",
      "": "1561222805139",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 115,
    "start": 0,
    "docs": [
      {
        "id": "161",
        "name": "Carnage",
        "Gender": "Male",
        "Eye color": "green",
        "Race": "Symbiote",
        "Hair color": "Red",
        "Height": 185,
        "Publisher": "Marvel Comics",
        "Skin color": "-",
        "Alignment": "bad",
        "Weight": 86,
        "_version_": 1637010158870593500
      },
      {
        "id": "619",
        "name": "Spider-Carnage",
        "Gender": "Male",
        "Eye color": "-",
        "Race": "Symbiote",
        "Hair color": "-",
        "Height": -99,
        "Publisher": "Marvel Comics",
        "Skin color": "-",
        "Alignment": "bad",
        "Weight": -99,

```

Solr10 - return all fields for all characters whose Publisher is DC (inner) and whose Hair color is Blond (outer) filtered by Height and Weight in the range of classical attractiveness. So it's a list of the Blond DC heartthrobs.

Request-Handler (qt)

/select

common

q

Hair\ color:"Blond" AND
query:"Publisher:"DC"

fq

leight:[170 TO 185] Weight:[75 TO 100]

sort

start, rows

010

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

json

☒ indent
☐ debugQuery

☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck

Execute Query

http://quickstart.cloudera:8983/solr/superheros_shard1_replica1/select?q=Hair%5C+color%3A%22Blond%22+AND+_query_%3A%22Publisher%3A%22&fq=Height%3A%5B1

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 3,
    "params": {
      "indent": "true",
      "q": "Hair\\ color:\\ \"Blond\\\" AND _query_:\\ \"Publisher:\\ \"DC\\\"\"",
      "wt": "json",
      "fq": "Height:[170 TO 185] Weight:[75 TO 100]"
    }
  },
  "response": {
    "numFound": 14,
    "start": 0,
    "docs": [
      {
        "id": "7",
        "name": "Adam Strange",
        "Gender": "Male",
        "Eye color": "blue",
        "Race": "Human",
        "Hair color": "Blond",
        "Height": 185,
        "Publisher": "DC Comics",
        "Skin color": "-",
        "Alignment": "good",
        "Weight": 88,
        "_version_": 1637010158769930200
      },
      {
        "id": "13",
        "name": "Alan Scott",
        "Gender": "Male",
        "Eye color": "blue",
        "Race": "-",
        "Hair color": "Blond",
        "Height": 180,
        "Publisher": "DC Comics",
        "Skin color": "-",
        "Alignment": "good",

```

Limitation

1. We were not able to run MapReduce on IntelliJ. There were a lot of errors. We shifted to the Eclipse and we perform the logic for the MapReduce Problems.
2. Eclipse configuration also took some time to understand.

Conclusion

We have learnt some advanced concepts of Mapreduce, Hive and Apache Solr.

References

<http://stevekrenzel.com/finding-friends-with-mapreduce>

<https://acadgild.com/blog/hive-complex-data-types-with-examples>

https://lucene.apache.org/solr/guide/6_6/introduction-to-solr-indexing.html

Contribution

Mapreduce Problem:

Usecase-1

1. Mapper algorithm: Eric chadwick
2. Reducer Algorithm: Maseerah Muradabadi, Nikita Goyal

Usecase-2

1. Mapper and Input: Nikita Goyal
2. Reducer: Eric Chadwick and Maseerah Muradabadi

Hive Usecase:

1. 4 queries -Maseerah Muradabadi
2. 3 queries -Nikita Goyal
3. 3 Queries- Eric chadwick

Solr Usecase:

1. 4 Nested queries: Eric Chadwick
2. 3 Queries: Nikita Goyal
3. 3 queries : Maseerah Muradabadi