**LAB 1**

**Problem Statement 1**

package Facebook;

import java.io.IOException;

import java.util.Arrays;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

import java.util.StringTokenizer;

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 Facebook {

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]));

}

}

}

}

public static class Reduce extends MapReduceBase

implements Reducer<Text, Text, Text, Text>{

public void reduce(Text key, Iterator<Text> values,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException{

Text[] texts = new Text[2];

int index = 0;

while(values.hasNext()){

texts[index++] = new Text(values.next());

}

String[] list1 = texts[0].toString().split(" ");

String[] list2 = texts[1].toString().split(" ");

List<String> list = new LinkedList<String>();

for(String friend1 : list1){

for(String friend2 : list2){

if(friend1.equals(friend2)){

list.add(friend1);

}

}

}

StringBuffer sb = new StringBuffer();

for(int i = 0; i < list.size(); i++){

sb.append(list.get(i));

if(i != list.size() - 1)

sb.append(" ");

}

output.collect(key, new Text(sb.toString()));

}

}

public static void main(String[] args) throws Exception{

JobConf conf = new JobConf(Facebook.class);

conf.setJobName("Friend");

conf.setMapperClass(Map.class);

conf.setReducerClass(Reduce.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(Text.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(Text.class);

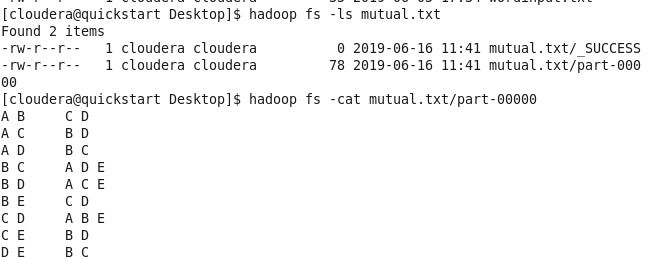
FileInputFormat.setInputPaths(conf, new Path(args[0]));

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

JobClient.runJob(conf);

}

}



**Problem statement 2**

package document;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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 document {

public static class Map

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class SumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(document.class);

job.setMapperClass(Map.class);

job.setCombinerClass(SumReducer.class);

job.setReducerClass(SumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

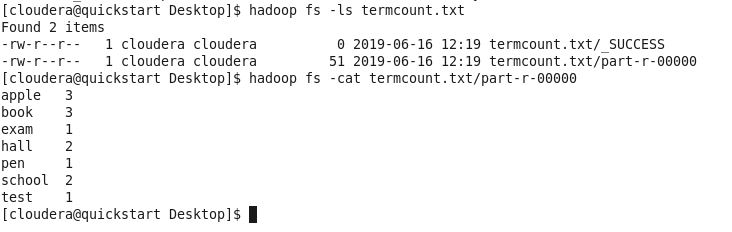
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}



**Problem Statement 3:**

**Hive usecase:**

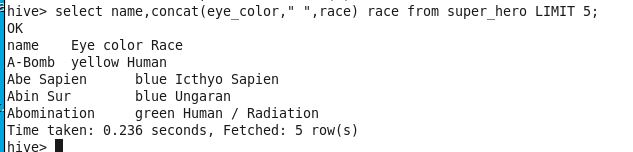
create table super\_hero(id int,name string,gender string,eye\_color string,race string,hair\_color string,height int,publisher string,skin\_color string,alignment string,weight int) row format delimited fields terminated by ',' stored as textfile;

Load data:

load data local inpath '/home/cloudera/Desktop/heroes\_information.csv' into table super\_hero;

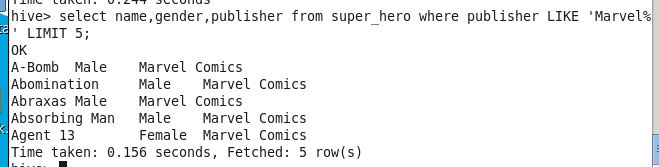
Query 1:

**Query 1: concatenating superheroes eye color and race using "concat"** Using Concat: select name,concat(eye\_color," ",race) race from super\_hero LIMIT 5;



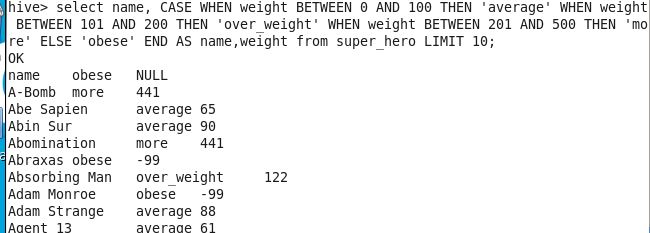
**Query 2:Finding super heroes by publisher marvel**

Using Like select name,gender,publisher from super\_hero where publisher LIKE 'Marvel%' LIMIT 5;

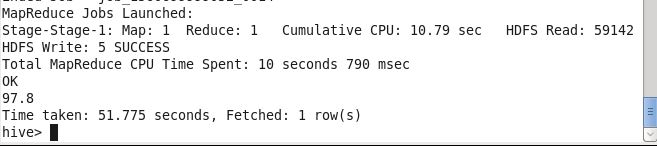


**Query 3: categorizing the super heroes based on weight using CASE**

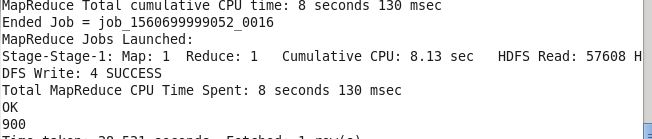
select name, CASE WHEN weight BETWEEN 0 AND 100 THEN 'average' WHEN weight BETWEEN 101 AND 200 THEN 'over\_weight' WHEN weight BETWEEN 201 AND 500 THEN 'more' ELSE 'obese' END AS name,weight from super\_hero LIMIT 10;



**Query 4: average weight of super heroes with no hair** select avg(weight) from super\_hero where hair\_color ="No Hair";

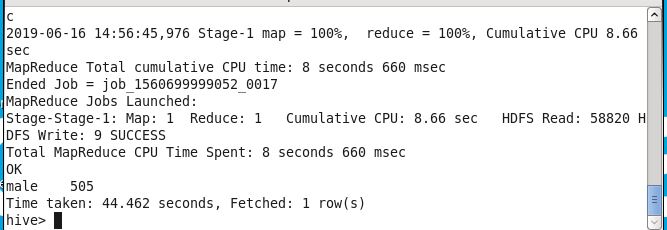


**Query 5: Finding maximum height of super heroes** select max(weight) from super\_hero;



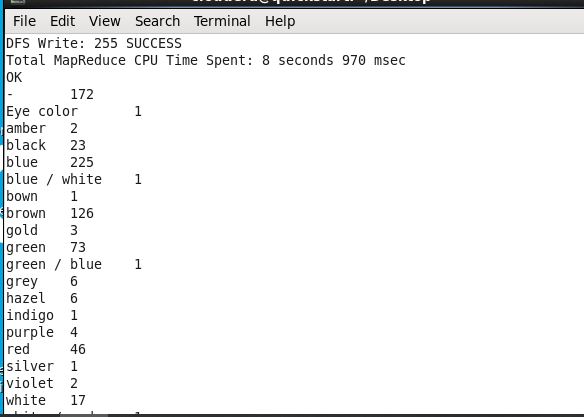
**Query 6: count of male super heroes**

select 'male',count(\*) from super\_hero where gender = 'Male';



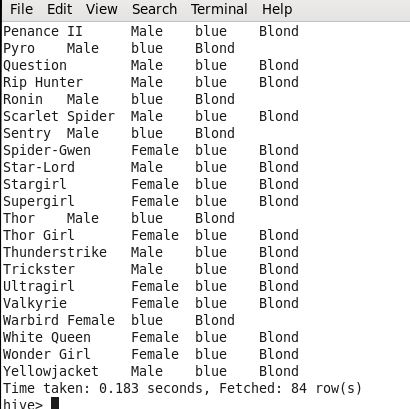
**Query 7:Trending eye color of super heroes**

select eye\_color,count(\*) from super\_hero group by eye\_color;



**Query 8: Super heroes with blond hair and blue eyes**

Select name,gender,eye\_color,hair\_color from super\_hero where eye\_color ="blue" and hair\_color = "Blond";



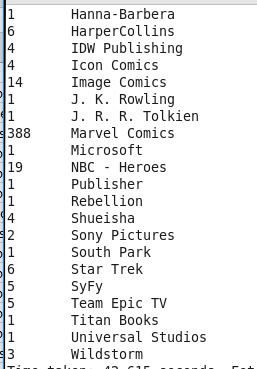
**Query 9: Super heroes with ave height and weight**

select name,gender,race from super\_hero where height>150 and weight >200;



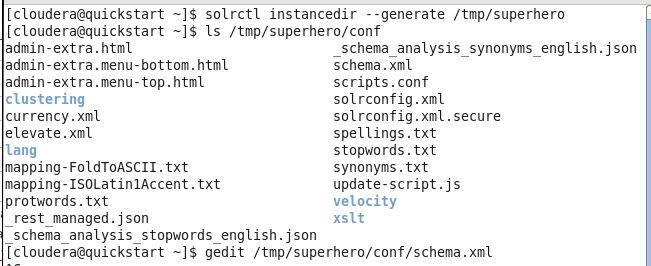
**Query 10: Super heroes from different publishers**

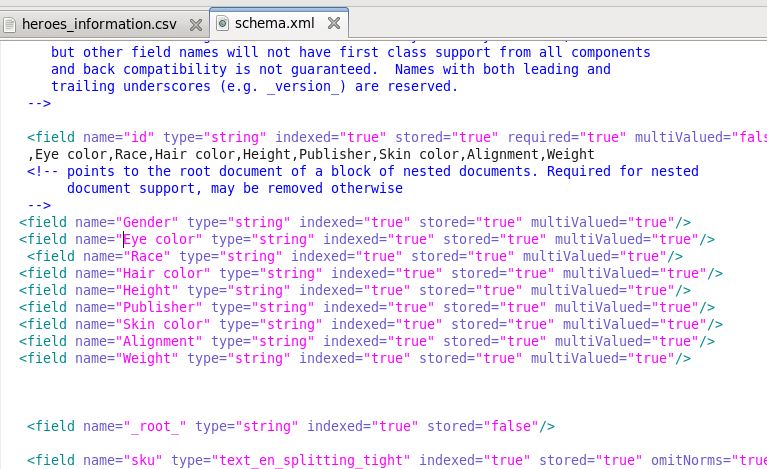
select count(\*),publisher from super\_hero group by publisher;

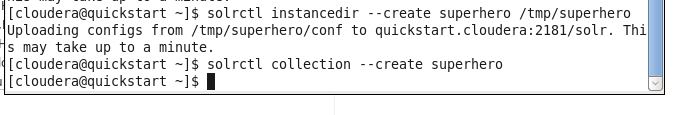


**SOLR**

**Creating Solr collection** Creating solr collection and editing schema.







**Query 1:**

Finding the Male count:



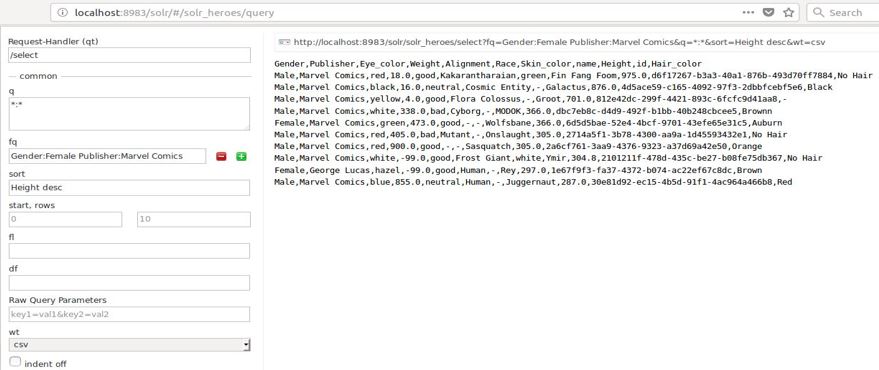
**Query 2:**

Details of the Female super heroes



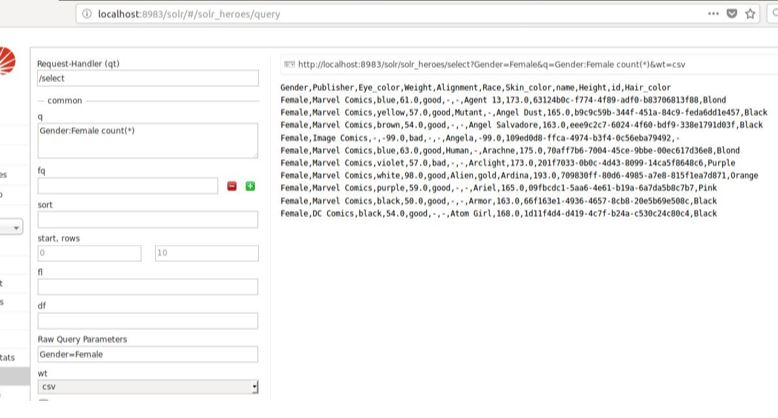
**Query 3:**

Female super heroes published by Marvel and height in descending order



**Query 4:**

Count of Female super heroes



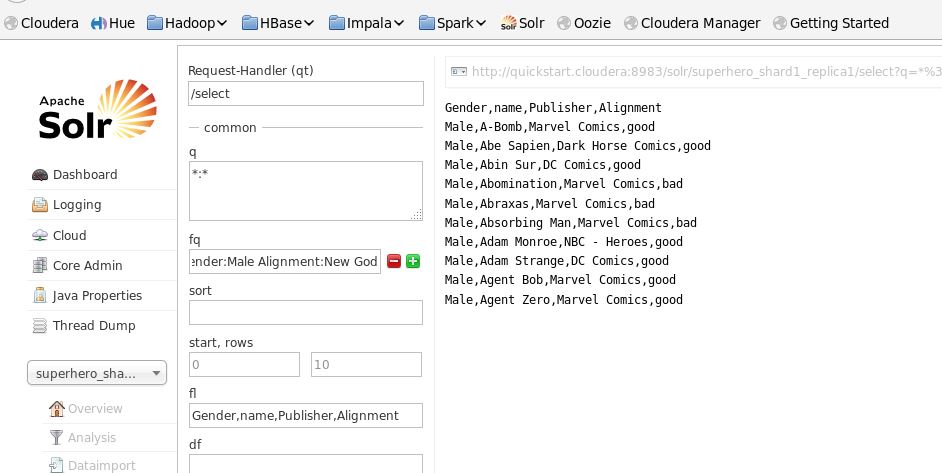
**Query 5:**

Facet query with Publisher

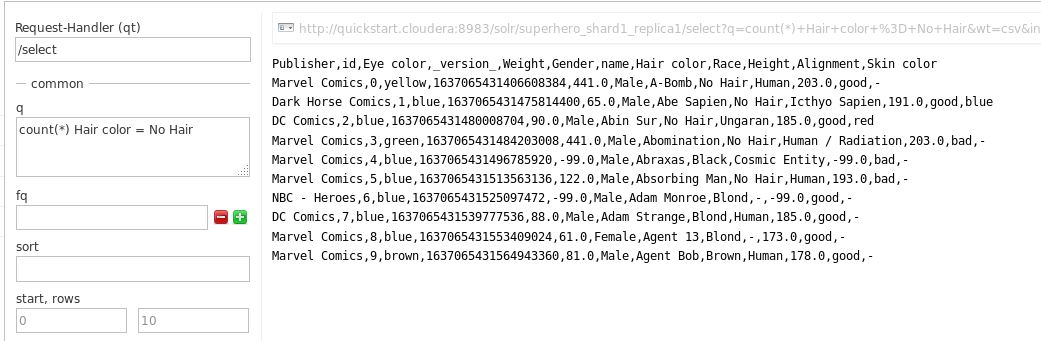


**Query 6:**

Gender Female with alignment good



**Query 7:**

super hero with bald head 

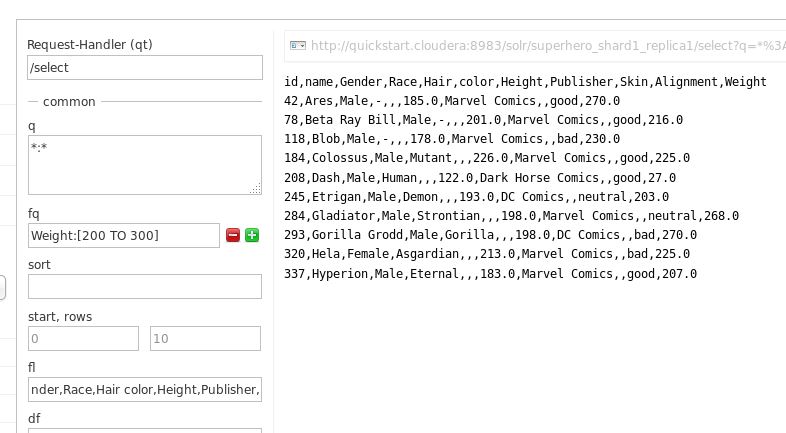
**Query 8:**

super hero with vampire race



**Query 9:**

super hero with weight range between 200-300



**Query 10:**

Female with height descending

