ACADGILD

MASTERING HADOOP WITH REAL WORLD USE CASES

10 real world use cases with complete source code and explanation

Table Of Content

Chapter - 1

MAPREDUCE USE CASE UBER DATA ANALYSIS

In this chapter, the Uber dataset in Hadoop will be analysed using MapReduce in Java

01 Uber Data Set Description

The Uber dataset consists of the following four columns:

- Dispatching_base_number
- Date
- Active_vehicles
- Trips

Download the Dataset from this link.

02 Problem Statement 1:

Find the days on which each basement has more number of trips.

A) SOLUTION

Source Code   Mapper Class

```
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
java.text.SimpleDateFormat format = new
java.text.SimpleDateFormat("MM/dd/yyyy");
String[] days ={"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};
private Text basement = new Text();
Date date = null;
private int trips;
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
String line = value.toString();
String[] splits = line.split(",");
basement.set(splits[0]);
try {
date = format.parse(splits[1]);
} catch (ParseException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
trips = new Integer(splits[3]);
String keys = basement.toString()+ " "+days[date.getDay()];
context.write(new Text(keys), new IntWritable(trips));
}
}
```

The combination of the **Dispatching_base_number** and the **day** of the week as **“key”** and the number of trips as **“value”** is derived from the Mapper

- First,the date will be parsed (which is in a string format) and converted into **date datatype** using the **SimpleDateFormat** class in Java
- getDay() will be used to obtain the day of the date, which will return an integer value to the day of the week's number.
- An array will be constructed that will consist of all the days from Sunday to Monday into this array the value returned by **getDay()** is to be passed in order to get the day of the week
- After this operation, the combination of **Dispatching_base_number+Day** of the week is returned as key and the number of trips as value.

Reducer Class

In the reducer, the sum of trips for each basement and for each particular day will be calculated, by using the followingcommands:

```
public static class IntSumReducer
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 (IntWritableval : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
```

To download the complete source code, click here.

Running the Program

A JAR file needs to be constructed for the above program, and then run it as a normal Hadoop program by passing the input dataset and the output file path as shown below

```
hadoop jar uber1.jar /uber /user/output1
```

To know how to execute your first MapReduce application, click here.

B) SAMPLE OUTPUT

In the output file directory, a part of the file is created that contains the following output:

```
B02512 Sat 15026      B02617 Fri 125067     B02682 Tue 76905      B02765 Sun 22536
B02512 Sun 10487      B02617 Mon 80591      B02682 Wed 86252      B02765 Thu 30408
B02512 Thu 15809      B02617 Sat 127902     B02764 Fri 326968     B02765 Tue 22741
B02512 Tue 12041      B02617 Sun 91722      B02764 Mon 214116     B02765 Wed 24340
B02512 Wed 12691      B02617 Thu 118254     B02764 Sat 356789
B02598 Fri 93126      B02617 Tue 86602      B02764 Sun 249896
B02598 Mon 60882      B02617 Wed 94887      B02764 Thu 304200
B02598 Sat 94588      B02682 Fri 114662     B02764 Tue 221343
B02598 Sun 66477      B02682 Mon 74939      B02764 Wed 241137
B02598 Thu 90333      B02682 Sat 120283     B02765 Fri 34934
B02598 Tue 63429      B02682 Sun 82825      B02765 Mon 21974
B02598 Wed 71956      B02682 Thu 106643     B02765 Sat 36737
```

03 Problem Statement 2:

Find the days on which each basement has more number of active vehicles.

Source Code ■ Mapper Class

```java
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
java.text.SimpleDateFormat format = new
java.text.SimpleDateFormat("MM/dd/yyyy");
String[] days ={"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};
private Text basement = new Text();
Date date = null;
private intactive_vehicles;
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
String line = value.toString();
String[] splits = line.split(",");
basement.set(splits[0]);
try {
date = format.parse(splits[1]);
} catch (ParseException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
active_vehicles = new Integer(splits[3]);
String keys = basement.toString()+ " "+days[date.getDay()];
context.write(new Text(keys), new IntWritable(active_vehicles));
}
}
```

- The combination of the **Dispatching_base_number** and the **day of the week** will be taken from the Mapper as key and the number of active vehicles as value.

- First, the date will be parsed (which is in a string format to date data type) using the **SimpleDateFormat** class in Java. Now, to obtain the day of the date, we will use **getDay()**, which will return an integer value to the day of the week's number
  - An array will be constructed that will consist of all the days from Sunday to Monday into this array the value returned by **getDay()** is to be passed in order to get the day of the week

- Next, the combination of **Dispatching_base_number+Day** of the week will be returned as key and the number of active vehicles as value.

Reducer Class

A sum of active vehicles for each basement will be calculated for every dayusing the following commands:

```
public static class IntSumReducer
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 (IntWritableval : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
```

To download the complete source code, click here.

Running the Program

First, a JAR file will be built for the above program and it will be run as a normal Hadoop program by passing the input dataset and the output file path as shown below.

```
hadoop jar uber2.jar /uber /user/output2
```

A) SAMPLE OUTPUT

In the output file directory, a part of the file is created that contains the following output:

```
B02512 Fri 2221
B02512 Mon 1676
B02512 Sat 1931
B02512 Sun 1447
B02512 Thu 2187
B02512 Tue 1763
B02512 Wed 1900
B02598 Fri 9830
B02598 Mon 7252
B02598 Sat 8893
B02598 Sun 7072
B02598 Thu 9782
B02598 Tue 7433
B02598 Wed 8391
B02617 Fri 13261
B02617 Mon 9758
B02617 Sat 12270
B02617 Sun 9894
B02617 Thu 13140
B02617 Tue 10172
B02617 Wed 11263
B02682 Fri 11938
B02682 Mon 8819
B02682 Sat 11141
B02682 Sun 8688
B02682 Thu 11678
B02682 Tue 9075
B02682 Wed 10092
B02764 Fri 36308
B02764 Mon 26927
B02764 Sat 33940
B02764 Sun 26929
B02764 Thu 35387
B02764 Tue 27569
B02764 Wed 30230
B02765 Fri 3893
B02765 Mon 2810
B02765 Sat 3612
B02765 Sun 2566
B02765 Thu 3646
B02765 Tue 2896
B02765 Wed 3152
```

Chapter - 2

MAPREDUCE USE CASE TITANIC DATA ANALYSIS

In this chapter, the Titanic dataset in Hadoop will be analysed using MapReduce in Java.

MapReduce Use Case - Titanic Data Analysis

There have been many cases of extreme disasters in the history of the mankind, but the magnitude of Titanic's disaster ranks as high as the depth it sank to. So much so, that it became a benchmark for the subsequent disasters that occurred, of which a lot were described as "titanic in proportion," implying huge losses.

Anybody who has read about the Titanic knows that a combination of natural events and human errors led to the sinking of the ship on its fateful maiden journey from Southampton to New York on April 14, 1912.

Several questions have been put forward to understand the cause(s) of the tragedy. The most common question asked about the tragedy is related to the reason behind its sinking, and the most intriguing one among them is how a 46,000-ton ship sank to a depth of 13,000 feet in a matter of 3 hours. The physics behind it is mind boggling indeed!

There have been as many investigations as there have been questions asked, and an equal number of analyses have been carried out to arrive at conclusions.

The analysis described ahead does not address why or what made the Titanic sink, it instead attempts to analyze the publicly available data on the ship.

The following analysis uses Hadoop MapReduce to analyze the following:

- Average age of the people (both male and female) who died.
- Number of people who managed to survive (class-wise).

The data used in this example is taken from a dataset that was publically available.

The Titanic dataset is described under the heading "Data Set Description," which will be used to gain meaningful insights.

01 Titanic Data Set Description

Column 1: Passenger ID	Column 7: SibSp
Column 2: Survived (survived=0 & died=1)	Column 8: Parch
Column 3: Pclass	Column 9: Ticket
Column 4: Name	Column 10: Fare
Column 5: Sex	Column 11: Cabin
Column 6: Age	Column 12: Embarked

Download the Dataset using this link.

02 Problem Statement 1:

In this problem relation, we will find the average age of males and females who died in the Titanic tragedy.

A) SOLUTION

Source Code

From the Mapper we will derive the following:

- Gender as “key”
- Age as “values”

These values will be passed to the shuffle and sort phase and further sent to the reducer phase where aggregation of the values will be performed.

Mapper Class

```
public class Average_age {
public static class Map extends Mapper<LongWritable, Text, Text,
IntWritable> {
private Text gender = new Text();
private IntWritable age = new IntWritable();
public void map(LongWritable key, Text value, Context context ) throws
IOException, InterruptedException {
   String line = value.toString();
   String str[]=line.split(",");
         if(str.length>6){
gender.set(str[4]);
         if((str[1].equals("0")) ){
            if(str[5].matches("\\d+")){
inti=Integer.parseInt(str[5]);
age.set(i);
    }
   }
 }
context.write(gender, age);
  }
}
```

Explanation for theMapper code.

Line 1 - A class is taken by the name “Average_age”

Line 2 - The Mapper default class is extended to have the arguments **keyIn** as **LongWritable**, **ValueIn** as Text, **KeyOut** as Text, and **ValueOut** as **IntWritable**.

Line 3 - The declared variable constitutes a private Text variable gender which will store the gender of the a person who died in the Titanic tragedy.

Line 4 - The declared variable constitutes a private IntWritable variable age which will store the age of the MapReducethat deals with Key and Value pairs. Here, the key is set as gender & value as age.

Line 5 - The map method has been overridden that will run one time for every line.

Line 6 - Line is stored in a string variable.

Line 7 - The line has been split by using a comma "," delimiter, and the values have been stored in a String array so that all the columns in a row are stored in it.

Line 8 - A condition is taken in which if we have the String array's length greater than 6, i.e., if the line or a row has at least 7 columns, then do the rest this helps in eliminating the ArrayIndexOutOf BoundsException in some cases if the data inside the data set is incorrect.

Line 9 - Stores the gender, which is in the fifth column.

Line 10 - A condition has been included in which the algorithm proceeds forward regardless the fact a person is alive or not.

Line 11 - Checks whether the data in that index is numeric or not by using a regular expression that can be achieved by **matches function in java**. If it is numeric, then the algorithm will proceed.

Line 12 - Numeric data has been converted into an integer data by type casting.

Line 13 - Age of the person is stored in age.

Line 17 - The key and value are written into the context that will be the output of the map method.

Reducer Code

```
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;
int l=0;
 for (IntWritableval : values) {
 l+=1;
 sum += val.get();
 }
 sum=sum/l;
context.write(key, new IntWritable(sum));
   }
 }
```

Explanation for the Reducer code.

Line 1 - Extends the default Reducer class with arguments KeyIn as Text and ValueIn as IntWritable, which are same as the outputs of the Mapper class and KeyOut as Text and ValueOut as IntWritbale (which will be the final output of our MapReduce program).

Line 2 - The Reduce method is overridden that will run each time for every key.

Line 3 - An integer sum is declared that will store the sum of all the ages of the people in it.

Line 4 - Here, another variable, l, is taken that will be incremented those many numbers of times as the values that are present for that key.

Line 5 - A for each loop is taken which will run each time for the number of values present inside the "Iterable values," which come in from the shuffle and sort phase after the mapper phase.

Line 6 - Value iterated.

Line 7 - Stores and calculates the sum of values.

Line 8 - Closing bracket.

Line 9 - An average is calculated for the sum obtained and the respected key and the obtained sum, as value to the context is written.

Configuration Code

```
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
```

These two configuration classes are included in the main class to specify the Output key and value type of the Mapper.

You can download the source code using this link:

To execute the JAR file in order to get the result of the first problem relation.

```
Hadoop jar average.jar /TitanicData.txt /avg_out
```

Here,“Hadoop” specifies that we are running a Hadoop command,“JAR” specifies which type of the application is being run,and “average.JAR” is the JAR file created that consists of the above source code. The paths of the Input file name in our case is “TitanicData.txt,” and the output file where output will be saved has been given the name“avg_out.”

To view the output, enter:

```
hadoop dfs -cat /avg_out/part-r-00000
```

Here, ‘hadoop’ specifies that we are running a Hadoop command,“dfs” specifies that we are performing an operation related to Hadoop Distributed File System, “- cat” is used to view the contents of a file, and “avg_out/part-r-00000” is the file where output is stored.Part file is created by default by the TextInputFormat class of Hadoop.

```
15/10/23 06:25:37 WARN util.NativeCodeLoader: Unable to load native-
hadooplibrary for your platform... using builtin-java classes where
applicable

15/10/23 06:25:40 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0 :8032

15/10/23 06:25:43 WARN mapreduce.JobSubmitter: Hadoop command-line option
parsin g not performed. Implement the Tool interface and execute your
application with ToolRunner to remedy this.

15/10/23 06:25:44 INFO input.FileInputFormat: Total input paths to process
: 1

15/10/23 06:25:44 INFO mapreduce.JobSubmitter: number of splits:1

15/10/23 06:25:45 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_14 45558139946 0012

15/10/23 06:25:46 INFO impl.YarnClientlmpl: Submitted application
application 14 45558139946_0012

15/10/23 06:25:46 INFO mapreduce.Job: The url to track the job:
http://localhost .localdomain:8088/proxy/application 1445558139946_0012/

15/10/23 06:25:46 INFO mapreduce.Job: Running job: job 1445558139946_0012

15/10/23 06:26:06 INFO mapreduce.Job: Job job_1445558139946_0012 running in
uber mode : false 15/10/23 06:26:06 INFO mapreduce.Job: map 0% reduce 0%

15/10/23 06:26:21 INFO mapreduce.Job: map 100% reduce 0%

15/10/23 06:26:39 INFO mapreduce.Job: map 100% reduce 100%

15/10/23 06:26:40 INFO mapreduce.Job: Job job_1445558139946_0012 completed
successfully
```

A) SAMPLE OUTPUT

Use of the following script to execute hdfs command is deprecated. Instead use the hdfs command for it.

```
[acadgild@localhost-]$hadoopdfs -cat /avg_out/part-r-00000 DEPRECATED

15/10/23 06:27:29 WARN util.NativeCodeLoader: Unable to load native-
hadooplibrary for your platform... using builtin-java classes where applicable

Sex 0

female 28

male 30
```

03 Problem Statement 2:

Find the number of people who died or survived in each class with their genders and ages.

A) SOLUTION

Source code

From the Mapper,our aim is to get the composite key, which is the combination of the passenger's classes, genders, and their ages. The final int value, '1,' is the value which will be passed to the shuffle and sort phase. These are further sent to the reducer phase where the aggregation of the values is performed.

Mapper code

```
public class Female_survived {
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
{
private Text people = new Text();
private final static IntWritable one = new IntWritable(1);
public void map(LongWritable key, Text value, Context context ) throws
IOException, InterruptedException {
  String line = value.toString();
  String str[]=line.split(",");
  if(str.length>6){
  String survived=str[1]+" "+str[4]+" "+str[5];
people.set(survived);
context.write(people, one);
  }
 }
}
```

Explanation for the Mapper code.

Line 1 - A class is taken with the name “survived.”

Line 2 - The Mapper default class is extended having the arguments:keyIn as LongWritable, ValueIn as Text, KeyOut as Text, and ValueOut as IntWritable.

Line 3 - A private Text variable ‘people’ is declared, which will store the Text value as String.

Line 4 - A private static final IntWritable variable ‘one’is declared that cannot be modified. MapReduce deals with the Key and Value pairs.Here, we can set the key as pclass and value as “1.”

Line 5 - The map method is overridden that will run one time for every line.

Line 6 - Stores a line in a string variable ‘line.’

Line 7 - Here, we are splitting the line by using a comma “,” delimiter and storing values in a String array so that all the columns in a row are stored in the string array.

Line 8 - States a condition; If the string array’s length is greater than 6, i.e., if the line or row has at least 7 columns, then subsequent steps are executed.This helps in eliminating the ArrayIndexOutOfBoundsException in some cases if the data inside the data set is incorrect.

Line 9 - Here, we are making a composite by combining the necessary columns. The 2nd column specifies whether the person is alive or not,the 5thcolumn specifies a person’s gender, and the 6th column specifies their age.
- This is the easiest way of forming a composite key in MapReduce and the functionality is very similar to the original way of making a composite key.

Line 10 - Stores and prepares the composite key in people.

Line 11 - Here, we are writing the key and value into the context which will be the output of the map method.

Reducer Code

```
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 (IntWritableval : values) {
   sum += val.get();
  }
context.write(key, new IntWritable(sum));
 }
}
```

Explanation for theReducer code.

Line 1 - Extends the default Reducer class with arguments KeyIn as Text and ValueInas IntWritable, which are same as the outputs of the Mapper class and KeyOut as Text and ValueOut as IntWritbale, which will be final outputs of our MapReduce program.

Line 2 - The Reduce method will be overridden,which will run every time for every key.

Line 3 - Declares an integer variable sum, which will store the sum of all the values for each key.

Line 4 - A foreach loop is taken which will run each time for the values inside the "Iterable values" that come from the shuffle and sort phase after the Mapper phase.

Line 5 - Stores and calculates the sum of values.

Line 6 - Writes the respective key and the obtained sum as value to the context

In the above example, the use of the composite key made the complicated program simple.

Conf Code

```
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
```

These two configuration classes are included in the main class, whereasto specify the Output key and value type of the Mapper.

The source code can be downloaded using this link.

Way to execute the JAR file to get the result of the given problem relation:

```
hadoopjar peope_survived.jar /TitanicData.txt /people_out
```

Here,"hadoop" specifies that we are running a Hadoop command, and "JAR" specifies which type of application we are running and people_survived.JAR is the JAR file which we have created that consists of the above source code and the path of the Input file name.In this case, it is "TitanicData.txt" and the output file where the output is stored isgiven the name,"people_out."

Way to view the output:

```
hadoopdfs –cat /people_out/part-r 00000
```

Here,“Hadoop” specifies that we are running a Hadoop command, dfs specifies that we are performing an operation related to Hadoop Distributed File System,“- cat” is used to view the contents of a file, and “people_out” or “part-r-00000” is the file where output is stored. Part file is created by default by the “TextInputFormat” class of Hadoop.

```
[acadgild@localhost-]$hadoopJAR peope_survived.JAR /TitanicData.txt
/people_out

15/10/23 03:11:21 WARN util.NativeCodeLoader: Unable to load native-
hadooplibrary for your platform... using builtin-java classes where applicable

15/10/23 03:11:26 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0 :8032

15/10/23 03:11:28 WARN mapreduce.JobSubmitter: Hadoop command-line option
parsin g not performed. Implement the Tool interface and execute your
application with ToolRunner to remedy this.

15/10/23 03:11:28 INFO input.FileInputFormat: Total input paths to process : 1

15/10/23 03:11:28 INFO mapreduce.JobSubmitter: number of splits:1

15/10/23 03:11:29 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_14 45558139946 0005

15/10/23 03:11:29 INFO impl.YarnClientImpl: Submitted application application
14 45558139946_0005

15/10/23 03:11:30 INFO mapreduce.Job: The url to track the job:
http://localhost .localdomain:8088/proxy/application 1445558139946_0005/

15/10/23 03:11:30 INFO mapreduce.Job: Running job: job_1445558139946 0005 1
```

B) SAMPLE OUTPUT

```
acadgild@localhost-]$hadoopdfs -cat /people_out/part-r-00000 DEPRECATED: Use
of this script to execute hdfs command is deprecated. Instead use the hdfs
command for it.

15/10/23 03:08:58 WARN util.NativeCodeLoader: Unable to load native-
hadooplibrary for your platform... using builtin-java classes where applicable

0 female 10 1
0 female 11 1
0 female 14 1
```

```
0 female 14.5 1
0 female 16 1
0 female 17 1
0 female 18 5
0 female 2 4
0 female 20 2
0 female 21 3
0 female 22 2
0 female 23 1
0 female 24 2
0 female 25 3
0 female 26 2
0 female 27 1
```

Chapter - 3

MAPREDUCE USE CASE SENTIMENT ANALYSIS ON TWITTER DATA

In this chapter, the Sentiment Analysis On Twitter dataset in Hadoop will be analysed using MapReduce in Java

MapReduce Use Case - Sentiment Analysis On Twitter Data

This part of the e-book will explain to you how to perform a sentiment analysis on the data retrieved from Twitter using MapReduce. We will use the concept of distributed cache to implement sentiment analysis on the Twitter data.

01 Problem Statement 1:

Performing sentiment analysis on tweeter data using the AFFIN dictionary.

A) SOLUTION

Map-side joins can be performed using distributed cache. In this case, we will join the dictionary dataset that contains sentiment values for each word. In order to perform a sentiment analysis, a dictionary called AFINN will be used.

AFINN is a dictionary that consists of 2500 words, rated from +5 to -5, which depends on their meaning.

Download AFINN dictionary from here.

B) SUMMARY OF THE STEPS TO ACHIEVE THE RESULT.

Following steps have been followed to perform the sentiment analysis:

- Implementing distributed caching
- Writing a Mapper class to calculate the sentiments
- Writing a Reducer class to display all the mapper output
- Writing a Driver class for our MapReduce program

C) IMPLEMENTING DISTRIBUTED CACHING

In MapReduce, map-side joins are carried by the distributed cache. Distributed cache is applied when there are two datasets, where the smaller dataset size is limited to the cache memory of the cluster. Here, the dictionary is the smaller dataset, so we are using distributed cache. Following is the implementation of the distributed cache:

```
private HashMap<String,String>AFINN_map = new HashMap<String,String>();
@Override
public void setup(Context context) throws IOException
{
files = DistributedCache.getCacheFiles(context.getConfiguration());
System.out.println("files:"+ files);
Path path = new Path(files[0]);
FileSystem fs = FileSystem.get(context.getConfiguration());
FSDataInputStream in = fs.open(path);
BufferedReaderbr = new BufferedReader(new InputStreamReader(in));
String line="";
while((line = br.readLine())!=null)
{
String splits[] = line.split("\t");
AFINN_map.put(splits[0], splits[1]);
}
br.close();
in.close();
}
```

At the start, we declared a HashMap by the name AFINN_map to store the **key** and **value** i.e., the **dictionary** & the **rating.** You can download the dataset from here (Hyper link to be added to the dataset below is the link)

https://drive.google.com/file/d/0ByJLBTmJojjzZ0d1RVdBTDVjT28/view

The data is tab separated, containing the word and its rating. After we have read the file using the **FSDataInputStream**, read the file line-by-line using the **readLine()** method, split the line using the **tab** delimiter, and store the **word** and its **rating** in the HashMap that has been created with the name **AFINN_map**.

```
files =DistributedCache.getCacheFiles(context.getConfiguration());
```

The above line reads the cached file in the cluster. With that, our cached file was read and processed successfully, and the distributed cache implementation is completed.

D) WRITING A MAP METHOD TO CALCULATE THE SENTIMENTS

```
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException{
String line = value.toString();.
JSONParser jsonParser = new JSONParser();
try{
JSONObject obj =(JSONObject) jsonParser.parse(line);
String tweet_id = (String) obj.get("id_str");
String tweet_text=(String) obj.get("text");
String[] splits = twt.toString().split(" ");
int sentiment_sum=0;
for(String word:splits){
if(AFINN_map.containsKey(word))
{
Integer x=new Integer(AFINN_map.get(word));
sentiment_sum+=x;
}
}
context.write(new Text(tweet_id),new Text(tweet_text+"\t----->\t"+new
Text(Integer.toString(sentiment_sum))));
}catch(Exception e){
e.printStackTrace();
}
```

- The map method takes each record as input, and the record is converted into a string using the **toString** method. This record is passed as a JSONObject to the JSONParser, this JSONParser will parse the tweet which is in the form of a **JSON**Object.

- The **tweet_id**and the **tweet_text**are extracted,which are required for sentiment analysis as shown below:

```
JSONObjectobj =(JSONObject) JSONParser.parse(line);
String tweet_id = (String) obj.get("id_str");
String tweet_text=(String) obj.get("text");
```

- The tweet_text is split into words, using a "**for loop,**" which repeats for all the words in the tweet_text. An **inner join** is performed to find the matches from the words in the AFINN dictionary.If there is a match, then the rating of the word is taken and added to it.

This has been shown below:

```
String[] splits = twt.toString().split(" ");
intsentiment_sum=0;
for(String word:splits){
if(AFINN_map.containsKey(word))
{
Integer x=new Integer(AFINN_map.get(word));
sentiment_sum+=x;
}
}
context.write(new Text(tweet_id),new Text(tweet_text+"\t----->\t"+new
Text(Integer.toString(sentiment_sum))));
```

Finally, in the context, we are writing **tweet_id** as the **"key"** and the combination of **tweet_text** and **sentiment_sum** as **"value"**.

E) WRITING A REDUCER CLASS TO DISPLAY THE MAPPER OUTPUT

```
public static class Reduce extends Reducer<Text,Text,Text,Text>{
public void reduce(Text key, Text value, Context context) throws IOException,
InterruptedException{
context.write(key,value);
}
}
```

In the Reducer class, we simply pass the input of the Mapper as its output.

F) WRITING A DRIVER CLASS FOR OUR MAPREDUCE PROGRAM.

```
public static void main(String[] args) throws Exception
{
ToolRunner.run(new Parse(),args);
}
@Override
public int run(String[] args) throws Exception {
// TODO Auto-generated method stub
Configuration conf = new Configuration();
if (args.length != 2) {
System.err.println("Usage: Parse <in><out>");
System.exit(2);
}
DistributedCache.addCacheFile(new URI("/AFINN.txt"),conf);
Job job = new Job(conf, "SentimentAnalysis");
job.setJARByClass(Sentiment_Analysis .class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(NullWritable.class);
job.setOutputValueClass(Text.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
return 0;
}
```

In the Driver class, we need to provide the path for the cached dataset using the following line:

```
DistributedCache.addCacheFile(new URI("/AFINN.txt"),conf);
```

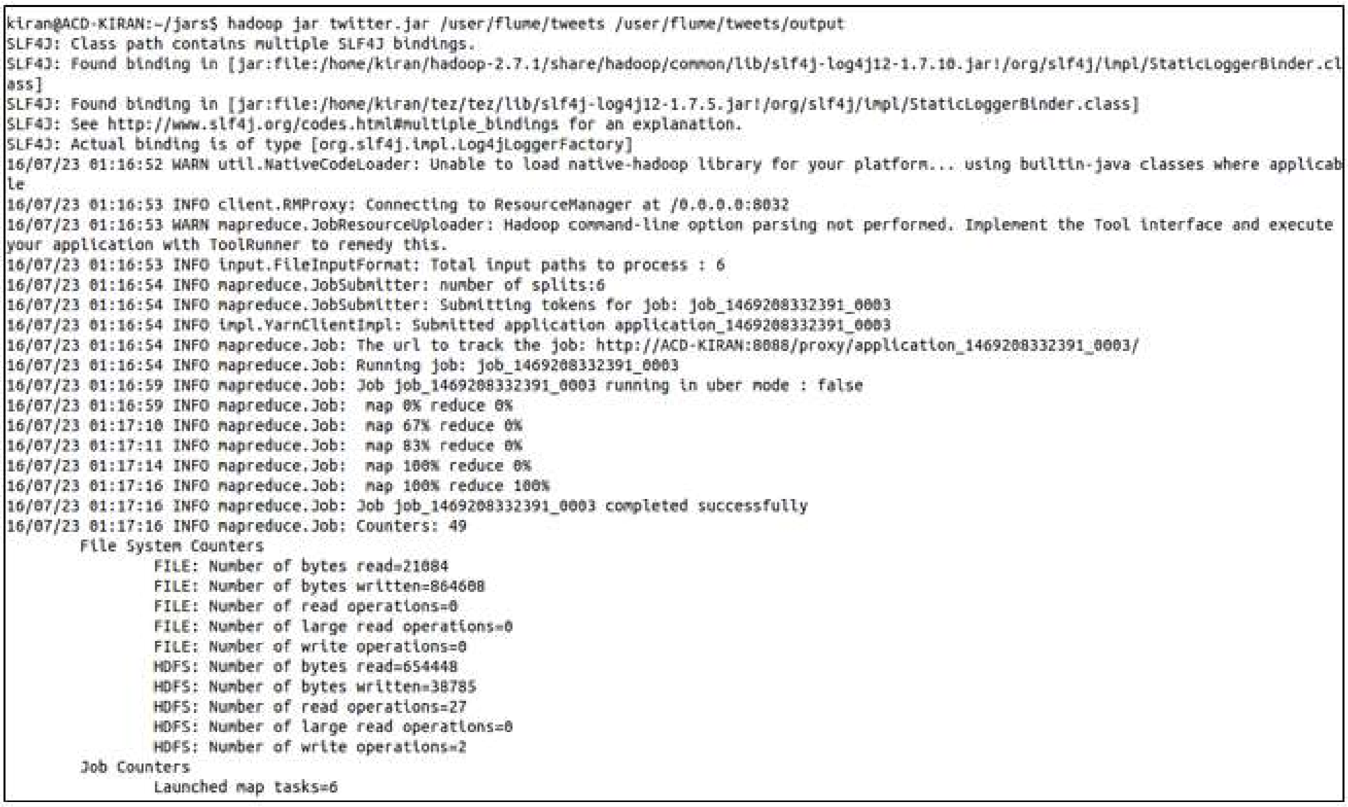
We also need to provide the input(tweets_folder) and output folder path as arguments.

Download the complete source code using this link.

G) HOW TO RUN THE MAPREDUCE ROGRAM?

In order to run this program,we need to build a JAR file of the project described above, using the normal

```
hadoopjar twitter.jar /user/flume/tweets /user/flume/tweets/output
```

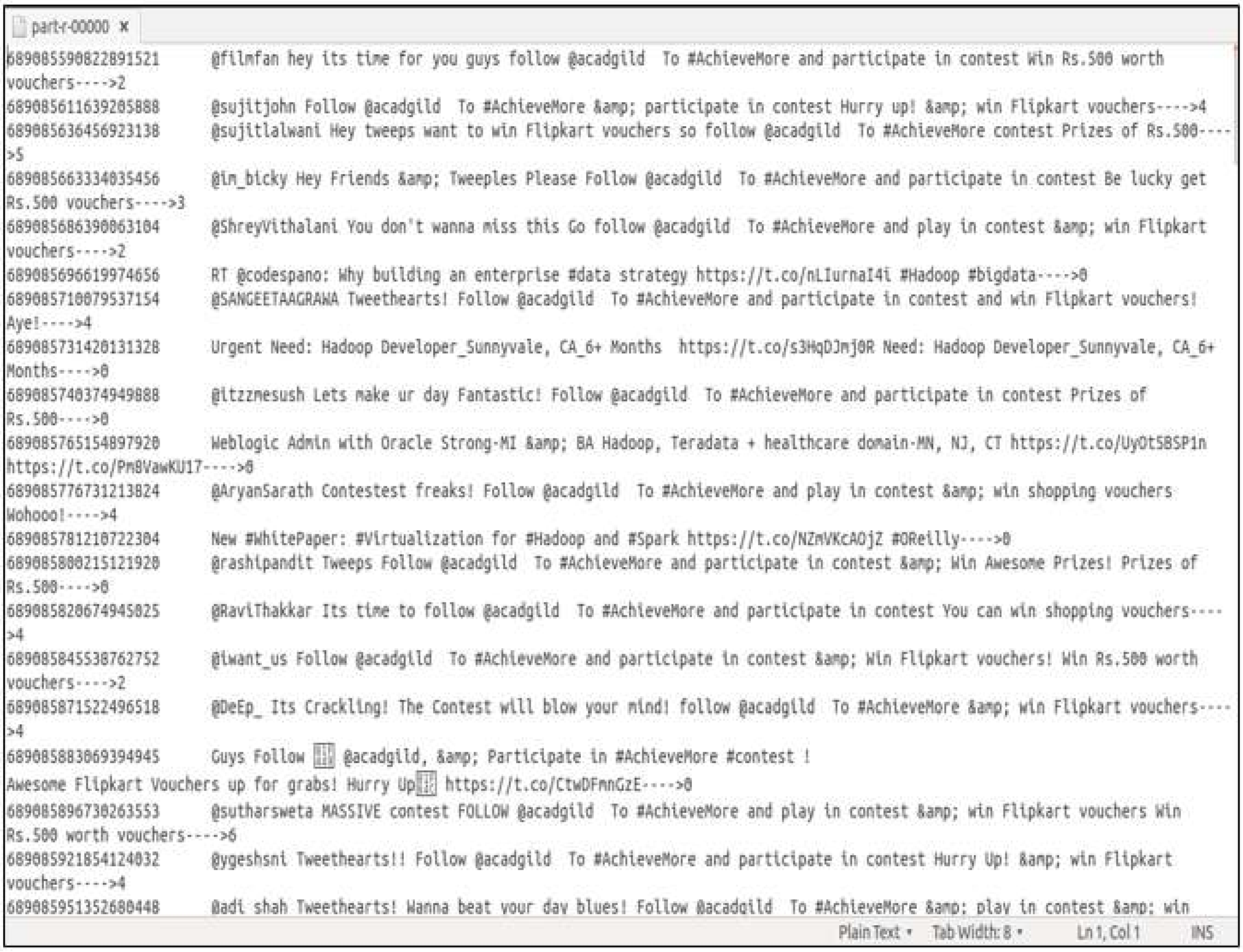


```
kiran@ACD-KIRAN:~/jars$ hadoop jar twitter.jar /user/flume/tweets /user/flume/tweets/output
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/kiran/hadoop-2.7.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: Found binding in [jar:file:/home/kiran/tez/tez/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/07/23 01:16:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicab
le
16/07/23 01:16:53 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/07/23 01:16:53 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute
your application with ToolRunner to remedy this.
16/07/23 01:16:53 INFO input.FileInputFormat: Total input paths to process : 6
16/07/23 01:16:54 INFO mapreduce.JobSubmitter: number of splits:6
16/07/23 01:16:54 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1469208332391_0003
16/07/23 01:16:54 INFO impl.YarnClientImpl: Submitted application application_1469208332391_0003
16/07/23 01:16:54 INFO mapreduce.Job: The url to track the job: http://ACD-KIRAN:8088/proxy/application_1469208332391_0003/
16/07/23 01:16:54 INFO mapreduce.Job: Running job: job_1469208332391_0003
16/07/23 01:16:59 INFO mapreduce.Job: Job job_1469208332391_0003 running in uber mode : false
16/07/23 01:16:59 INFO mapreduce.Job:  map 0% reduce 0%
16/07/23 01:17:10 INFO mapreduce.Job:  map 67% reduce 0%
16/07/23 01:17:11 INFO mapreduce.Job:  map 83% reduce 0%
16/07/23 01:17:14 INFO mapreduce.Job:  map 100% reduce 0%
16/07/23 01:17:16 INFO mapreduce.Job:  map 100% reduce 100%
16/07/23 01:17:16 INFO mapreduce.Job: Job job_1469208332391_0003 completed successfully
16/07/23 01:17:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=21084
                FILE: Number of bytes written=864608
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=654448
                HDFS: Number of bytes written=38785
                HDFS: Number of read operations=27
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=6
```

Exhibit 1: Screenshot depicting the steps to run a MapReduce program

H) SAMPLE OUTPUT

The output will be created in the part file in the directory **/user/flume/tweets/part-r-00000**.
Refer to **Exhibit 2** for the result.



```
689085590822891521	@filmfan hey its time for you guys follow @acadgild  To #AchieveMore and participate in contest Win Rs.500 worth
vouchers---->2
689085611639205888	@sujitjohn Follow @acadgild  To #AchieveMore &amp; participate in contest Hurry up! &amp; win Flipkart vouchers---->4
689085636456923138	@sujitlalwani Hey tweeps want to win Flipkart vouchers so follow @acadgild  To #AchieveMore contest Prizes of Rs.500----
>5
689085663334035456	@im_bicky Hey Friends &amp; Tweeples Please Follow @acadgild  To #AchieveMore and participate in contest Be lucky get
Rs.500 vouchers---->3
689085686390063104	@ShreyVithalani You don't wanna miss this Go follow @acadgild  To #AchieveMore and play in contest &amp; win Flipkart
vouchers---->2
689085696619974656	RT @codespano: Why building an enterprise #data strategy https://t.co/nLIurnaI4i #Hadoop #bigdata---->0
689085710079537154	@SANGEETAAGRAWA Tweethearts! Follow @acadgild  To #AchieveMore and participate in contest and win Flipkart vouchers!
Aye!---->4
689085731420131328	Urgent Need: Hadoop Developer_Sunnyvale, CA_6+ Months  https://t.co/s3HqDJmj0R Need: Hadoop Developer_Sunnyvale, CA_6+
Months---->0
689085740374949888	@itzzmesush Lets make ur day Fantastic! Follow @acadgild  To #AchieveMore and participate in contest Prizes of
Rs.500---->0
689085765154897920	Weblogic Admin with Oracle Strong-MI &amp; BA Hadoop, Teradata + healthcare domain-MN, NJ, CT https://t.co/UyOt5BSP1n
https://t.co/Pm8VawKU17---->0
689085776731213824	@AryanSarath Contestest freaks! Follow @acadgild  To #AchieveMore and play in contest &amp; win shopping vouchers
Wohooo!---->4
689085781210722304	New #WhitePaper: #Virtualization for #Hadoop and #Spark https://t.co/NZmVKcAOjZ #OReilly---->0
689085800215121920	@rashipandit Tweeps Follow @acadgild  To #AchieveMore and participate in contest &amp; Win Awesome Prizes! Prizes of
Rs.500---->0
689085820674945025	@RaviThakkar Its time to follow @acadgild  To #AchieveMore and participate in contest You can win shopping vouchers----
>4
689085845538762752	@iwant_us Follow @acadgild  To #AchieveMore and participate in contest &amp; Win Flipkart vouchers! Win Rs.500 worth
vouchers---->2
689085871522496518	@DeEp_ Its Crackling! The Contest will blow your mind! follow @acadgild  To #AchieveMore &amp; win Flipkart vouchers----
>4
689085883069394945	Guys Follow ▯ @acadgild, &amp; Participate in #AchieveMore #contest !
Awesome Flipkart Vouchers up for grabs! Hurry Up▯ https://t.co/CtwDFmnGzE---->0
689085896730263553	@sutharsweta MASSIVE contest FOLLOW @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vouchers Win
Rs.500 worth vouchers---->6
689085921854124032	@ygeshsni Tweethearts!! Follow @acadgild  To #AchieveMore and participate in contest Hurry Up! &amp; win Flipkart
vouchers---->4
689085951352680448	@adi shah Tweethearts! Wanna beat your day blues! Follow @acadgild  To #AchieveMore &amp; play in contest &amp; win
```

Exhibit 2: Screenshot of the output of sentiment analysis

We hope this section has been helpful in understanding how to perform sentiment analysis using Map Reduce.

Chapter - 4

PIG USE CASE THE DAILY SHOW DATA ANALYSIS

PART - I

Pig Use Case - The Daily Show Data Analysis Part - I

In this part of the e-book, we will be looking at the case of "**The Daily Show**." Here, we will work on some problem relations and come up with solutions using Pig scripts.

We have with us historical data of The Daily Show guests from 1999 to 2004.

Download the Dataset using this link.

- **Year:** The year an episode was aired.
- **GoogleKnowledge_Occupation:** Their occupation or office, according to Google's Knowledge Graph. On the other hand, if they are not in there, how Stewart introduced them on the program.
- **Show:** Air date of an episode. Not unique, as some shows had more than one guest.
- **Group:**A larger group designation for the occupation. For instance, US senators, US presidents, and former presidents are all under "politicians."
- **Raw_Guest_List:**The person or list of people who appeared on the show, according to Wikipedia. The GoogleKnowlege_Occupation only refers to one of them in a given row.

01 Problem Statement 1:

Find the top five kinds of GoogleKnowlege_Occupation people who were guests in the show, in a particular time period.

A) SOLUTION - Problem Statement 1

Source Code

```
A = load '/home/kiran/dialy_shows' using PigStorage(',') AS
(year:chararray,occupation:chararray,date:chararray,group:chararray,gusetlist:
chararray);

B = foreach A generate occupation,date;

C = foreach B generate occupation,ToDate(date,'MM/dd/yy') as date;

D = filter C by ((date>ToDate('1/11/99','MM/dd/yy')) AND
(date<ToDate('6/11/99','MM/dd/yy')));
#Date range can be modified by the user

E = group D by occupation;
F = foreach E generate group, COUNT(D) as cnt;
```

```
G = order F by cntdesc;

H = limit G 5;
```

- In relation**A**, we are loading the dataset using PigStorage along with the schema of the file.
- In relation**B**, we are extracting the required columns i.e., **occupation** and **date**.
- In relation**C**, we are converting the **date** in string format to date using **ToDate**function in Pig.
- In relation**D**, we are filtering the dates in a specific range.
  - Here, we have given the date range from 1/11/99 to 6/11/99 i.e., we are taking out the data for 6 months.
- In relation**E**, we are grouping relation **D** by occupation.
  - If you describe relation E,then you can see the schema of the relation as shown below:

```
describe E;
E: {group: chararray,D: {(occupation: chararray,date:datetime)}}
```

- In relation**F**, we are generating the group and the **Count** of values. Here, we will get the occupation of the guest and the number of times that occupation guest came to the show within this span of 6 months.
- In relation**G**, we are ordering the relation **F** by descending order.
- In relation**H**, we are limiting the records of relation **G** to 5.
- With this, we will get the top five GoogleKnowlege_Occupation guests in the show in a particular period.

B) SAMPLE OUTPUT - Problem Statement 1

When we dump the relation, we will get the following result:

```
(actor,28)

(actress,20)

(comedian,4)

(television actress,3)

(singer,2)
```

02 Problem Statement 2:

Find out the number of politicians who came each year.

A) SOLUTION

Source Code

```
A = load '/home/kiran/dialy_shows' using PigStorage(',') AS
(year:chararray,occupation:chararray,date:chararray,group:chararray,gusetlist:
chararray);

B = foreach A generate year,group;

C = filter B by group == 'Politician';

D = group C by year;

E = foreach D generate group, COUNT(C) as cnt;

F = order E by cntdesc;
```

- In relation**A**, we are loading the dataset using PigStorage along with the schema of the file.
- In relation**B**, we are extracting the required columns i.e., **year** and the **group**.
- In relation**C**, we are filtering the group by **Politician**.
- In relation**D**, we are grouping the relation **C** by year.
  - If you describe relation then you can see the schema of the relation as shown below:

```
describe D;
D: {group: chararray,C: {(year: chararray,group: chararray)}}
```

- In relation**E**, we are generating the group and the Count of values in the relation **C**.
- In relation**F**, we are ordering the values in the relation **F** by descending order.

B) SAMPLE OUTPUT

When we dump the relation F, we will get the number of politicians who were guests on the show each year and the result is as displayed below.

```
(2004,32)    (2010,25)    (2003,14)    (2001,3)
(2012,29)    (2011,23)    (2014,13)    (1999,2)
(2008,27)    (2005,22)    (2000,13)
(2009,26)    (2007,21)    (2013,11)
(2006,25)    (2015,14)    (2002,8)
```

Chapter – 5

PIG USE CASE THE DAILY SHOW DATA ANALYSIS

PART – II

Pig Use Case - The Daily Show Data Analysis Part - II

In this part of the e-book, we will be looking at the case of the"**The Daily Show**." Here, we will work on some problem relations and come up with solutions using Pig.

We have a historical data of The Daily Show for guests that attended the show from 1999 to 2004.

The dataset can be downloaded from here.

01 Dataset Description - The Daily Show

- **Year:** The year the episode was aired
- **GoogleKnowledge_Occupation:** Their occupation or office, according to Google's Knowledge Graph. On the other hand, if they are not in there, how Stewart introduced them on the program.
- **Show:** Air date of the episode. Not unique, as some shows had more than one guest.
- **Group:** A larger group designation for the occupation. For instance, US senators, US presidents, and former presidents are all under "politicians."
- **Raw_Guest_List:** The person or list of people who appeared on the show, according to Wikipedia. The GoogleKnowlege_Occupation only refers to one of them in a given row.

01 Problem Statement 1:

Find the number of GoogleKnowledge occupation types in each group who have been guests on the show.

A) SOLUTION

Source Code

```
A = load '/home/kiran/dialy_shows' using PigStorage(',') AS
(year:chararray,occupation:chararray,date:chararray,grp:chararray,gusetlist:ch
ararray);

B = foreach A generate occupation,grp;

C = group B by grp;

D = foreach C generate group, COUNT(B) as cnt;

E = order D by cntdesc;
```

- In relation**A**, we are loading the dataset using PigStorage along with the schema of the file.
- In relation**B**, we are extracting the required columns i.e., **occupation** and the **grp**.
- In relation**C**, we are grouping the relation **B** by the **grp**.
- If you describe the relation **C**, then you can see the schema of the relation as shown below:

```
describe C;
C: {group: chararray,B: {(occupation: chararray,grp: chararray)}}
```

- In relation**D**, we are generating the group and the **Count** of values in relation **B**.
- In relation**E**, we are displaying the count of the number of Google_knowledge_occupation types in each group, who have been guests on the show and the result is displayed below.

B) SAMPLE OUTPUT - Problem Statement 1

```
(Acting,930)
(Media,751)
(Politician,308)
(Comedy,150)
(Musician,123)
(Academic,103)
(Athletics,52)
(Misc,45)
(Government,40)
(Political Aide,36)
(NA,31)
(Science,28)
(Business,25)
(Advocacy,24)
(Consultant,18)
(Military,16)
(Clergy,8)
(media,5)
```

02 Problem Statement 2:

To verify the problem relation 1, we will find out what are the combinations of group and the Google_knowledge_occupation types who have been guests on the show.

A) SOLUTION - Problem Statement 2

Source Code

```
A = load '/home/kiran/dialy_shows' using PigStorage(',') AS
(year:chararray,occupation:chararray,date:chararray,group:chararray,gusetlist:
chararray);

B = foreach A generate occupation,group;

C = group B by (group,occupation);

D = foreach C generate group, COUNT(B) as cnt;

E = order D by group;
```

- In relation**A**,we are loading the dataset using PigStorage along with the schema of the file.
- In relation**B**,we are extracting the required columns i.e., **occupation** and the **group.**
- In relation**C**,we are grouping the relation **B** by the group and the occupation.
- If you describe relation **C**,then you can see the schema of the relation as shown below:

```
describe C;
C: {group: (group: chararray,occupation: chararray),B: {(occupation:
chararray,group: chararray)}}
```

- In relation**D**, we are generating the group and the **Count** of values in relation **B**.
- In relation**E**, we are displaying the count of the number of combinations of Google_knowledge_occupation types each group, who have been guests on the show and the sample result is displayed below.

B) SAMPLE OUTPUT - Problem Statement 2

```
((Acting,Film actor),9)
((Acting,Film actress),9)
((Acting,actor),596)
((Acting,actress),271)
((Acting,film actor),10)
((Acting,film actress),12)
((Acting,stuntperfomrer),5)
((Acting,television Actor),2)
((Acting,television actor),1)
((Acting,television actress),13)
((Acting,televison actor),1)
((Acting,telvision actor),1)
```

If you count all the combinations, you will get a total of 930 that have been displayed for **Acting** in the above problem relation.

Chapter - 6

APACHE PIG USE CASE ELECTRICAL BULB TESTING

Apache Pig Use Case - Electrical Bulb Testing

In this part of the e-book, we will work on a case involving electric bulbs and work with the date and time concepts in Pig.

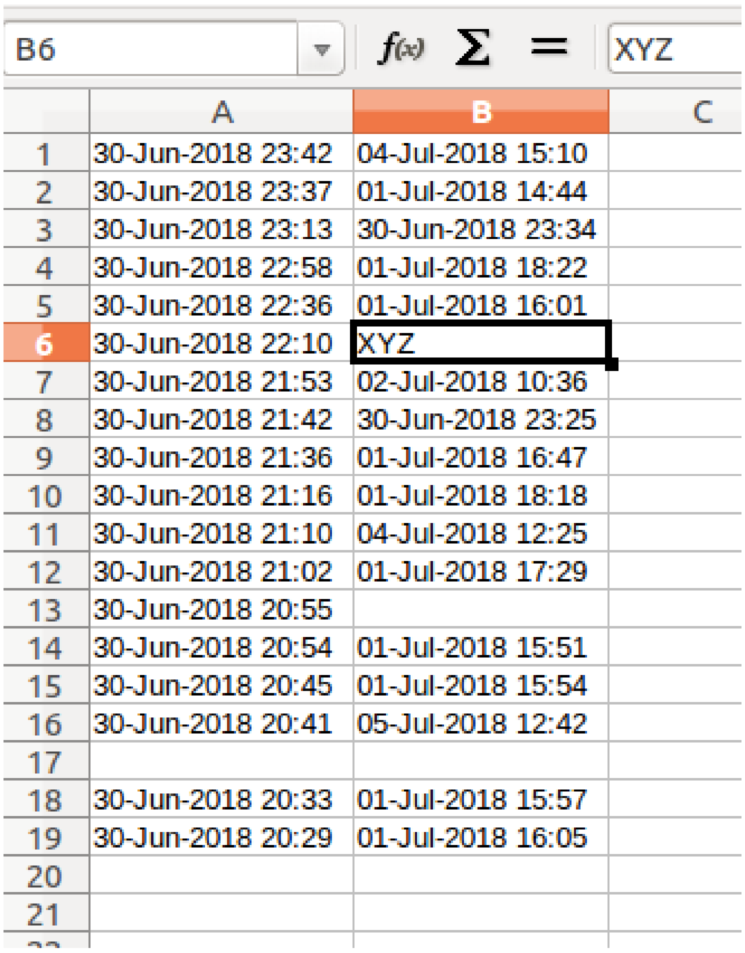
In this example, Pig is used in the local mode to load the local data. We can use Pig in HDFS mode as per our convenience.

01 Problem Statement 2:

In the research center of bulb manufacturing companies, the longevity of bulbs is tested by subjecting them to adverse conditions.

DATASET DESCRIPTION

The dataset used in this case is a sample from a light bulb production house where bulbs are tested at random intervals of time. In **Exhibit 3**, The first column is **StartDate**which is the date and time when the testing of the bulb started, and the second column is **EndDate**which is the date when the testing ended.



B6 | XYZ

	A	B	C
1	30-Jun-2018 23:42	04-Jul-2018 15:10	
2	30-Jun-2018 23:37	01-Jul-2018 14:44	
3	30-Jun-2018 23:13	30-Jun-2018 23:34	
4	30-Jun-2018 22:58	01-Jul-2018 18:22	
5	30-Jun-2018 22:36	01-Jul-2018 16:01	
6	30-Jun-2018 22:10	XYZ	
7	30-Jun-2018 21:53	02-Jul-2018 10:36	
8	30-Jun-2018 21:42	30-Jun-2018 23:25	
9	30-Jun-2018 21:36	01-Jul-2018 16:47	
10	30-Jun-2018 21:16	01-Jul-2018 18:18	
11	30-Jun-2018 21:10	04-Jul-2018 12:25	
12	30-Jun-2018 21:02	01-Jul-2018 17:29	
13	30-Jun-2018 20:55		
14	30-Jun-2018 20:54	01-Jul-2018 15:51	
15	30-Jun-2018 20:45	01-Jul-2018 15:54	
16	30-Jun-2018 20:41	05-Jul-2018 12:42	
17			
18	30-Jun-2018 20:33	01-Jul-2018 15:57	
19	30-Jun-2018 20:29	01-Jul-2018 16:05	
20			
21			

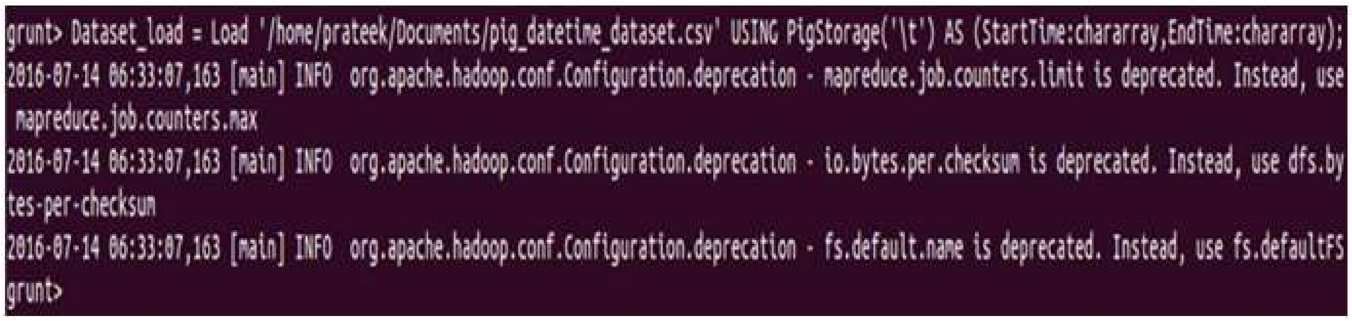
Exhibit 3: Testing schedule of bulbs from a light bulb production house

A few rows may be empty that indicates that data is not available, this may be due to various reasons. But as developers, we need not worry about the missing data. With the help of Data Filtering, we can remove the unnecessary data.

The dataset can be downloaded from here.

Loading Data into the Pig Environment

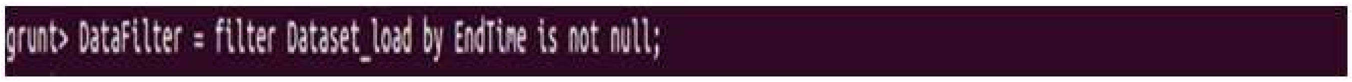
Since Pig uses default as tab(\t) delimited data, it is not mandatory to state USING PigStorage('\t') in the code while loading, nevertheless, it is good practice to write it. You have to use this parameter depending on the dataset as given in **Exhibit 4**.



```
grunt> Dataset_load = Load '/home/prateek/Documents/pig_datetime_dataset.csv' USING PigStorage('\t') AS (StartTime:chararray,EndTime:chararray);
2016-07-14 06:33:07,163 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapreduce.job.counters.limit is deprecated. Instead, use
 mapreduce.job.counters.max
2016-07-14 06:33:07,163 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.by
tes-per-checksum
2016-07-14 06:33:07,163 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

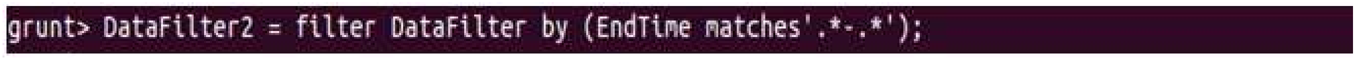
Exhibit 4: Loading data into the Pig environment

• Now that we have data inside Pig, the first step is to filter data in the column we are working on.Here, we remove all the rows with null data.



```
grunt> DataFilter = filter Dataset_load by EndTime is not null;
```

In this step, it is mandatory to filter all the data in EndTime containing the"–" symbol.



```
grunt> DataFilter2 = filter DataFilter by (EndTime matches'.*-.*');
```

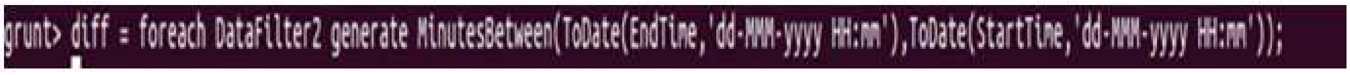
• Now that we have data inside Pig, the first step is to filter data in the column we are working on.Here, we remove all the rows with null data.

Here, we use two predefined functions.

```
ToDate()
MinutesBetween()
```

Where

```
ToDate()- Converts the character array to date-time readable structure that
can be interpreted by Pig
MinutesBetween() - Takes the difference between any two date-time parameters
provided
```



```
grunt> diff = foreach DataFilter2 generate MinutesBetween(ToDate(EndTime,'dd-MMM-yyyy HH:mm'),ToDate(StartTime,'dd-MMM-yyyy HH:mm'));
```

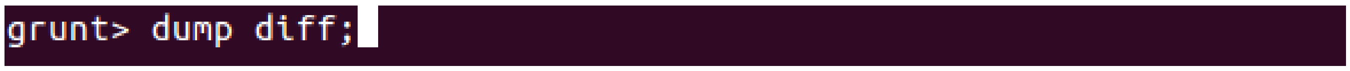
The ToDate function can be used in different formats of a year, month, and date. Example

```
YYYY-MM-DD
DD/MM/YYYY
DD-YY-MM
```

Depending on the appropriate structure in the dataset provided, we can choose the format.

After simple filtering and carrying out the conversion of the character array data to a date-time format, we now determine the difference in terms of minutes for every bulb which was in the "switched on" state during testing.

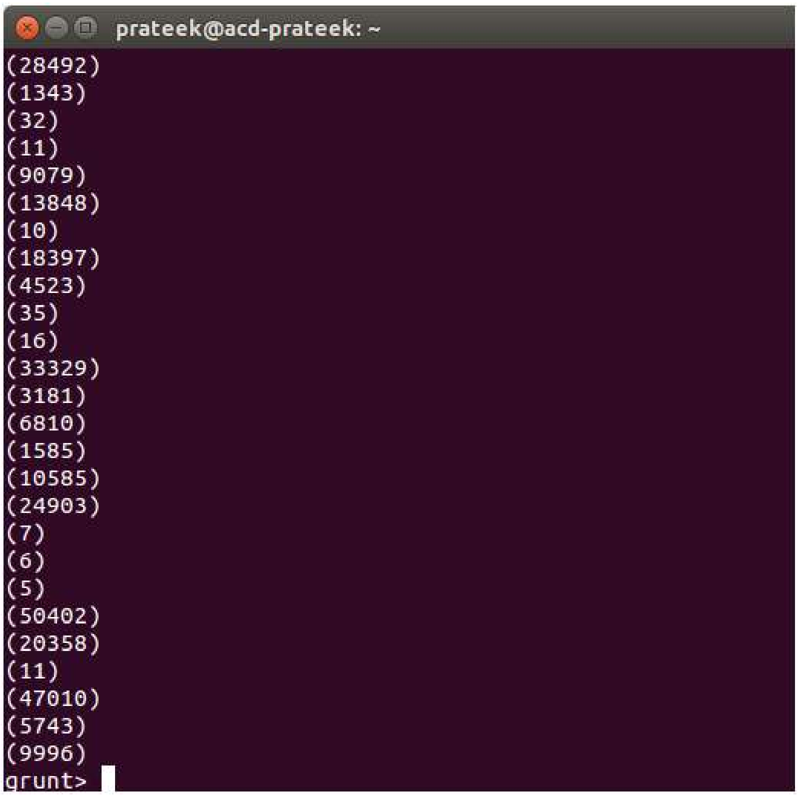
We can see the results with the dump command.



```
grunt> dump diff;
```

A) SAMPLE OUTPUT

The result is displayed in minutes **(Exhibit 5)**.

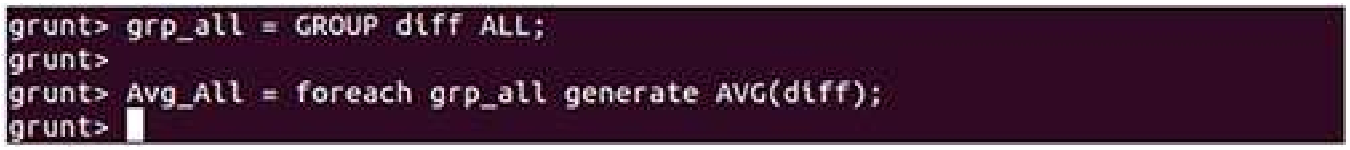


```
(28492)
(1343)
(32)
(11)
(9079)
(13848)
(10)
(18397)
(4523)
(35)
(16)
(33329)
(3181)
(6810)
(1585)
(10585)
(24903)
(7)
(6)
(5)
(50402)
(20358)
(11)
(47010)
(5743)
(9996)
grunt>
```

Exhibit 5: Result of lightbulb testing that displays the time interval for which a lightbulb stays "switched on"

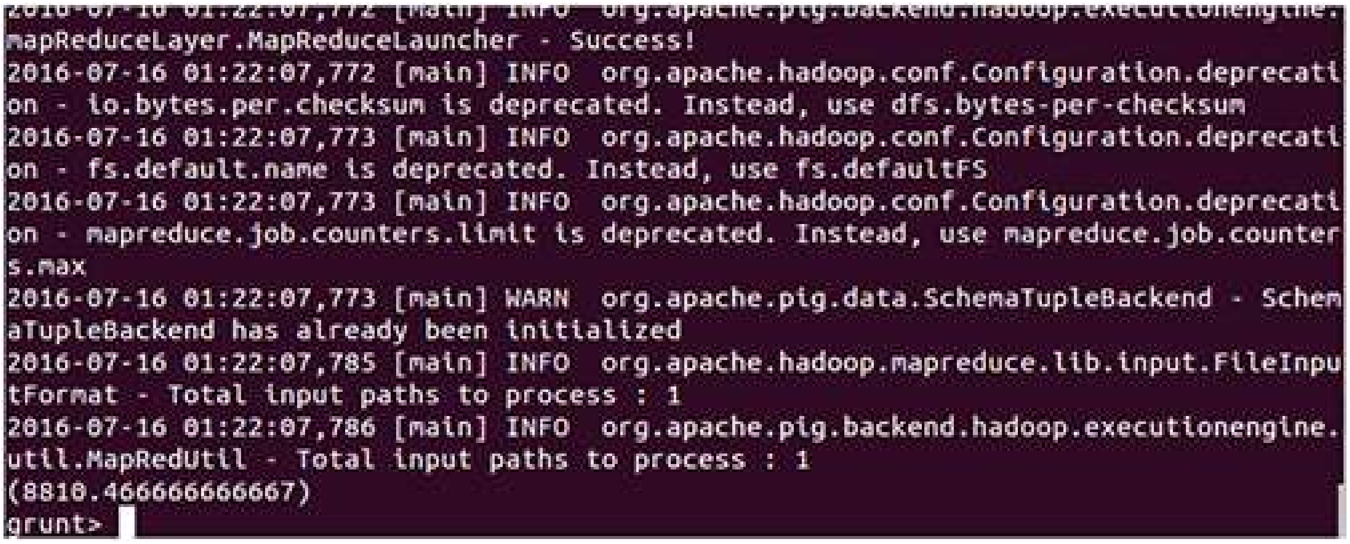
We next,analysis the result, for example, to find the maximum or minimum time a bulb can stay“switched on” among other things.

Shown below is the result for the average time for which the bulbs were “switched on” during the testing phase.



```
grunt> grp_all = GROUP diff ALL;
grunt>
grunt> Avg_All = foreach grp_all generate AVG(diff);
grunt>
```

Dump Avg_ALL;



```
2016-07-16 01:22:07,772 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
mapReduceLayer.MapReduceLauncher - Success!
2016-07-16 01:22:07,772 [main] INFO  org.apache.hadoop.conf.Configuration.deprecati
on - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-07-16 01:22:07,773 [main] INFO  org.apache.hadoop.conf.Configuration.deprecati
on - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-07-16 01:22:07,773 [main] INFO  org.apache.hadoop.conf.Configuration.deprecati
on - mapreduce.job.counters.limit is deprecated. Instead, use mapreduce.job.counter
s.max
2016-07-16 01:22:07,773 [main] WARN  org.apache.pig.data.SchemaTupleBackend - Schem
aTupleBackend has already been initialized
2016-07-16 01:22:07,785 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInpu
tFormat - Total input paths to process : 1
2016-07-16 01:22:07,786 [main] INFO  org.apache.pig.backend.hadoop.executionengine.
util.MapRedUtil - Total input paths to process : 1
(8810.466666666667)
grunt>
```

This way we can perform analysis on the filtered result and get results with the help of Pig in a matter of minutes from a large set of data

Chapter – 7

PIG USE CASE SENTIMENT ANALYSIS ON TWEETS USING AFINN DICTIONARY

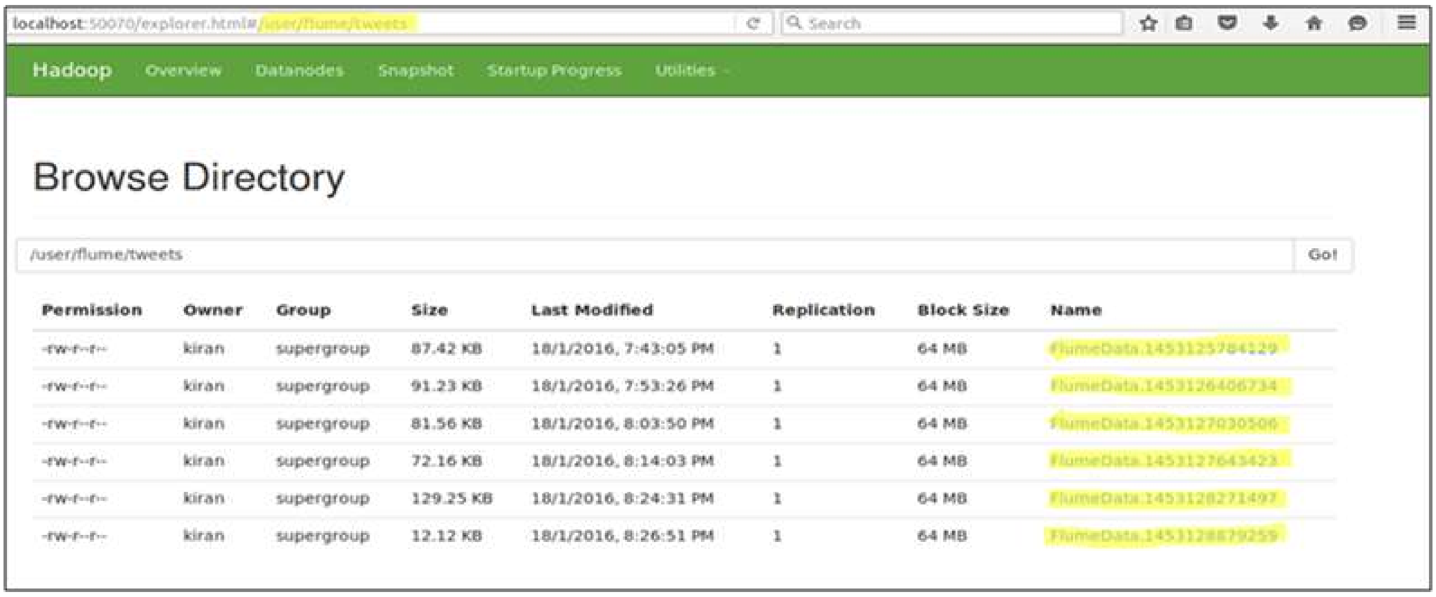
Pig Use Case - Sentiment Analysis on Tweets Using AFINN Dictionary

01 Problem Statement 1:

In this part of the e-book, we will discuss how to perform sentiment analysis on Twitter data using Pig. To begin with, we will be collecting real-time tweets from Twitter using Flume.

You can ***refer to this blog*** to get a clear idea of how to collect tweets in real time.

All the real-time tweets are kept it the location **"/user/flume/tweets"** HDFS (refer to **Exhibit 6**).



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	kiran	supergroup	87.42 KB	18/1/2016, 7:43:05 PM	1	64 MB	FlumeData.1453125784129
-rw-r--r--	kiran	supergroup	91.23 KB	18/1/2016, 7:53:26 PM	1	64 MB	FlumeData.1453126406734
-rw-r--r--	kiran	supergroup	81.56 KB	18/1/2016, 8:03:50 PM	1	64 MB	FlumeData.1453127030506
-rw-r--r--	kiran	supergroup	72.16 KB	18/1/2016, 8:14:03 PM	1	64 MB	FlumeData.1453127643423
-rw-r--r--	kiran	supergroup	129.25 KB	18/1/2016, 8:24:31 PM	1	64 MB	FlumeData.1453128271497
-rw-r--r--	kiran	supergroup	12.12 KB	18/1/2016, 8:26:51 PM	1	64 MB	FlumeData.1453128879259

Exhibit 6: Real-time tweets location

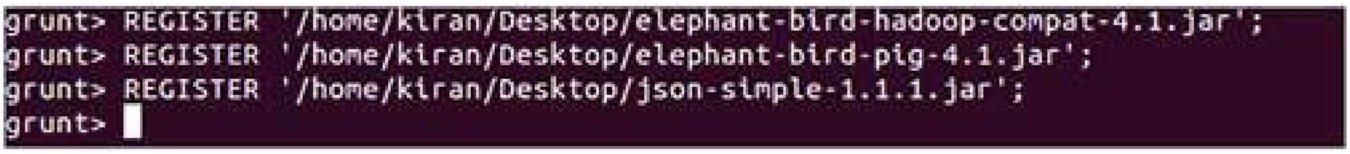
Click here to download the required JAR files.

```
REGISTER '/home/kiran/Desktop/elephant-bird-hadoop-compat-4.1.jar';

REGISTER '/home/kiran/Desktop/elephant-bird-pig-4.1.jar';

REGISTER '/home/kiran/Desktop/JSON-simple-1.1.1.jar';
```

Refer to the following screenshot for the same.



```
grunt> REGISTER '/home/kiran/Desktop/elephant-bird-hadoop-compat-4.1.jar';
grunt> REGISTER '/home/kiran/Desktop/elephant-bird-pig-4.1.jar';
grunt> REGISTER '/home/kiran/Desktop/json-simple-1.1.1.jar';
grunt> 
```

***Note:** You need to provide the path of the JAR file accordingly.

- After registering the required JARs, we can now write a Pig script to perform sentiment analysis.

Sample Tweet for Sentiment Analysis

Sample tweets collected for this purpose are displayed below.

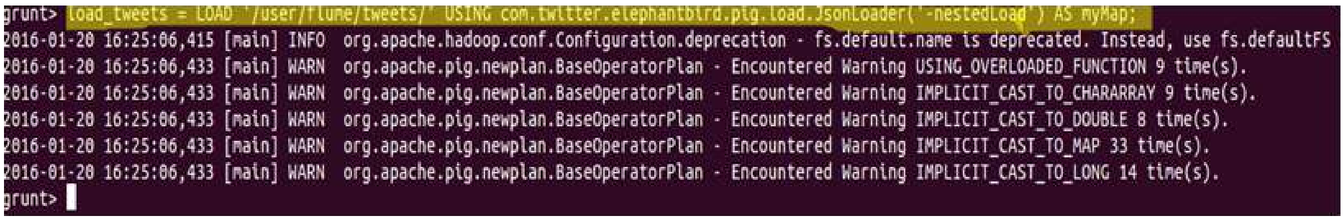
```
{"filter_level":"low","retweeted":false,"in_reply_to_screen_name":"FilmFan"
,"truncated":false,"lang":"en","in_reply_to_status_id_str":null,"id":689085
590822891521,"in_reply_to_user_id_str":"6048122","timestamp_ms":"1453125782
100","in_reply_to_status_id":null,"created_at":"Mon Jan 18 14:03:02 +0000
2016","favorite_count":0,"place":null,"coordinates":null,"text":"@filmfan
hey its time for you guys follow @acadgild To #AchieveMore and participate
in contest Win Rs.500 worth
vouchers","contributors":null,"geo":null,"entities":{"symbols":[],"urls":[]
,"hashtags":[{"text":"AchieveMore","indices":[56,68]}],"user_mentions":[{"i
d":6048122,"name":"Tanya","indices":[0,8],"screen_name":"FilmFan","id_str":
"6048122"},{"id":2649945906,"name":"ACADGILD","indices":[42,51],"screen_nam
e":"acadgild","id_str":"2649945906"}]},"is_quote_status":false,"source":"<a
href=\"https://about.twitter.com/products/tweetdeck\"
rel=\"nofollow\">TweetDeck<\/a>","favorited":false,"in_reply_to_user_id":60
48122,"retweet_count":0,"id_str":"689085590822891521","user"  "location":"I
ndia
","default_profile":false,"profile_background_tile":false,"statuses_count":
86548,"lang":"en","profile_link_color":"94D487","profile_banner_url":"https
://pbs.twimg.com/profile_banners/197865769/1436198000","id":197865769,"foll
owing":null,"protected":false,"favourites_count":1002,"profile_text_color":
"000000","verified":false,"description":"Proud Indian, Digital Marketing
Consultant,Traveler, Foodie, Adventurer, Data Architect, Movie Lover, Namo
Fan","contributors_enabled":false,"profile_sidebar_border_color":"000000","
name":"Bahubali","profile_background_color":"000000","created_at":"Sat Oct
02 17:41:02 +0000
2010","default_profile_screenshot":false,"followers_count":4467,"profile_sc
reenshot_url_https":"https://pbs.twimg.com/profile_screenshots/664486535040
000000/GOjDUiuK_normal.jpg","geo_enabled":true,"profile_background_screensh
ot_url":"http://abs.twimg.com/screenshots/themes/theme1/bg.png","profile_ba
ckground_screenshot_url_https":"https://abs.twimg.com/screenshots/themes/th
eme1/bg.png","follow_request_sent":null,"url":null,"utc_offset":19800,"time
_zone":"Chennai","notifications":null,"profile_use_background_screenshot":f
alse,"friends_count":810,"profile_sidebar_fill_color":"000000","screen_name
":"Ashok_Uppuluri","id_str":"197865769","profile_screenshot_url":"http://pb
s.twimg.com/profile_screenshots/664486535040000000/GOjDUiuK_normal.jpg","li
sted_count":50,"is_translator":false}}
```

A) SOLUTION

The tweets are in nested in the JSON format and consist of map data types. We need to load the tweets using JSONLoaderthat supports maps. In this example, we are using the**elephant bird JSONLoader**to load the tweets.

Refer to **Exhibit 7** for the first Pig relation required to load the tweets into Pig:

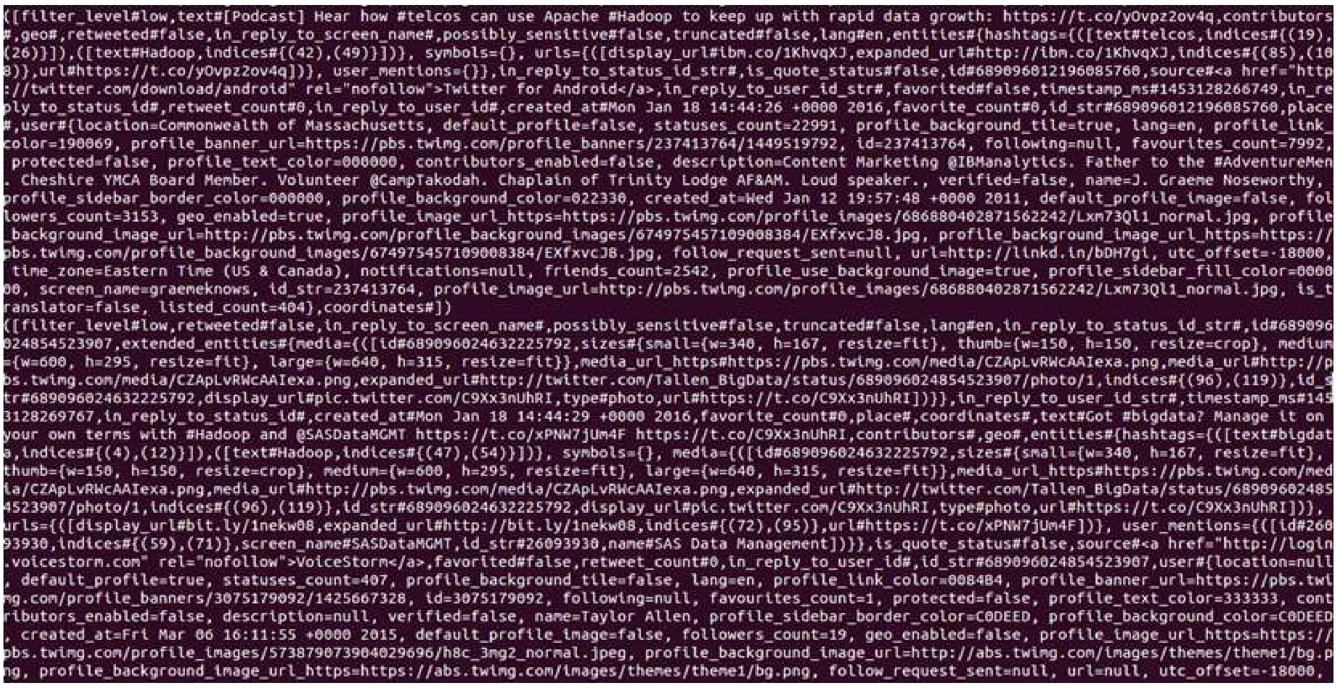
```
load_tweets = LOAD '/user/flume/tweets/' USING
com.twitter.elephantbird.pig.load.JSONLoader('-nestedLoad') AS myMap;
```



```
grunt> load_tweets = LOAD '/user/flume/tweets/' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS myMap;
2016-01-20 16:25:06,415 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-01-20 16:25:06,433 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 9 time(s).
2016-01-20 16:25:06,433 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 9 time(s).
2016-01-20 16:25:06,433 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 16:25:06,433 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 33 time(s).
2016-01-20 16:25:06,433 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt>
```

Exhibit 7: Pig relation required to load the tweets into Pig

When we dump the above relation, we can see that all the tweets got loaded successfully **(Exhibit 8)**.

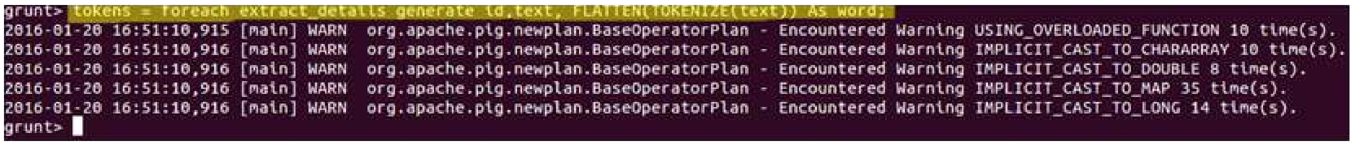


```
([filter_level#low,text#[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,contributors
#,geo#,retweeted#false,in_reply_to_screen_name#,possibly_sensitive#false,truncated#false,lang#en,entities#{hashtags={([text#telcos,indices#{(19),
(26)}]),([text#Hadoop,indices#{(42),(49)}])}, symbols={}, urls={([display_url#ibm.co/1KhvqXJ,expanded_url#http://ibm.co/1KhvqXJ,indices#{(85),(10
8)},url#https://t.co/yOvpz2ov4q])}, user_mentions={}},in_reply_to_status_id_str#,is_quote_status#false,id#689096012196085760,source#<a href="http
://twitter.com/download/android" rel="nofollow">Twitter for Android</a>,in_reply_to_user_id_str#,favorited#false,timestamp_ms#1453128266749,in_re
ply_to_status_id#,retweet_count#0,in_reply_to_user_id#,created_at#Mon Jan 18 14:44:26 +0000 2016,favorite_count#0,id_str#689096012196085760,place
#,user#{location=Commonwealth of Massachusetts, default_profile=false, statuses_count=22991, profile_background_tile=true, lang=en, profile_link_
color=190069, profile_banner_url=https://pbs.twimg.com/profile_banners/237413764/1449519792, id=237413764, following=null, favourites_count=7992,
 protected=false, profile_text_color=000000, contributors_enabled=false, description=Content Marketing @IBManalytics. Father to the #AdventureMen
. Cheshire YMCA Board Member. Volunteer @CampTakodah. Chaplain of Trinity Lodge AF&AM. Loud speaker., verified=false, name=J. Graeme Noseworthy,
profile_sidebar_border_color=000000, profile_background_color=022330, created_at=Wed Jan 12 19:57:48 +0000 2011, default_profile_image=false, fol
lowers_count=3153, geo_enabled=true, profile_image_url_https=https://pbs.twimg.com/profile_images/686880402871562242/Lxm73Ql1_normal.jpg, profile
_background_image_url=http://pbs.twimg.com/profile_background_images/674975457109008384/EXfxvcJ8.jpg, profile_background_image_url_https=https://
pbs.twimg.com/profile_background_images/674975457109008384/EXfxvcJ8.jpg, follow_request_sent=null, url=http://linkd.in/bDH7gi, utc_offset=-18000,
 time_zone=Eastern Time (US & Canada), notifications=null, friends_count=2542, profile_use_background_image=true, profile_sidebar_fill_color=0000
00, screen_name=graemeknows, id_str=237413764, profile_image_url=http://pbs.twimg.com/profile_images/686880402871562242/Lxm73Ql1_normal.jpg, is_t
ranslator=false, listed_count=404},coordinates#])
([filter_level#low,retweeted#false,in_reply_to_screen_name#,possibly_sensitive#false,truncated#false,lang#en,in_reply_to_status_id_str#,id#689096
024854523907,extended_entities#{media={([id#689096024632225792,sizes#{small={w=340, h=167, resize=fit}, thumb={w=150, h=150, resize=crop}, medium
={w=600, h=295, resize=fit}, large={w=640, h=315, resize=fit}},media_url_https#https://pbs.twimg.com/media/CZApLvRWcAAIexa.png,media_url#http://p
bs.twimg.com/media/CZApLvRWcAAIexa.png,expanded_url#http://twitter.com/Tallen_BigData/status/689096024854523907/photo/1,indices#{(96),(119)},id_s
tr#689096024632225792,display_url#pic.twitter.com/C9Xx3nUhRI,type#photo,url#https://t.co/C9Xx3nUhRI])}},in_reply_to_user_id_str#,timestamp_ms#145
3128269767,in_reply_to_status_id#,created_at#Mon Jan 18 14:44:29 +0000 2016,favorite_count#0,place#,coordinates#,text#Got #bigdata? Manage it on
your own terms with #Hadoop and @SASDataMGMT https://t.co/xPNW7jUm4F https://t.co/C9Xx3nUhRI,contributors#,geo#,entities#{hashtags={([text#bigdat
a,indices#{(4),(12)}]),([text#Hadoop,indices#{(47),(54)}])}, symbols={}, media={([id#689096024632225792,sizes#{small={w=340, h=167, resize=fit},
thumb={w=150, h=150, resize=crop}, medium={w=600, h=295, resize=fit}, large={w=640, h=315, resize=fit}},media_url_https#https://pbs.twimg.com/med
ia/CZApLvRWcAAIexa.png,media_url#http://pbs.twimg.com/media/CZApLvRWcAAIexa.png,expanded_url#http://twitter.com/Tallen_BigData/status/68909602485
4523907/photo/1,indices#{(96),(119)},id_str#689096024632225792,display_url#pic.twitter.com/C9Xx3nUhRI,type#photo,url#https://t.co/C9Xx3nUhRI])},
urls={([display_url#bit.ly/1nekw08,expanded_url#http://bit.ly/1nekw08,indices#{(72),(95)},url#https://t.co/xPNW7jUm4F])}, user_mentions={([id#260
93930,indices#{(59),(71)},screen_name#SASDataMGMT,id_str#26093930,name#SAS Data Management])}},is_quote_status#false,source#<a href="http://login
.voicestorm.com" rel="nofollow">VoiceStorm</a>,favorited#false,retweet_count#0,in_reply_to_user_id#,id_str#689096024854523907,user#{location=null
, default_profile=true, statuses_count=407, profile_background_tile=false, lang=en, profile_link_color=0084B4, profile_banner_url=https://pbs.twi
mg.com/profile_banners/3075179092/1425667328, id=3075179092, following=null, favourites_count=1, protected=false, profile_text_color=333333, cont
ributors_enabled=false, description=null, verified=false, name=Taylor Allen, profile_sidebar_border_color=C0DEED, profile_background_color=C0DEED
, created_at=Fri Mar 06 16:11:55 +0000 2015, default_profile_image=false, followers_count=19, geo_enabled=false, profile_image_url_https=https://
pbs.twimg.com/profile_images/573879073904029696/h8c_3mg2_normal.jpeg, profile_background_image_url=http://abs.twimg.com/images/themes/theme1/bg.p
ng, profile_background_image_url_https=https://abs.twimg.com/images/themes/theme1/bg.png, follow_request_sent=null, url=null, utc_offset=-18000,
```

Exhibit 8: Successfully loaded tweets in Pig

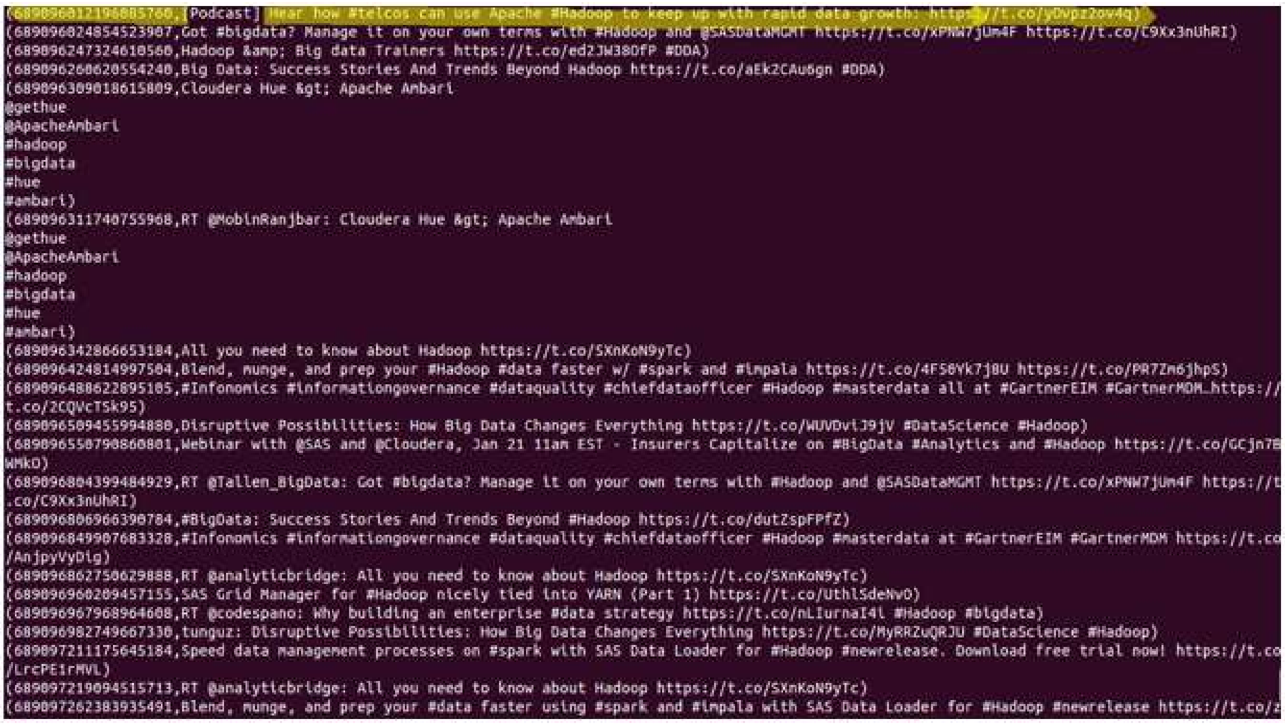
- Now, we extract the **tweet_id** and **tweeted_text** from the above tweets. The Pig relation necessary to perform this is shown below:

```
extract_details = FOREACH load_tweets GENERATE myMap#'id' asid,myMap#'text'
as text;
```



```
grunt> tokens = foreach extract_details generate id,text, FLATTEN(TOKENIZE(text)) As word;
2016-01-20 16:51:10,915 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt>
```

We can see the extracted **tweet_id** and **tweeted_text** from the tweets in Exhibit 9.

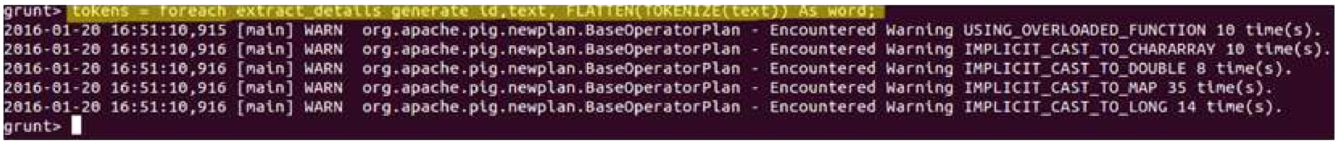


```
(689896012196885760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yDvpz2ov4q)
(689896024854523907,Got #bigdata? Manage it on your own terms with #Hadoop and @SASDataMGMT https://t.co/xPNW7jUm4F https://t.co/C9Xx3nUhRI)
(689896247324610560,Hadoop &amp; Big data Trainers https://t.co/ed2JW38DfP #DDA)
(689896268620554240,Big Data: Success Stories And Trends Beyond Hadoop https://t.co/aEk2CAu6gn #DDA)
(689896309018615809,Cloudera Hue &gt; Apache Ambari
@gethue
@ApacheAmbari
#hadoop
#bigdata
#hue
#ambari)
(689896311740755968,RT @MobinRanjbar: Cloudera Hue &gt; Apache Ambari
@gethue
@ApacheAmbari
#hadoop
#bigdata
#hue
#ambari)
(689896342866653184,All you need to know about Hadoop https://t.co/SXnKoN9yTc)
(689896424814997504,Blend, munge, and prep your #Hadoop #data faster w/ #spark and #impala https://t.co/4F58Yk7j8U https://t.co/PR7Zm6jhpS)
(689896488622895105,#Infonomics #informationgovernance #dataquality #chiefdataofficer #Hadoop #masterdata all at #GartnerEIM #GartnerMDM https://
t.co/2CQVcTSk95)
(689896509455994880,Disruptive Possibilities: How Big Data Changes Everything https://t.co/WUVDviJ9jV #DataScience #Hadoop)
(689896558790860801,Webinar with @SAS and @Cloudera, Jan 21 11am EST - Insurers Capitalize on #BigData #Analytics and #Hadoop https://t.co/GCjn7B
WMkO)
(689896804399484929,RT @Tallen_BigData: Got #bigdata? Manage it on your own terms with #Hadoop and @SASDataMGMT https://t.co/xPNW7jUm4F https://t
.co/C9Xx3nUhRI)
(689896806966390784,#BigData: Success Stories And Trends Beyond #Hadoop https://t.co/dutZspFPfZ)
(689896849907683328,#Infonomics #informationgovernance #dataquality #chiefdataofficer #Hadoop #masterdata at #GartnerEIM #GartnerMDM https://t.co
/AnjpyVyDig)
(689896862750629888,RT @analyticbridge: All you need to know about Hadoop https://t.co/SXnKoN9yTc)
(689896960209457155,SAS Grid Manager for #Hadoop nicely tied into YARN (Part 1) https://t.co/UthlSdeNvO)
(689896967968964608,RT @codespano: Why building an enterprise #data strategy https://t.co/nLIurnaI4l #Hadoop #bigdata)
(689896982749667330,tunguz: Disruptive Possibilities: How Big Data Changes Everything https://t.co/MyRRZuQRJU #DataScience #Hadoop)
(689897211175645184,Speed data management processes on #spark with SAS Data Loader for #Hadoop #newrelease. Download free trial now! https://t.co
/LrcPE1rMVL)
(689897219094515713,RT @analyticbridge: All you need to know about Hadoop https://t.co/SXnKoN9yTc)
(689897262383935491,Blend, munge, and prep your #data faster using #spark and #impala with SAS Data Loader for #Hadoop #newrelease https://t.co/z
```

Exhibit 9: Extracted id and tweet textfrom tweets loaded

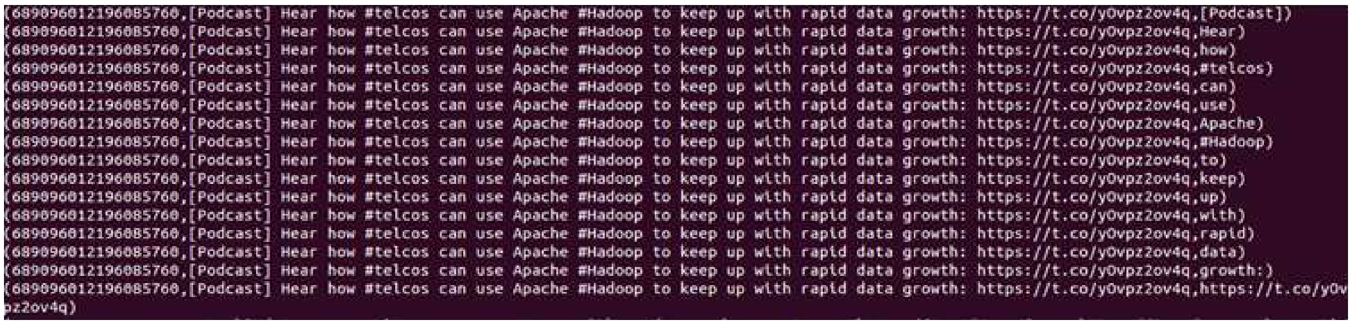
- Now, we have the tweet id and the text in the relation named as **extract_details**
- Now, we shall extract the words from the text using the TOKENIZE key word in Pig.

```
tokens = foreachextract_details generate id,text,FLATTEN(TOKENIZE(text)) As
word;
```



```
grunt> tokens = foreach extract_details generate id,text, FLATTEN(TOKENIZE(text)) As word;
2016-01-20 16:51:10,915 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt> 
```

- From **Exhibit 10**, we can see that the text has been divided into words.

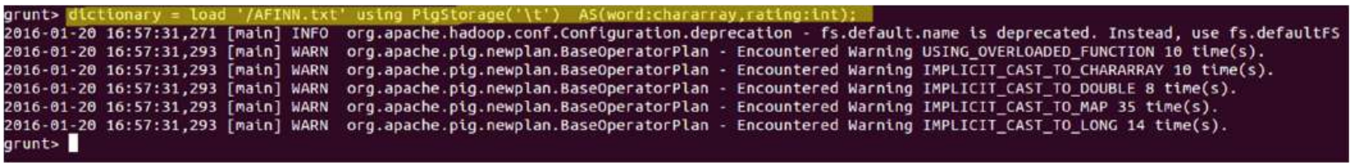


```
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,[Podcast])
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,Hear)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,how)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,#telcos)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,can)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,use)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,Apache)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,#Hadoop)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,to)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,keep)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,up)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,with)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,rapid)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,data)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,growth:)
(689096012196085760,[Podcast] Hear how #telcos can use Apache #Hadoop to keep up with rapid data growth: https://t.co/yOvpz2ov4q,https://t.co/yOv
pz2ov4q)
```

Exhibit 10: Tweet text is divided into words

- Now, we have to analyse the sentiment for the tweet by using the words in the text.
- We will rate the word as per its meaning, from +5 to -5, using the AFINN dictionary.
    - The AFINN is a dictionary which consists of 2500 words that are rated from +5 to -5 depending on their meaning. You can download ***AFINN dictionary from here***.
- We will load the dictionary into pig by using the following relation:

```
dictionary = load '/AFINN.txt' using
PigStorage('\t')AS(word:chararray,rating:int);
```



```
grunt> dictionary = load '/AFINN.txt' using PigStorage('\t')  AS(word:chararray,rating:int);
2016-01-20 16:57:31,271 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-01-20 16:57:31,293 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 16:57:31,293 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 16:57:31,293 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 16:57:31,293 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 16:57:31,293 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt> 
```

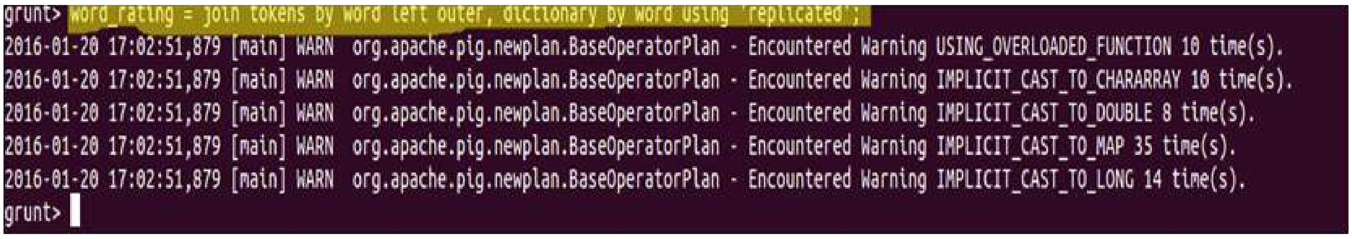
- We can see the contents of the AFINN dictionary in the **Exhibit 10**.

```
grunt> tokens = foreach extract_details generate id,text, FLATTEN(TOKENIZE(text)) As word;
2016-01-20 16:51:10,915 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 16:51:10,916 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt> 
```

Exhibit 10: Contents of the AFINN dictionary

- Now, let us perform a map side-join by joining the tokens relation and the dictionary contents using the following command:

```
word_rating = join tokens by word left outer, dictionary by wordusing
'replicated';
```

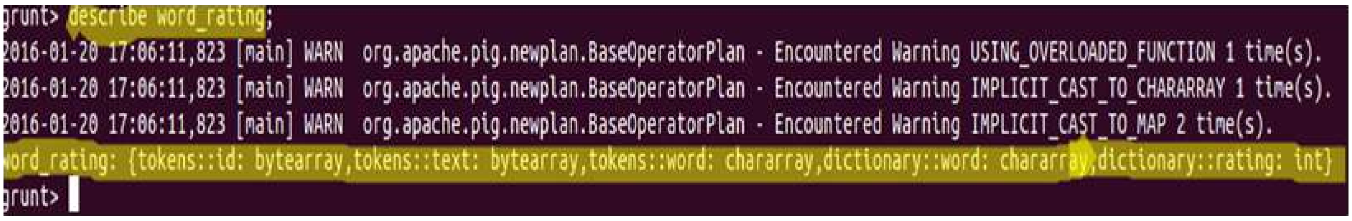


```
grunt> word_rating = join tokens by word left outer, dictionary by word using 'replicated';
2016-01-20 17:02:51,879 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 17:02:51,879 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 17:02:51,879 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 17:02:51,879 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 17:02:51,879 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt>
```

- We can see the schema of the relation after performing the join operation by using the following command:

```
describe word_rating;
```

In **Exhibit 11**, we can see that the word_rating has joined the **tokens**(consists of id, tweet text, and word) relation and the **dictionary**(consists of word and rating).

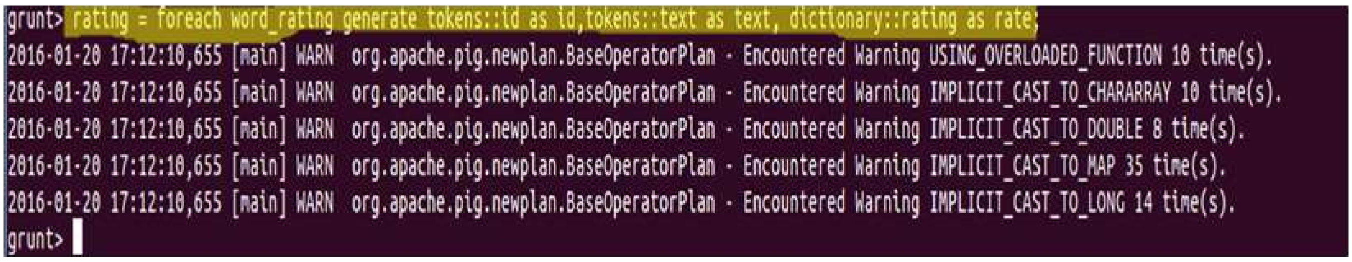


```
grunt> describe word_rating;
2016-01-20 17:06:11,823 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 1 time(s).
2016-01-20 17:06:11,823 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 1 time(s).
2016-01-20 17:06:11,823 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 2 time(s).
word_rating: {tokens::id: bytearray,tokens::text: bytearray,tokens::word: chararray,dictionary::word: chararray,dictionary::rating: int}
grunt>
```

Exhibit 11: Word_rating has joined the tokensrelation and the dictionary

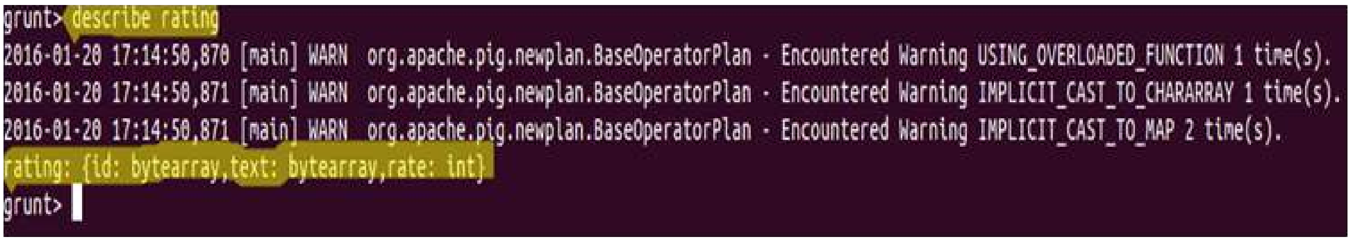
Now we will extract the tweet id, tweeted_text, and word rating (from the dictionart) by using the following relation:

```
rating = foreachword_rating generate tokens::id as id,tokens::text as
text,dictionary::rating as rate;
```



```
grunt> rating = foreach word_rating generate tokens::id as id,tokens::text as text, dictionary::rating as rate;
2016-01-20 17:12:10,655 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 17:12:10,655 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 17:12:10,655 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 17:12:10,655 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 17:12:10,655 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt>
```

We can now see the schema of the relation rating by using the command **describe rating.**
In Exhibit 12, we can see that our relation now consists of **tweet_id,tweeted_text** and **rate**(for each word).

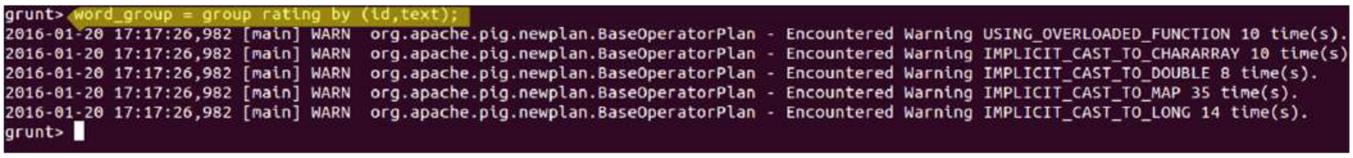


```
grunt> describe rating
2016-01-20 17:14:50,870 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 1 time(s).
2016-01-20 17:14:50,871 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 1 time(s).
2016-01-20 17:14:50,871 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 2 time(s).
rating: {id: bytearray,text: bytearray,rate: int}
grunt> 
```

Exhibit 12: Relations consisting of consists of id,tweet text and rate(for every word)

Now, we will group the rating of all the words in a tweet by using the following relation:

```
word_group = group rating by (id,text);
```



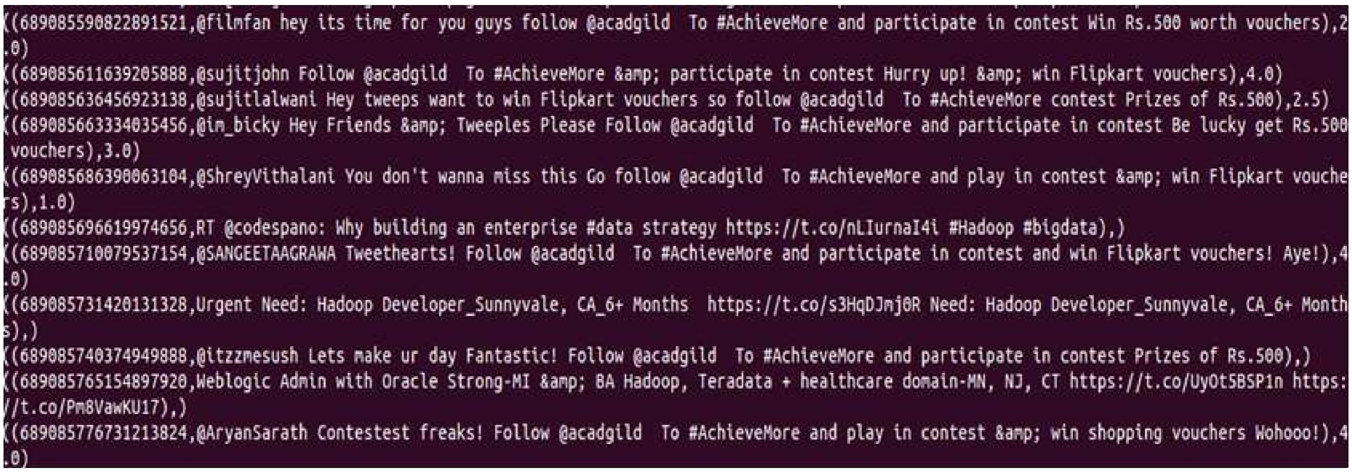
```
grunt> word_group = group rating by (id,text);
2016-01-20 17:17:26,982 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 17:17:26,982 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s)
2016-01-20 17:17:26,982 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 8 time(s).
2016-01-20 17:17:26,982 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 17:17:26,982 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt> 
```

Here we have grouped by two constraints:Tweet **id** and **text.**

Now, let's perform the Average operation on the rating of the words per each tweet.

```
avg_rate = foreachword_group generate group  AVG(rating.rate) as  tweet_rating;
```

Now, we have calculated the Average rating of tweets using the rating for every word **(Exhibit 13)**.

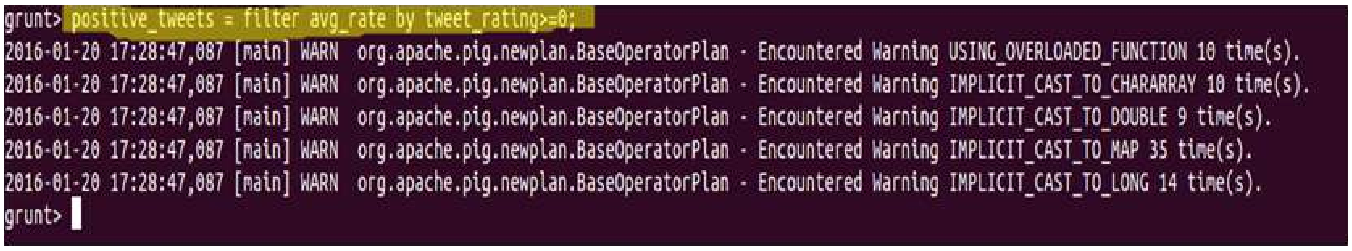


```
((689085590822891521,@filmfan hey its time for you guys follow @acadgild  To #AchieveMore and participate in contest Win Rs.500 worth vouchers),2
.0)
((689085611639205888,@sujitjohn Follow @acadgild  To #AchieveMore &amp; participate in contest Hurry up! &amp; win Flipkart vouchers),4.0)
((689085636456923138,@sujitlalwani Hey tweeps want to win Flipkart vouchers so follow @acadgild  To #AchieveMore contest Prizes of Rs.500),2.5)
((689085663334035456,@im_bicky Hey Friends &amp; Tweeples Please Follow @acadgild  To #AchieveMore and participate in contest Be lucky get Rs.500
 vouchers),3.0)
((689085686390063104,@ShreyVithalani You don't wanna miss this Go follow @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vouche
rs),1.0)
((689085696619974656,RT @codespano: Why building an enterprise #data strategy https://t.co/nLIurnaI4i #Hadoop #bigdata),)
((689085710079537154,@SANGEETAAGRAWA Tweethearts! Follow @acadgild  To #AchieveMore and participate in contest and win Flipkart vouchers! Aye!),4
.0)
((689085731420131328,Urgent Need: Hadoop Developer_Sunnyvale, CA_6+ Months  https://t.co/s3HqDJmj0R Need: Hadoop Developer_Sunnyvale, CA_6+ Month
s),)
((689085740374949888,@itzzmesush Lets make ur day Fantastic! Follow @acadgild  To #AchieveMore and participate in contest Prizes of Rs.500),)
((689085765154897920,Weblogic Admin with Oracle Strong-MI &amp; BA Hadoop, Teradata + healthcare domain-MN, NJ, CT https://t.co/UyOt5BSP1n https:
//t.co/Pm8VawKU17),)
((689085776731213824,@AryanSarath Contestest freaks! Follow @acadgild  To #AchieveMore and play in contest &amp; win shopping vouchers Wohooo!),4
.0)
```

Exhibit 13: Average rating of tweets

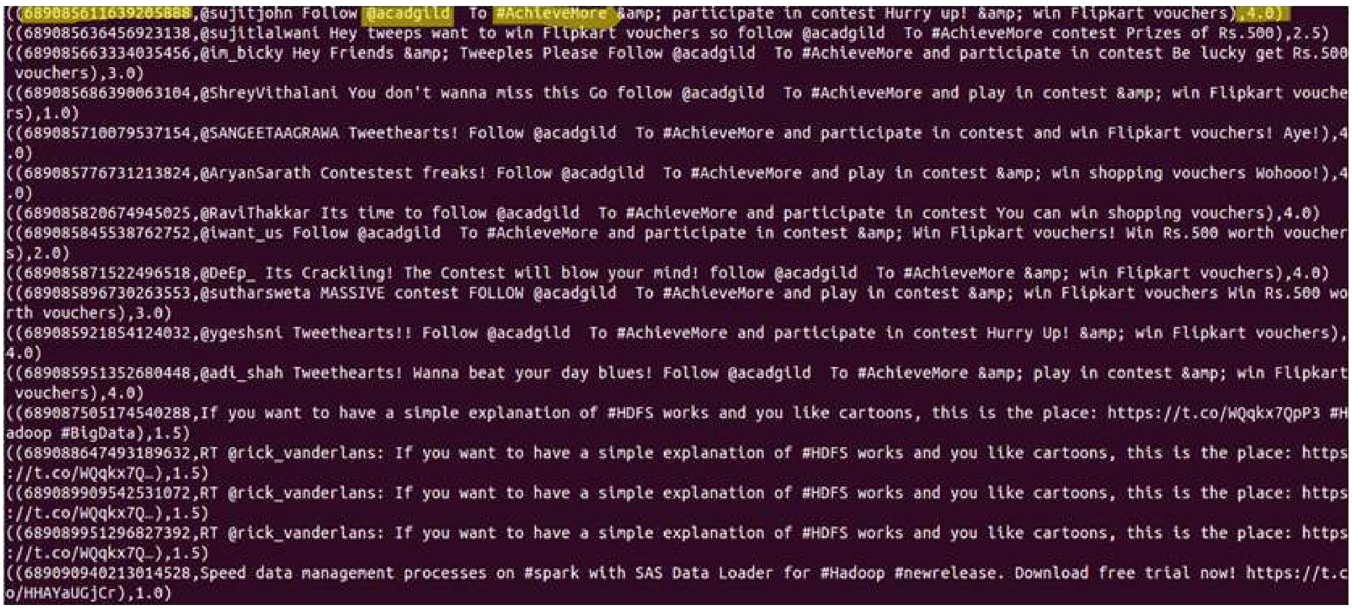
- From the above relation, we can receive all the tweets i.e., both positive and negative.
- Next, we can classify the positive tweets by taking the rating of the tweet that can range from 0–5. We can classify the negative tweets by taking the rating of the tweet from -5 to -1.
- We have now successfully performed the sentiment analysis on Twitter data using Pig.
- We will now have the tweets and its rating, so let us perform an operation to filter out the positive tweets.
- Now we will filter the positive tweets using the relation given below.

```
positive_tweets = filter avg_rate by tweet_rating>=0;
```



```
grunt> positive_tweets = filter avg_rate by tweet_rating>=0;
2016-01-20 17:28:47,087 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning USING_OVERLOADED_FUNCTION 10 time(s).
2016-01-20 17:28:47,087 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_CHARARRAY 10 time(s).
2016-01-20 17:28:47,087 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 9 time(s).
2016-01-20 17:28:47,087 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_MAP 35 time(s).
2016-01-20 17:28:47,087 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 14 time(s).
grunt>
```

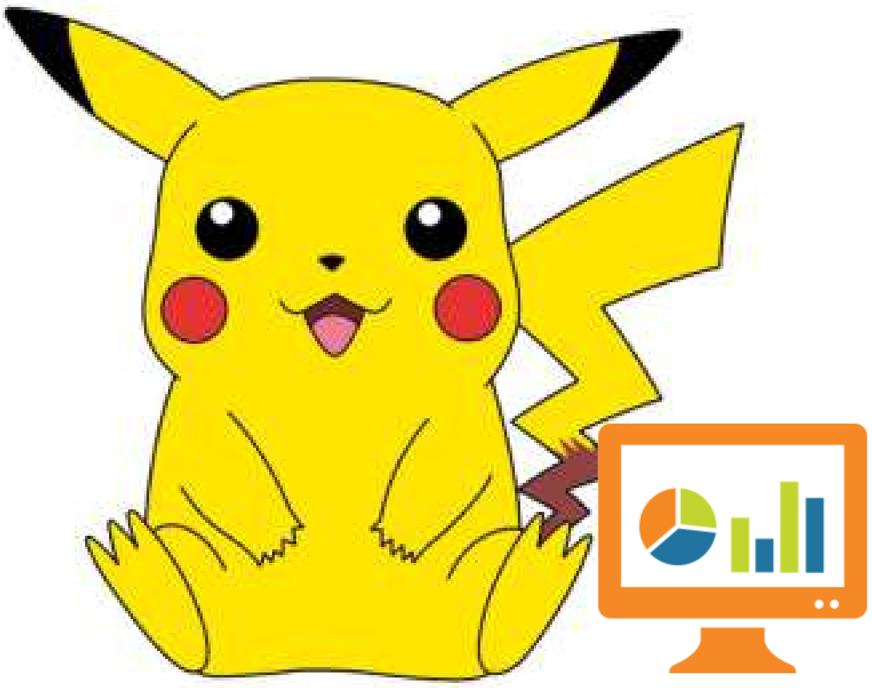
- Next, we can see the positive tweets and their rating in **Exhibit 14**.shown below.



```
((689085611639205888,@sujitjohn Follow @acadgild  To #AchieveMore &amp; participate in contest Hurry up! &amp; win Flipkart vouchers),4.0)
((689085636456923138,@sujitlalwani Hey tweeps want to win Flipkart vouchers so follow @acadgild  To #AchieveMore contest Prizes of Rs.500),2.5)
((689085663334035456,@im_bicky Hey Friends &amp; Tweeples Please Follow @acadgild  To #AchieveMore and participate in contest Be lucky get Rs.500
 vouchers),3.0)
((689085686390063104,@ShreyVithalani You don't wanna miss this Go follow @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vouche
rs),1.0)
((689085710079537154,@SANGEETAAGRAWA Tweethearts! Follow @acadgild  To #AchieveMore and participate in contest and win Flipkart vouchers! Aye!),4
.0)
((689085776731213824,@AryanSarath Contestest freaks! Follow @acadgild  To #AchieveMore and play in contest &amp; win shopping vouchers Wohooo!),4
.0)
((689085820674945025,@RaviThakkar Its time to follow @acadgild  To #AchieveMore and participate in contest You can win shopping vouchers),4.0)
((689085845538762752,@iwant_us Follow @acadgild  To #AchieveMore and participate in contest &amp; Win Flipkart vouchers! Win Rs.500 worth voucher
s),2.0)
((689085871522496518,@DeEp_ Its Crackling! The Contest will blow your mind! follow @acadgild  To #AchieveMore &amp; win Flipkart vouchers),4.0)
((689085896730263553,@sutharsweta MASSIVE contest FOLLOW @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vouchers Win Rs.500 wo
rth vouchers),3.0)
((689085921854124032,@ygeshsni Tweethearts!! Follow @acadgild  To #AchieveMore and participate in contest Hurry Up! &amp; win Flipkart vouchers),
4.0)
((689085951352680448,@adi_shah Tweethearts! Wanna beat your day blues! Follow @acadgild  To #AchieveMore &amp; play in contest &amp; win Flipkart
 vouchers),4.0)
((689087505174540288,If you want to have a simple explanation of #HDFS works and you like cartoons, this is the place: https://t.co/WQqkx7QpP3 #H
adoop #BigData),1.5)
((689088647493189632,RT @rick_vanderlans: If you want to have a simple explanation of #HDFS works and you like cartoons, this is the place: https
://t.co/WQqkx7Q…),1.5)
((689089909542531072,RT @rick_vanderlans: If you want to have a simple explanation of #HDFS works and you like cartoons, this is the place: https
://t.co/WQqkx7Q…),1.5)
((689089951296827392,RT @rick_vanderlans: If you want to have a simple explanation of #HDFS works and you like cartoons, this is the place: https
://t.co/WQqkx7Q…),1.5)
((689090940213014528,Speed data management processes on #spark with SAS Data Loader for #Hadoop #newrelease. Download free trial now! https://t.c
o/HHAYaUGjCr),1.0)
```

Exhibit 14: Here we can see the tweet_id,tweet_text, and their rating

Chapter - 8



HIVE USE CASE POKEMON DATA ANALYSIS

Hive Use Case - Pokemon Data Analysis

In this part of the e-book, we will be performing certain Hive queries to perform data analysis on the Pokémon Go characters.

What is Pokémon Go?

Pokémon Go is a free-to-play, location-based augmented reality game developed by Niantic for iOS and Android devices. It was released in July 2016 and in a few selected countries. You can download Pokémon for free of cost and start playing. You can also use PokéCoins to purchase Pokéballs, the in-game item you need to be able to catch Pokémons with.

Let us see how to performing Pokémon data analysis.

01 Dataset Description

The dataset necessary for this data analysis can be downloaded from here.

The dataset consists of 11 columns and their respective description is as follows:

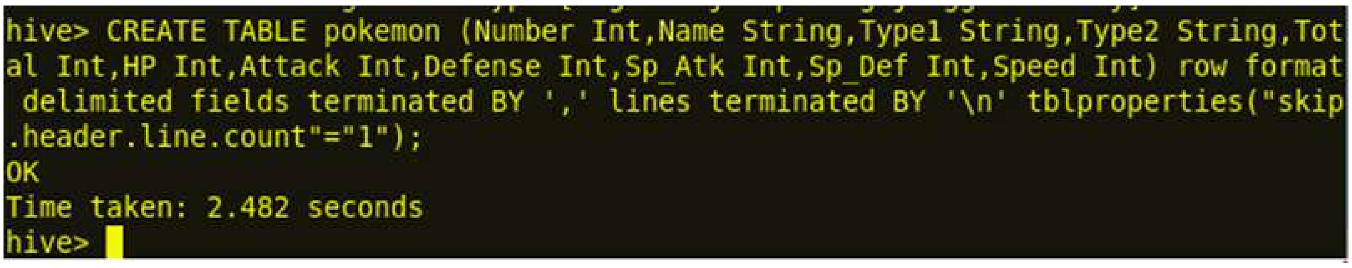
- **Pokemonid_Number:** This column represents the ids of every Pokémon.
- **Name:** This column represents the name of a Pokémon.
- **Type 1:** This column represents the property of a Pokémon.
- **Type 2:** This column represents the extended property of the same Pokémon.
    - A Pokémon may be of one or bothtypes. For instance, Charmander is a Fire type, while Bulbasaur is both a Grass type as well as a Poison type. With the current 18-type system, there are 324 possible ways to assign these types to Pokémon, along with 171 unique combinations. As of Generation VI, 133 different type combinations have been used.
- **Total:** This column represents the sum of all character points of a Pokémon (Hit Points, attack, defense, special attack, specialdefense, and speed).
- **Hit Points (HP):** This column represents Pokémon Hit Points, which is a value that determines how much damage a Pokémon can receive. When a Pokémon's HP is down to ‘0’, the Pokémon will faint. HP is the most frequently affected stat of them all, as a depleting HP is a key factor in winning a battle.
- **Attack:** This column represents the Attack stat.
- **Defense:** This column represents the Defense stat.
- **Special Attack (Sp.Atk):** This column represents a Pokémon’s Special Attack stat.

- **Special Defense (Sp. Def):** This column represents a Pokémon's Special Defense stat.
- **Speed:** This column represents the speed stat of a Pokémon.

Dataset can be downloaded from here.

- Let us begin by creating a table to hold the dataset, as shown below.

```
CREATE TABLE pokemon (Number Int,Name String,Type1 String,Type2
String,TotalInt,HPInt,AttackInt,DefenseInt,Sp_AtkInt,Sp_DefInt,SpeedInt)
row format delimited fields terminated BY ',' lines terminated BY '\n'
tblproperties("skip.header.line.count"="1");
```

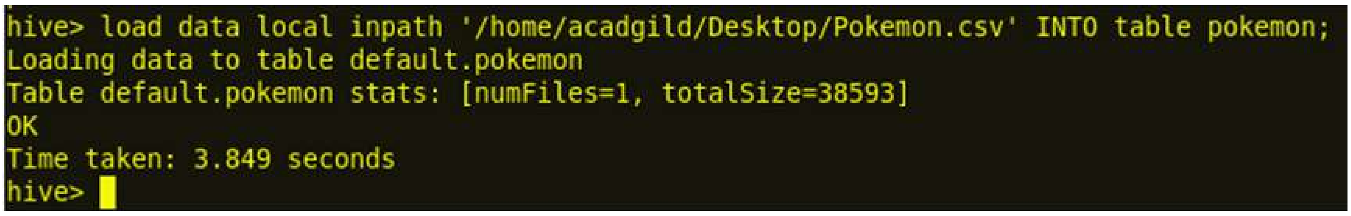


```
hive> CREATE TABLE pokemon (Number Int,Name String,Type1 String,Type2 String,Tot
al Int,HP Int,Attack Int,Defense Int,Sp_Atk Int,Sp_Def Int,Speed Int) row format
 delimited fields terminated BY ',' lines terminated BY '\n' tblproperties("skip
.header.line.count"="1");
OK
Time taken: 2.482 seconds
hive> 
```

The above command will create a table called 'pokemon', with the fields as shown in the dataset description. We have given the parameter to skip the header line, so while loading the dataset, this 'pokemon' table will ignore the header line.

- Next, let's load the dataset into the table as shown below.

```
load data local inpath '/home/acadgild/Desktop/Pokemon.csv ' INTO table
pokemon;
```



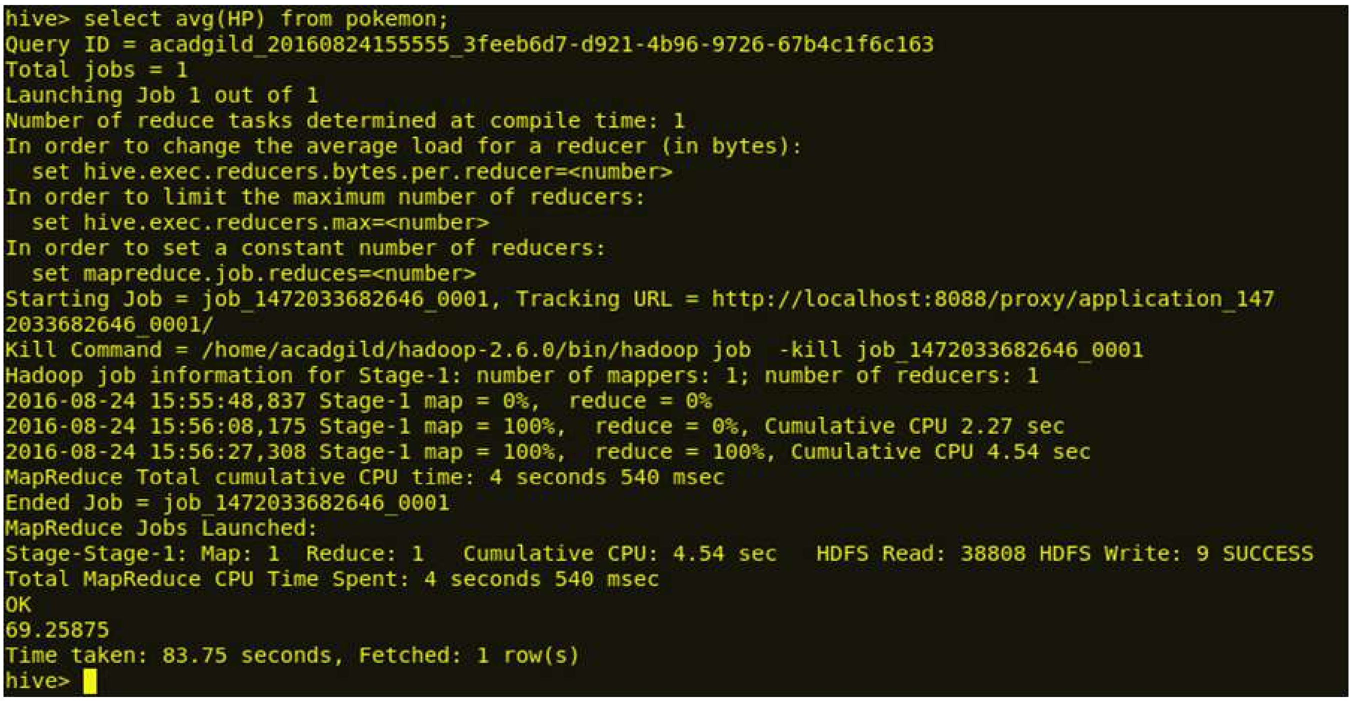
```
hive> load data local inpath '/home/acadgild/Desktop/Pokemon.csv' INTO table pokemon;
Loading data to table default.pokemon
Table default.pokemon stats: [numFiles=1, totalSize=38593]
OK
Time taken: 3.849 seconds
hive> 
```

02 Problem Statement 1:

Find out the average HP (Hit Points) of all the Pokémons, using the query mentioned below.

```
Select avg(HP) from pokemon;
```

In **Exhibit 15,** you can see that the average Hit Points of all thePokémons is **69.25875**.



```
hive> select avg(HP) from pokemon;
Query ID = acadgild_20160824155555_3feeb6d7-d921-4b96-9726-67b4c1f6c163
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0001, Tracking URL = http://localhost:8088/proxy/application_147
2033682646_0001/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 15:55:48,837 Stage-1 map = 0%,  reduce = 0%
2016-08-24 15:56:08,175 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.27 sec
2016-08-24 15:56:27,308 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.54 sec
MapReduce Total cumulative CPU time: 4 seconds 540 msec
Ended Job = job_1472033682646_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.54 sec   HDFS Read: 38808 HDFS Write: 9 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 540 msec
OK
69.25875
Time taken: 83.75 seconds, Fetched: 1 row(s)
hive>
```

Exhibit 15: Average Hit Points of all Pokémons

03 Problem Statement 2:

Create and insert values of the existing table “pokemon” into a new table “pokemon1,” with an additional column “power_rate” to find the count of “powerful” and “moderate” from the table “pokemon1.”

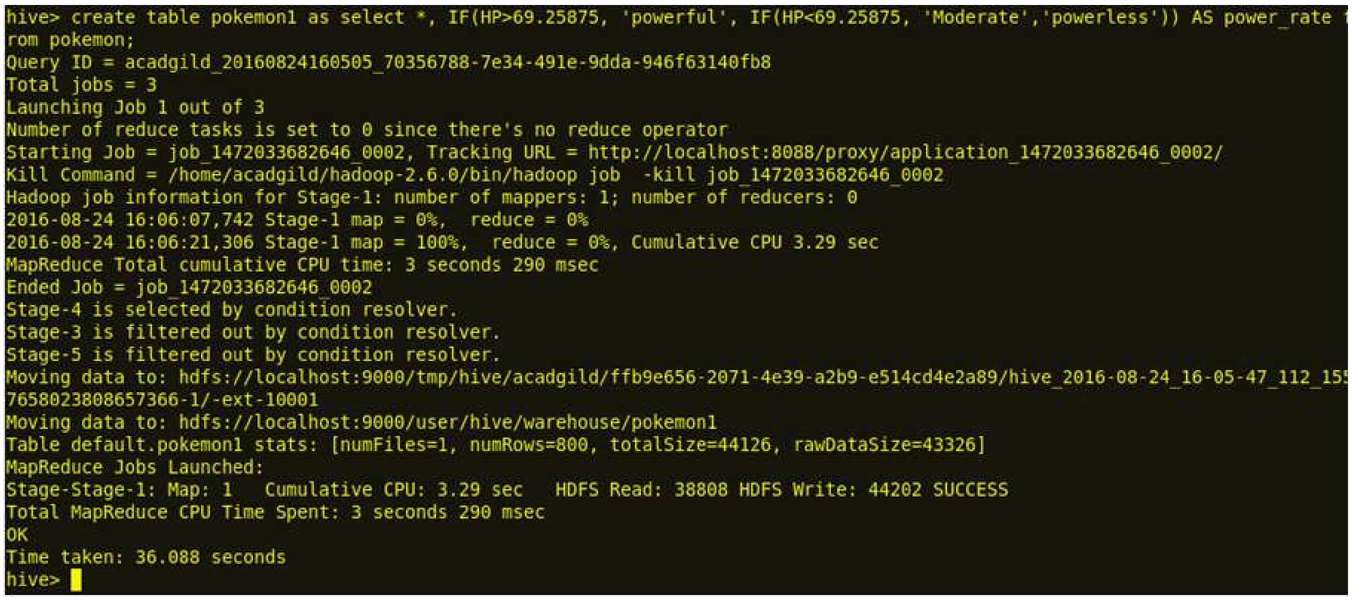
- Now, based on the average hit points, we will create another column called “power_rate.”

  In order to segregate the Pokémons, we will use if condition inside the select relation, which will create one more column in our dataset. The if condition should be used in the following manner inside a Hive query.

```
if(expr1,expr2,expr3)
```

- Now, we will create a table based on the condition that if the HP is greater than the average HP, then it is powerful, and if the HP is less than the average, then it is Moderate and a neutral condition is considered as powerless. The same is given as a Hive query below.

```
create table pokemon1 as select *, IF(HP>69.25875, 'powerful',
IF(HP<69.25875, 'Moderate','powerless')) AS power_rate from pokemon;
```



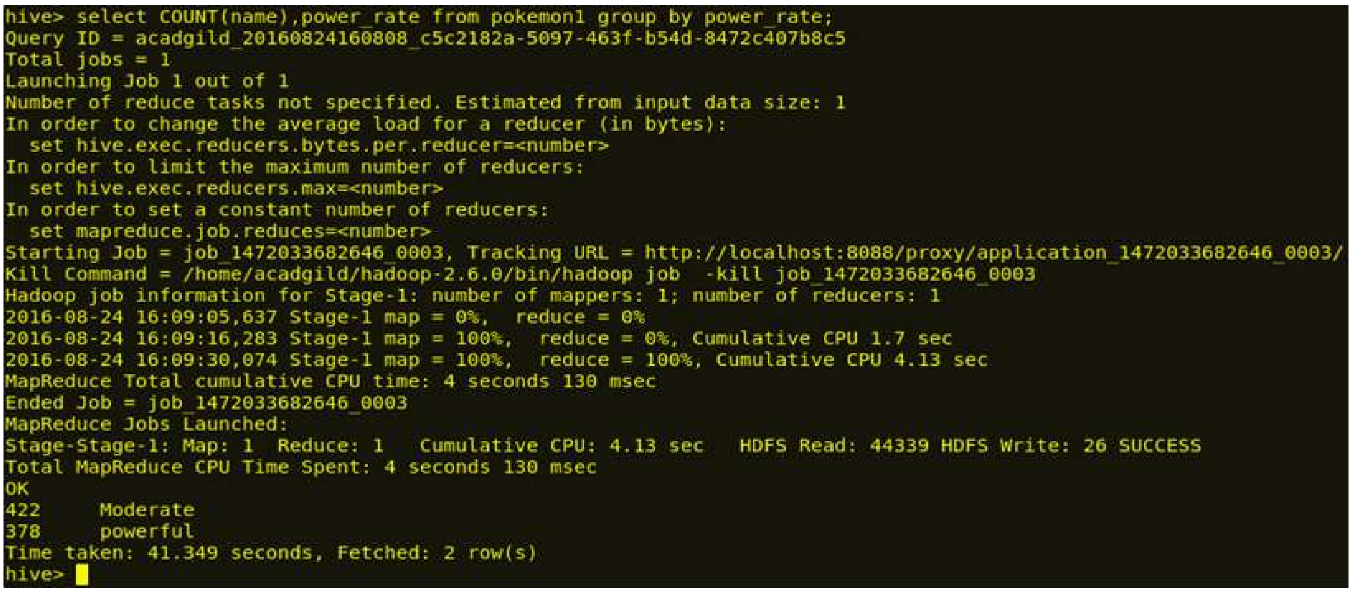
```
hive> create table pokemon1 as select *, IF(HP>69.25875, 'powerful', IF(HP<69.25875, 'Moderate','powerless')) AS power_rate f
rom pokemon;
Query ID = acadgild_20160824160505_70356788-7e34-491e-9dda-946f63140fb8
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1472033682646_0002, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0002/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-08-24 16:06:07,742 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:06:21,306 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.29 sec
MapReduce Total cumulative CPU time: 3 seconds 290 msec
Ended Job = job_1472033682646_0002
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive/acadgild/ffb9e656-2071-4e39-a2b9-e514cd4e2a89/hive_2016-08-24_16-05-47_112_155
7658023808657366-1/-ext-10001
Moving data to: hdfs://localhost:9000/user/hive/warehouse/pokemon1
Table default.pokemon1 stats: [numFiles=1, numRows=800, totalSize=44126, rawDataSize=43326]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 3.29 sec   HDFS Read: 38808 HDFS Write: 44202 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 290 msec
OK
Time taken: 36.088 seconds
hive> 
```

With the above query, a new column power_rate has been created.

- Using the following query, we will find out the number of powerful and moderate HP Pokémons present.

```
select COUNT(name),power_rate from pokemon1 group by power_rate;
```

In **Exhibit 16**, we have the result as **422 Pokémons** with moderate power and **378 Pokémons** with high power.



```
hive> select COUNT(name),power_rate from pokemon1 group by power_rate;
Query ID = acadgild_20160824160808_c5c2182a-5097-463f-b54d-8472c407b8c5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0003, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0003/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:09:05,637 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:09:16,283 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.7 sec
2016-08-24 16:09:30,074 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.13 sec
MapReduce Total cumulative CPU time: 4 seconds 130 msec
Ended Job = job_1472033682646_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.13 sec   HDFS Read: 44339 HDFS Write: 26 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 130 msec
OK
422     Moderate
378     powerful
Time taken: 41.349 seconds, Fetched: 2 row(s)
hive>
```

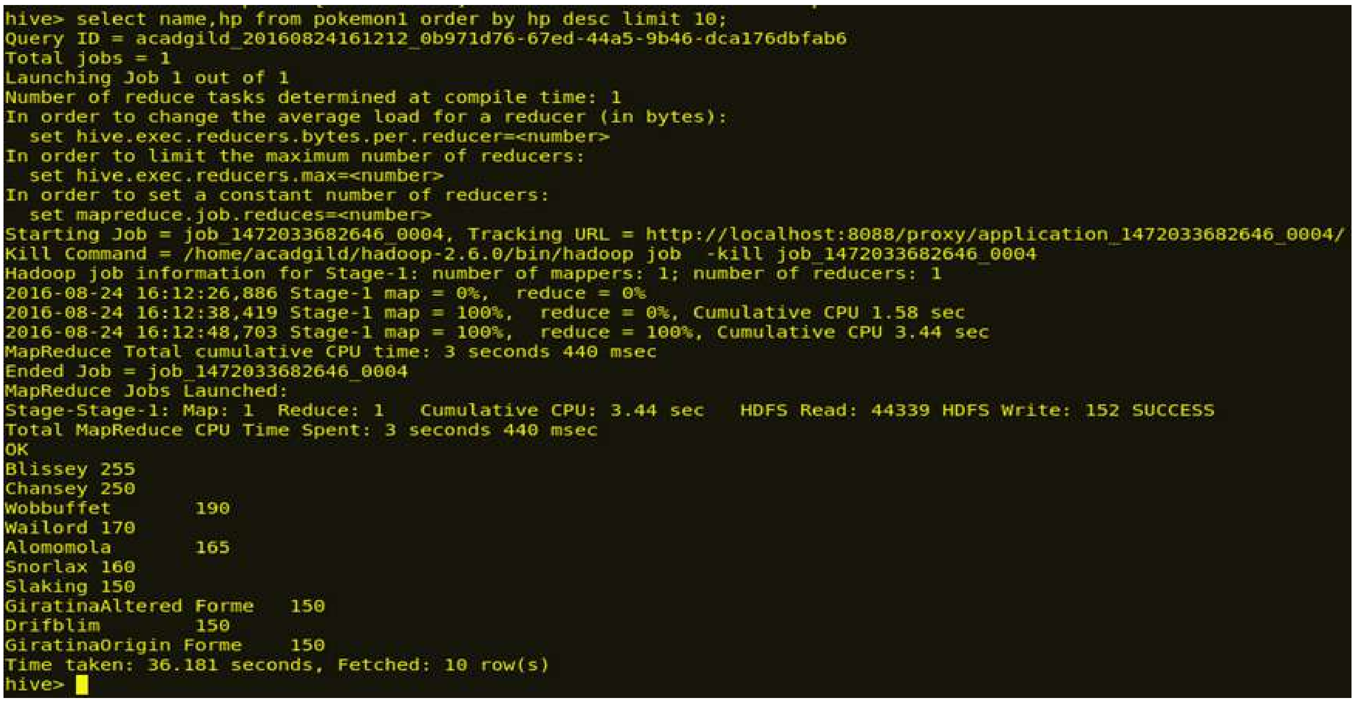
Exhibit 16: Number of powerful and moderate HP Pokémons

04 Problem Statement 3:

Find out the top 10 Pokémons according to their HP's using the below query.

```
select name,hp from pokemon1 order by hpdesc limit 10;
```

In **Exhibit 17,** we can see the list of top 10 Pokémons according to their HPs.



```
hive> select name,hp from pokemon1 order by hp desc limit 10;
Query ID = acadgild_20160824161212_0b971d76-67ed-44a5-9b46-dca176dbfab6
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0004, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0004/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:12:26,886 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:12:38,419 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.58 sec
2016-08-24 16:12:48,703 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.44 sec
MapReduce Total cumulative CPU time: 3 seconds 440 msec
Ended Job = job_1472033682646_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.44 sec   HDFS Read: 44339 HDFS Write: 152 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 440 msec
OK
Blissey 255
Chansey 250
Wobbuffet       190
Wailord 170
Alomomola       165
Snorlax 160
Slaking 150
GiratinaAltered Forme   150
Drifblim        150
GiratinaOrigin Forme    150
Time taken: 36.181 seconds, Fetched: 10 row(s)
hive>
```

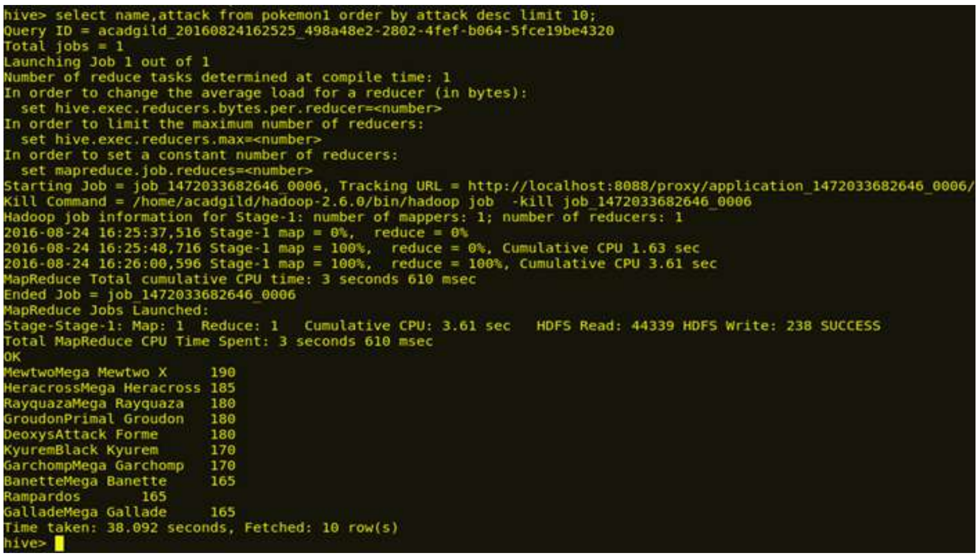
Exhibit 17: List of top 10 Pokémons according to their HPs

05 Problem Statement 4:

Find out the top 10 Pokémons based on their Attack stat, using the below query.

```
select name,attack from pokemon1 order by attack desc limit 10;
```

Exhibit 18: List of top 10 Pokémons according to their attack



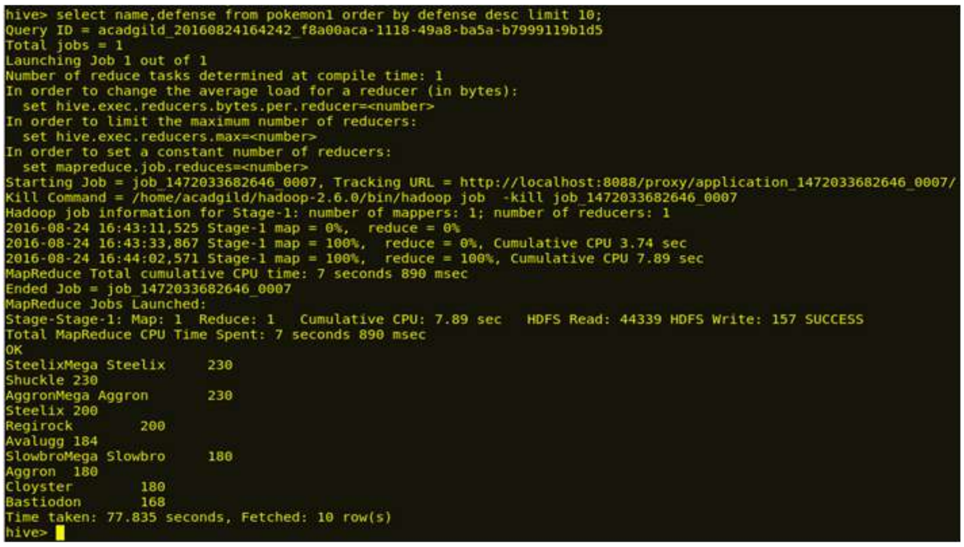
```
hive> select name,attack from pokemon1 order by attack desc limit 10;
Query ID = acadgild_20160824162525_498a48e2-2802-4fef-b064-5fce19be4320
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0006, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0006/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:25:37,516 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:25:48,716 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.63 sec
2016-08-24 16:26:00,596 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.61 sec
MapReduce Total cumulative CPU time: 3 seconds 610 msec
Ended Job = job_1472033682646_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.61 sec   HDFS Read: 44339 HDFS Write: 238 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 610 msec
OK
MewtwoMega Mewtwo X     190
HeracrossMega Heracross 185
RayquazaMega Rayquaza   180
GroudonPrimal Groudon   180
DeoxysAttack Forme      180
KyuremBlack Kyurem      170
GarchompMega Garchomp   170
BanetteMega Banette     165
Rampardos       165
GalladeMega Gallade     165
Time taken: 38.092 seconds, Fetched: 10 row(s)
hive> 
```

Exhibit 18: List of top 10 Pokémons according to their attack

06 Problem Statement 5:

Find outtop 10 Pokémons based on their defense stat byusing the following query.

```
select name,defense from pokemon1 order by defensedesc limit 10;
```



```
hive> select name,defense from pokemon1 order by defense desc limit 10;
Query ID = acadgild_20160824164242_f8a00aca-1118-49a8-ba5a-b7999119b1d5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0007, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0007/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:43:11,525 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:43:33,867 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.74 sec
2016-08-24 16:44:02,571 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.89 sec
MapReduce Total cumulative CPU time: 7 seconds 890 msec
Ended Job = job_1472033682646_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.89 sec   HDFS Read: 44339 HDFS Write: 157 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 890 msec
OK
SteelixMega Steelix     230
Shuckle 230
AggronMega Aggron       230
Steelix 200
Regirock        200
Avalugg 184
SlowbroMega Slowbro     180
Aggron  180
Cloyster        180
Bastiodon       168
Time taken: 77.835 seconds, Fetched: 10 row(s)
hive> 
```

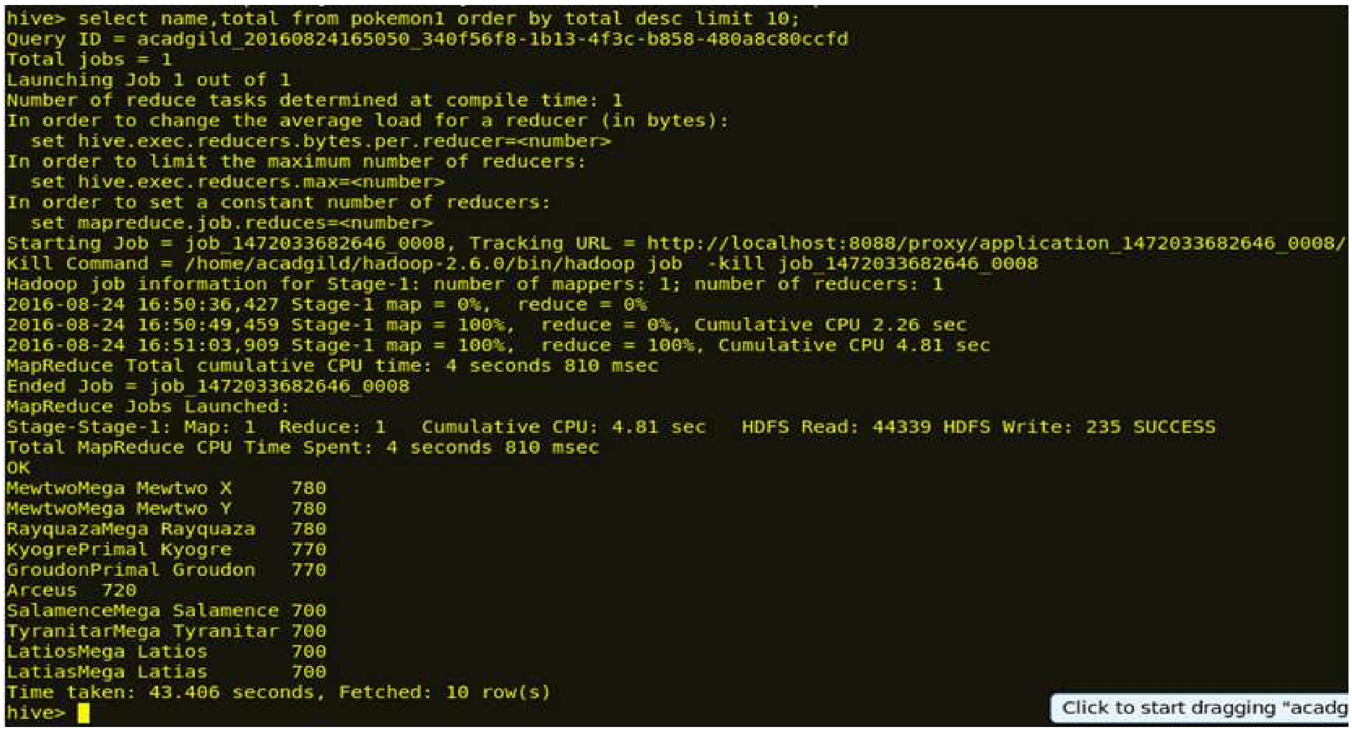
Exhibit 19: List of top 10 Pokémons according to their defense

07 Problem Statement 6:

Find out the top 10 Pokémons based on their total power by using the following query.

```
select name,total from pokemon1 order by total desc limit 10;
```

In Exhibit 20, you can see a list of top 10 Pokémons according to their all-round powers.



```
hive> select name,total from pokemon1 order by total desc limit 10;
Query ID = acadgild_20160824165050_340f56f8-1b13-4f3c-b858-480a8c80ccfd
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0008, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0008/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:50:36,427 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:50:49,459 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.26 sec
2016-08-24 16:51:03,909 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.81 sec
MapReduce Total cumulative CPU time: 4 seconds 810 msec
Ended Job = job_1472033682646_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.81 sec   HDFS Read: 44339 HDFS Write: 235 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 810 msec
OK
MewtwoMega Mewtwo X     780
MewtwoMega Mewtwo Y     780
RayquazaMega Rayquaza   780
KyogrePrimal Kyogre     770
GroudonPrimal Groudon   770
Arceus  720
SalamenceMega Salamence 700
TyranitarMega Tyranitar 700
LatiosMega Latios       700
LatiasMega Latias       700
Time taken: 43.406 seconds, Fetched: 10 row(s)
hive>
```

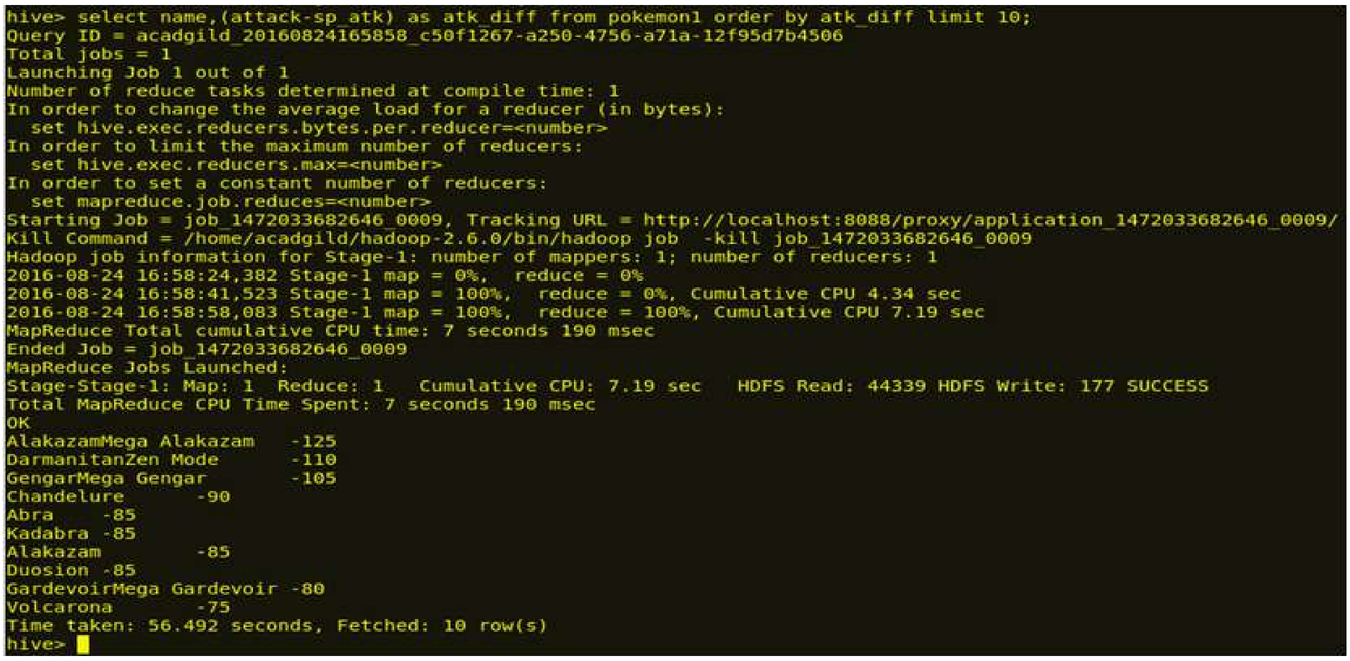
Exhibit 20: List of top 10 Pokémons according to their all-round powers

08 Problem Statement 7:

Find out the top 10 Pokémons having a drastic change in their attack and sp.attack, using the below query.

```
select name,(attack-sp_atk) as atk_diff from pokemon1 order by atk_diff
limit 10;
```

In **Exhibit 21**, a list of top 10 Pokémonsdisplaying drastic changes in their attack and special attack schemes have been displayed.



```
hive> select name,(attack-sp_atk) as atk_diff from pokemon1 order by atk_diff limit 10;
Query ID = acadgild_20160824165858_c50f1267-a250-4756-a71a-12f95d7b4506
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0009, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0009/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 16:58:24,382 Stage-1 map = 0%,  reduce = 0%
2016-08-24 16:58:41,523 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.34 sec
2016-08-24 16:58:58,083 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.19 sec
MapReduce Total cumulative CPU time: 7 seconds 190 msec
Ended Job = job_1472033682646_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.19 sec   HDFS Read: 44339 HDFS Write: 177 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 190 msec
OK
AlakazamMega Alakazam   -125
DarmanitanZen Mode      -110
GengarMega Gengar       -105
Chandelure      -90
Abra    -85
Kadabra -85
Alakazam        -85
Duosion -85
GardevoirMega Gardevoir -80
Volcarona       -75
Time taken: 56.492 seconds, Fetched: 10 row(s)
hive>
```

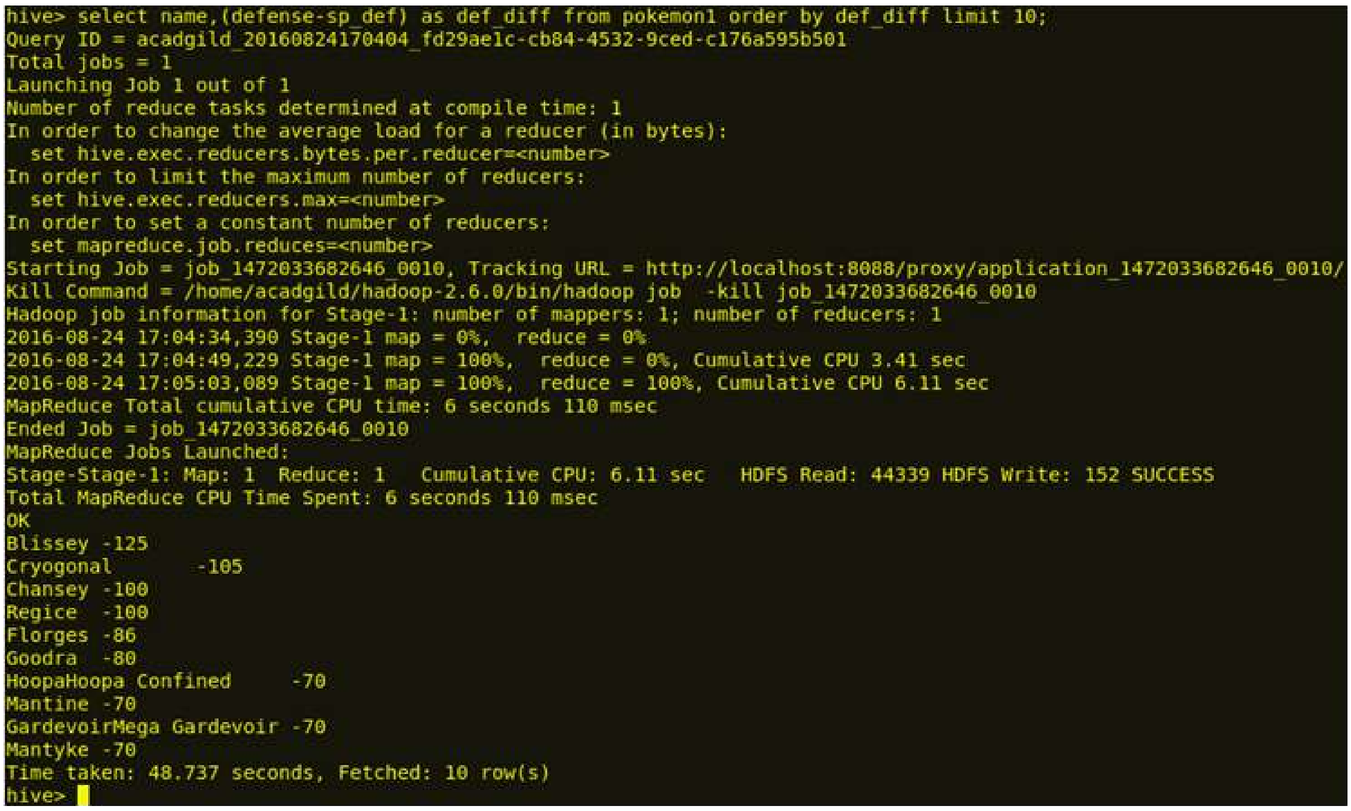
Exhibit 21: List of top 10 Pokémonsdisplaying drastic changes in their attack and special attack schemes

09 Problem Statement 8:

Find the top 10 Pokémons having a drastic change in their defense and special defense, using the below query.

```
select name,(defense-sp_defense) as def_diff from pokemon1 order by
def_diff limit 10;
```

In **Exhibit 22**, a list of top 10 Pokémons displaying drastic change in their defense and special defences has been displayed.



```
hive> select name,(defense-sp_def) as def_diff from pokemon1 order by def_diff limit 10;
Query ID = acadgild_20160824170404_fd29ae1c-cb84-4532-9ced-c176a595b501
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0010, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0010/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 17:04:34,390 Stage-1 map = 0%,  reduce = 0%
2016-08-24 17:04:49,229 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.41 sec
2016-08-24 17:05:03,089 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.11 sec
MapReduce Total cumulative CPU time: 6 seconds 110 msec
Ended Job = job_1472033682646_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.11 sec   HDFS Read: 44339 HDFS Write: 152 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 110 msec
OK
Blissey -125
Cryogonal       -105
Chansey -100
Regice  -100
Florges -86
Goodra  -80
HoopaHoopa Confined     -70
Mantine -70
GardevoirMega Gardevoir -70
Mantyke -70
Time taken: 48.737 seconds, Fetched: 10 row(s)
hive> 
```

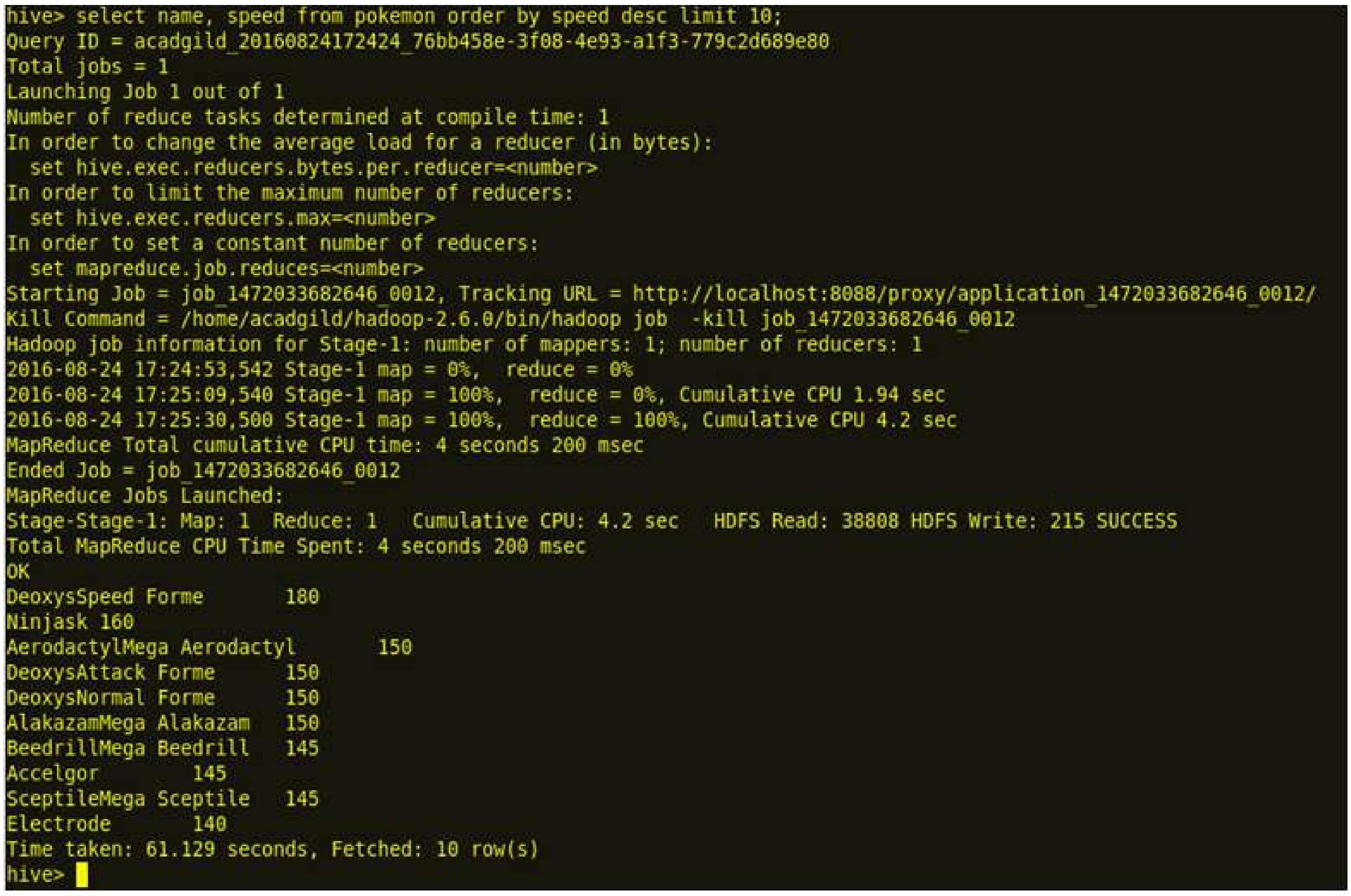
Exhibit 22: List of top 10 Pokémons displaying drastic changes in their defense and special defense schemes

10 Problem Statement 9:

Find out the fastest 10 Pokémons by using the query mentioned below.

```
Select name, speed from pokemon order by speed desc limit 10;
```

In Exhibit 23, a list of fastest 10 Pokémons have been displayed.



```
hive> select name, speed from pokemon order by speed desc limit 10;
Query ID = acadgild_20160824172424_76bb458e-3f08-4e93-a1f3-779c2d689e80
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1472033682646_0012, Tracking URL = http://localhost:8088/proxy/application_1472033682646_0012/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1472033682646_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-24 17:24:53,542 Stage-1 map = 0%,  reduce = 0%
2016-08-24 17:25:09,540 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.94 sec
2016-08-24 17:25:30,500 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.2 sec
MapReduce Total cumulative CPU time: 4 seconds 200 msec
Ended Job = job_1472033682646_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.2 sec   HDFS Read: 38808 HDFS Write: 215 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 200 msec
OK
DeoxysSpeed Forme       180
Ninjask 160
AerodactylMega Aerodactyl       150
DeoxysAttack Forme      150
DeoxysNormal Forme      150
AlakazamMega Alakazam   150
BeedrillMega Beedrill   145
Accelgor        145
SceptileMega Sceptile   145
Electrode       140
Time taken: 61.129 seconds, Fetched: 10 row(s)
hive> 
```

Exhibit 23: List of fastest 10 Pokémons

Chapter - 9

HIVE USE CASE SENTIMENT ANALYSIS ON TWEETS USING AFINN DICTIONARY

Hive Use Case - Sentiment Analysis on Tweets Using AFINN Dictionary

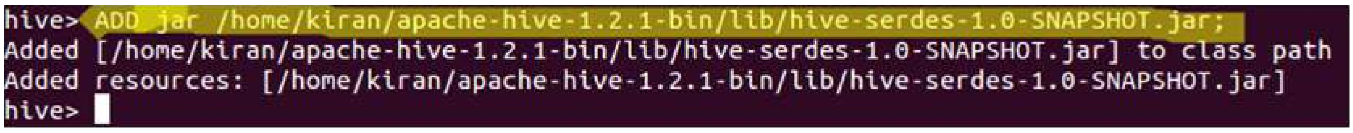
In this part of the e-book, we will see how to perform sentiment analysis on tweets collected from Twitter(using flume as a fetching tool) using Hive.

This section demonstrates how to collect tweets from Twitter in real-time.

The tweets that come in from twitter are in the"JSON" format, hence, we need to load the tweets into Hive using the JSON input format. For this,Cloudera Hive JSONSerDe has been used.

- Cloudera JSONSerDe can be downloaded using ***this link.***
- After downloading Cloudera JSONSerDe, we need to copy the JAR file into lib directory of your installed Hive folder. We need to ADD the JAR file into Hive as per the syntax shown below.

```
ADD JAR 'path of the JAR file';
```



```
hive> ADD jar /home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar]
hive>
```

After successfully adding the JAR file, we need to create a Hive table to store the Twitter data.For performing sentiment analysis, we need the tweet_id and tweet_text, for which we will create a Hive table that will extract the id and tweet_text from the tweets using the Cloudera JSONSerDe.

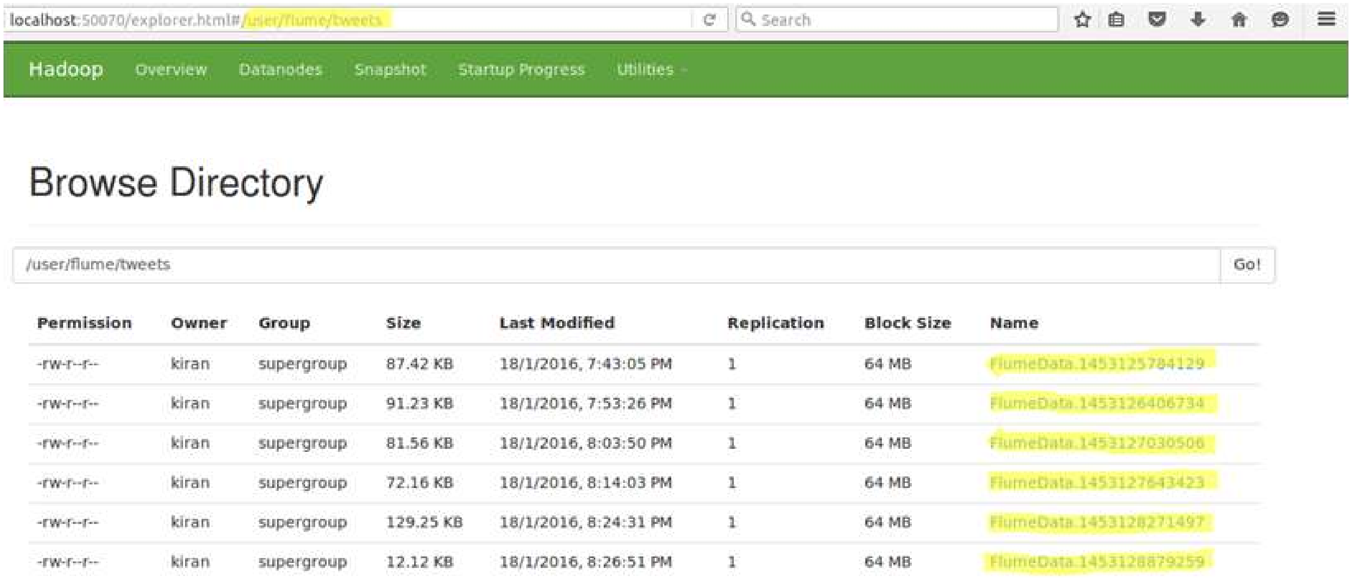
01 Sample Tweet for Sentiment Analysis

Displayed below is one of the tweets that has been collected.

```
{"filter_level":"low","retweeted":false,"in_reply_to_screen_name":"FilmFan"
,"truncated":false,"lang":"en","in_reply_to_status_id_str":null,"id":689085
590822891521,"in_reply_to_user_id_str":"6048122","timestamp_ms":"1453125782
100","in_reply_to_status_id":null,"created_at":"Mon Jan 18 14:03:02 +0000
2016","favorite_count":0,"place":null,"coordinates":null,"text":"@filmfan
hey its time for you guys follow @acadgild To #AchieveMore and participate
in contest Win Rs.500 worth
vouchers","contributors":null,"geo":null,"entities":{"symbols":[],"urls":[]
,"hashtags":[{"text":"AchieveMore","indices":[56,68]}],"user_mentions":[{"i
d":6048122,"name":"Tanya","indices":[0,8],"screen_name":"FilmFan","id_str":
"6048122"},{"id":2649945906,"name":"ACADGILD","indices":[42,51],"screen_nam
e":"acadgild","id_str":"2649945906"}]},"is_quote_status":false,"source":"<a
href=\"https://about.twitter.com/products/tweetdeck\"
rel=\"nofollow\">TweetDeck<\/a>","favorited":false,"in_reply_to_user_id":60
48122,"retweet_count":0,"id_str":"689085590822891521","user":{"location":"I
ndia
","default_profile":false,"profile_background_tile":false,"statuses_count":
86548,"lang":"en","profile_link_color":"94D487","profile_banner_url":"https
://pbs.twimg.com/profile_banners/197865769/1436198000","id":197865769,"foll
owing":null,"protected":false,"favourites_count":1002,"profile_text_color":
"000000","verified":false,"description":"Proud Indian, Digital Marketing
Consultant,Traveler, Foodie, Adventurer, Data Architect, Movie Lover, Namo
Fan","contributors_enabled":false,"profile_sidebar_border_color":"000000","
name":"Bahubali","profile_background_color":"000000","created_at":"Sat Oct
02 17:41:02 +0000
2010","default_profile_screenshot":false,"followers_count":4467,"profile_sc
reenshot_url_https":"https://pbs.twimg.com/profile_screenshots/664486535040
000000/GOjDUiuK_normal.jpg","geo_enabled":true,"profile_background_screensh
ot_url":"http://abs.twimg.com/screenshots/themes/theme1/bg.png","profile_ba
ckground_screenshot_url_https":"https://abs.twimg.com/screenshots/themes/th
eme1/bg.png","follow_request_sent":null,"url":null,"utc_offset":19800,"time
_zone":"Chennai","notifications":null,"profile_use_background_screenshot":f
alse,"friends_count":810,"profile_sidebar_fill_color":"000000","screen_name
":"Ashok_Uppuluri","id_str":"197865769","profile_screenshot_url":"http://pb
s.twimg.com/profile_screenshots/664486535040000000/GOjDUiuK_normal.jpg","li
sted_count":50,"is_translator":false}}
```

The tweet is in the nested JSON format. FFrom this tweet we will extract its id, which is the tweet_id and text, which is the tweet_text.

Our tweets are stored in the **'/user/flume/tweets/'** directory of HDFS.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	kiran	supergroup	87.42 KB	18/1/2016, 7:43:05 PM	1	64 MB	FlumeData.1453125784129
-rw-r--r--	kiran	supergroup	91.23 KB	18/1/2016, 7:53:26 PM	1	64 MB	FlumeData.1453126406734
-rw-r--r--	kiran	supergroup	81.56 KB	18/1/2016, 8:03:50 PM	1	64 MB	FlumeData.1453127030506
-rw-r--r--	kiran	supergroup	72.16 KB	18/1/2016, 8:14:03 PM	1	64 MB	FlumeData.1453127643423
-rw-r--r--	kiran	supergroup	129.25 KB	18/1/2016, 8:24:31 PM	1	64 MB	FlumeData.1453128271497
-rw-r--r--	kiran	supergroup	12.12 KB	18/1/2016, 8:26:51 PM	1	64 MB	FlumeData.1453128879259

A) SOLUTION

An external table in Hive has been created in the same directory where our tweets are present i.e., "/user/flume/tweets/," so that the tweets that are present in this location will be automatically stored in the Hive table.

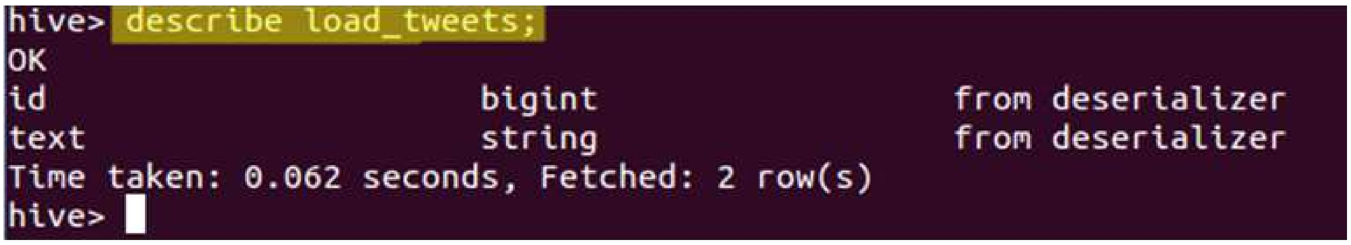
- The command for creating a Hive table to store the id and text of the tweets is as follows:

```
create external table load_tweets(id BIGINT,text STRING) ROW FORMAT SERDE
'com.cloudera.hive.serde.JSONSerDe' LOCATION '/user/flume/tweets'
```

- The schema of the table can be checked using the following command:

```
describe load_tweets;
```

In **Exhibit 24**, a Hive table with two rows ie; id and text have been created.



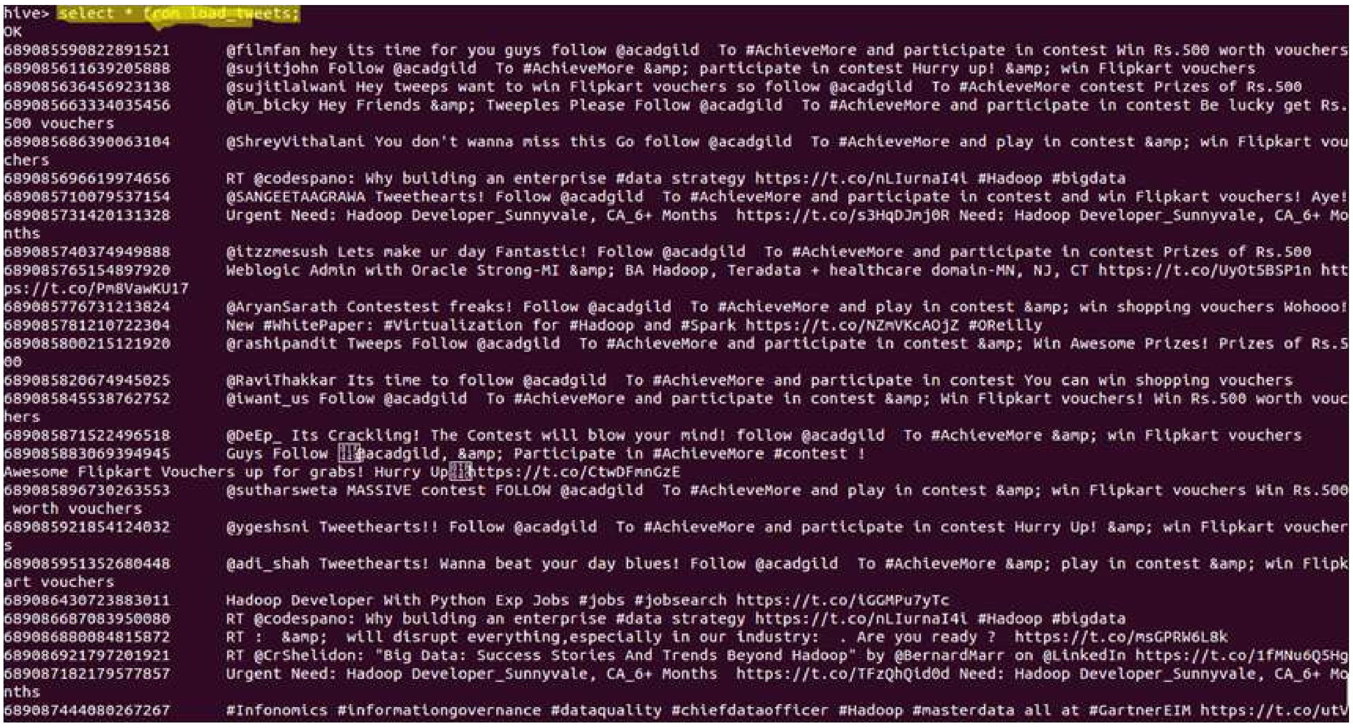
```
hive> describe load_tweets;
OK
id                      bigint                  from deserializer
text                    string                  from deserializer
Time taken: 0.062 seconds, Fetched: 2 row(s)
hive>
```

Exhibit 24: Hive table with two rows for id and text

- The tweet_id and tweet_text present in the table can be viewed using the following command.

```
select * from load_tweets;
```

In **Exhibit 25**, we can see that tweet_id and tweet_text have been successfully loaded into the table.

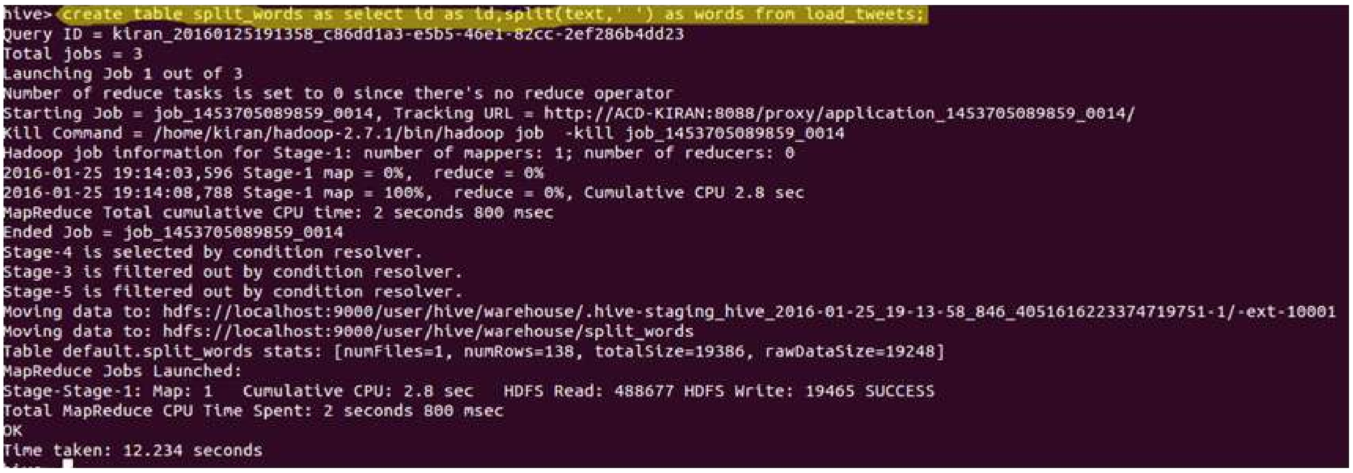


```
hive> select * from load_tweets;
OK
689085590822891521      @filmfan hey its time for you guys follow @acadgild  To #AchieveMore and participate in contest Win Rs.500 worth vouchers
689085611639205888      @sujitjohn Follow @acadgild  To #AchieveMore &amp; participate in contest Hurry up! &amp; win Flipkart vouchers
689085636456923138      @sujitlalwani Hey tweeps want to win Flipkart vouchers so follow @acadgild  To #AchieveMore contest Prizes of Rs.500
689085663334035456      @im_bicky Hey Friends &amp; Tweeples Please Follow @acadgild  To #AchieveMore and participate in contest Be lucky get Rs.
500 vouchers
689085686390063104      @ShreyVithalani You don't wanna miss this Go follow @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vou
chers
689085696619974656      RT @codespano: Why building an enterprise #data strategy https://t.co/nLIurnaI4l #Hadoop #bigdata
689085710079537154      @SANGEETAAGRAWA Tweethearts! Follow @acadgild  To #AchieveMore and participate in contest and win Flipkart vouchers! Aye!
689085731420131328      Urgent Need: Hadoop Developer_Sunnyvale, CA_6+ Months  https://t.co/s3HqDJmj0R Need: Hadoop Developer_Sunnyvale, CA_6+ Mo
nths
689085740374949888      @itzzmesush Lets make ur day Fantastic! Follow @acadgild  To #AchieveMore and participate in contest Prizes of Rs.500
689085765154897920      Weblogic Admin with Oracle Strong-MI &amp; BA Hadoop, Teradata + healthcare domain-MN, NJ, CT https://t.co/UyOt5BSP1n htt
ps://t.co/Pm8VawKU17
689085776731213824      @AryanSarath Contestest freaks! Follow @acadgild  To #AchieveMore and play in contest &amp; win shopping vouchers Wohooo!
689085781210722304      New #WhitePaper: #Virtualization for #Hadoop and #Spark https://t.co/NZmVKcAOjZ #OReilly
689085800215121920      @rashipandit Tweeps Follow @acadgild  To #AchieveMore and participate in contest &amp; Win Awesome Prizes! Prizes of Rs.5
00
689085820674945025      @RaviThakkar Its time to follow @acadgild  To #AchieveMore and participate in contest You can win shopping vouchers
689085845538762752      @iwant_us Follow @acadgild  To #AchieveMore and participate in contest &amp; Win Flipkart vouchers! Win Rs.500 worth vouc
hers
689085871522496518      @DeEp_ Its Crackling! The Contest will blow your mind! follow @acadgild  To #AchieveMore &amp; win Flipkart vouchers
689085883069394945      Guys Follow @acadgild, &amp; Participate in #AchieveMore #contest !
Awesome Flipkart Vouchers up for grabs! Hurry Up https://t.co/CtwDFmnGzE
689085896730263553      @sutharsweta MASSIVE contest FOLLOW @acadgild  To #AchieveMore and play in contest &amp; win Flipkart vouchers Win Rs.500
 worth vouchers
689085921854124032      @ygeshsni Tweethearts!! Follow @acadgild  To #AchieveMore and participate in contest Hurry Up! &amp; win Flipkart voucher
s
689085951352680448      @adi_shah Tweethearts! Wanna beat your day blues! Follow @acadgild  To #AchieveMore &amp; play in contest &amp; win Flipk
art vouchers
689086430723883011      Hadoop Developer With Python Exp Jobs #jobs #jobsearch https://t.co/iGGMPu7yTc
689086687083950080      RT @codespano: Why building an enterprise #data strategy https://t.co/nLIurnaI4l #Hadoop #bigdata
689086880084815872      RT :  &amp;  will disrupt everything,especially in our industry:  . Are you ready ?  https://t.co/msGPRW6L8k
689086921797201921      RT @CrShelidon: "Big Data: Success Stories And Trends Beyond Hadoop" by @BernardMarr on @LinkedIn https://t.co/1fMNu6Q5Hg
689087182179577857      Urgent Need: Hadoop Developer_Sunnyvale, CA_6+ Months  https://t.co/TFzQhQid0d Need: Hadoop Developer_Sunnyvale, CA_6+ Mo
nths
689087444080267267      #Infonomics #informationgovernance #dataquality #chiefdataofficer #Hadoop #masterdata all at #GartnerEIM https://t.co/utV
```

Exhibit 25: Table with tweet_id and tweet_text

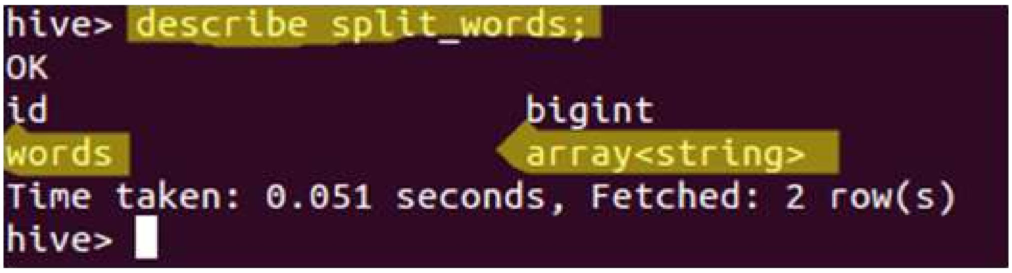
- The text can be split into words using the **split()** UDF available in Hive. If the split() function to split the text as words is used, it will return an array of values. Hence, another Hive table is created to store the tweet_id and the array of words.

```
create table split_words as select id as id,split(text,' ') as words from
load_tweets;
```



```
hive> create table split_words as select id as id,split(text,' ') as words from load_tweets;
Query ID = kiran_20160125191358_c86dd1a3-e5b5-46e1-82cc-2ef286b4dd23
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1453705089859_0014, Tracking URL = http://ACD-KIRAN:8088/proxy/application_1453705089859_0014/
Kill Command = /home/kiran/hadoop-2.7.1/bin/hadoop job  -kill job_1453705089859_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-01-25 19:14:03,596 Stage-1 map = 0%,  reduce = 0%
2016-01-25 19:14:08,788 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.8 sec
MapReduce Total cumulative CPU time: 2 seconds 800 msec
Ended Job = job_1453705089859_0014
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/user/hive/warehouse/.hive-staging_hive_2016-01-25_19-13-58_846_4051616223374719751-1/-ext-10001
Moving data to: hdfs://localhost:9000/user/hive/warehouse/split_words
Table default.split_words stats: [numFiles=1, numRows=138, totalSize=19386, rawDataSize=19248]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.8 sec   HDFS Read: 488677 HDFS Write: 19465 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 800 msec
OK
Time taken: 12.234 seconds
```

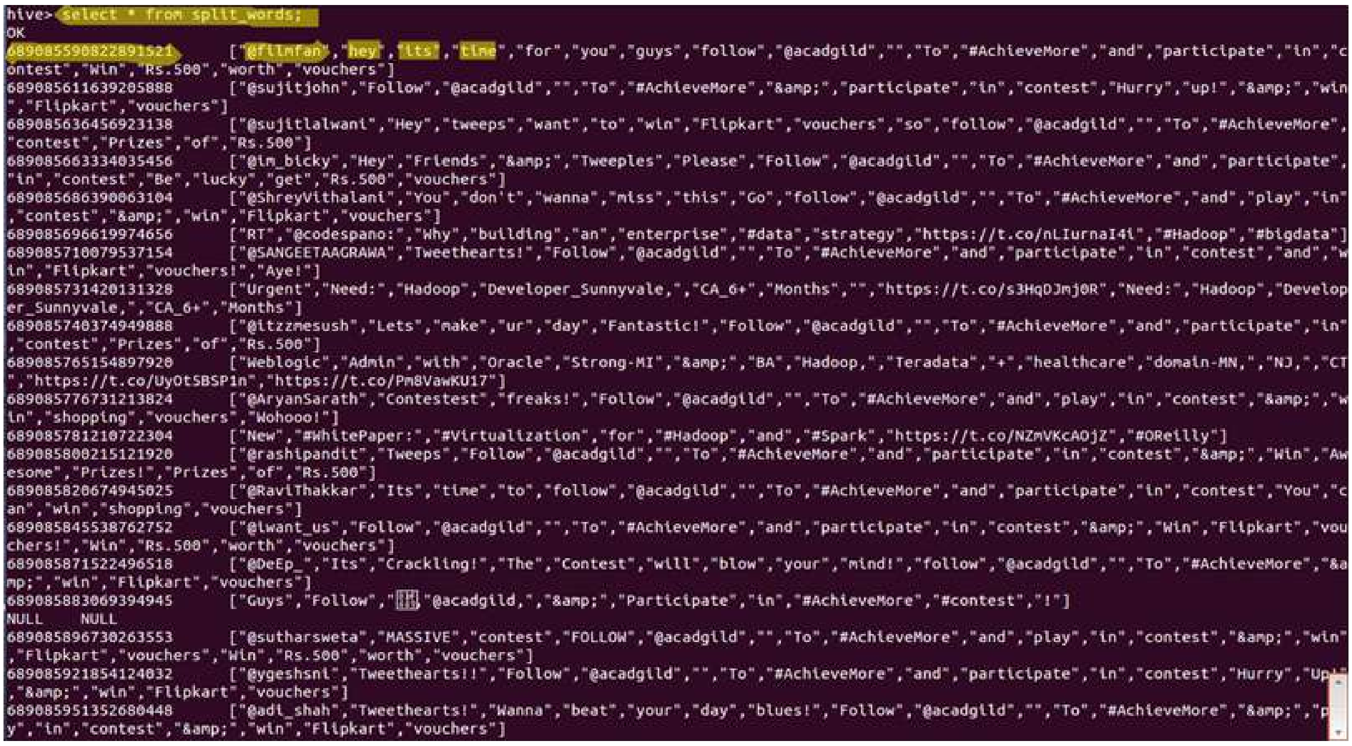
- The schema of the table can be viewed using the **describe** command.



```
hive> describe split_words;
OK
id                      bigint
words                   array<string>
Time taken: 0.051 seconds, Fetched: 2 row(s)
hive>
```

- The contents of the table can be viewed using the following command:

```
select * from split_words;
```



```
hive> select * from split_words;
OK
689085590822891521      ["@filmfan","hey","its","time","for","you","guys","follow","@acadgild","","To","#AchieveMore","and","participate","in","c
ontest","Win","Rs.500","worth","vouchers"]
689085611639205888      ["@sujitjohn","Follow","@acadgild","","To","#AchieveMore","&amp;","participate","in","contest","Hurry","up!","&amp;","win
","Flipkart","vouchers"]
689085636456923138      ["@sujitlalwani","Hey","tweeps","want","to","win","Flipkart","vouchers","so","follow","@acadgild","","To","#AchieveMore",
"contest","Prizes","of","Rs.500"]
689085663334035456      ["@im_bicky","Hey","Friends","&amp;","Tweeples","Please","Follow","@acadgild","","To","#AchieveMore","and","participate",
"in","contest","Be","lucky","get","Rs.500","vouchers"]
689085686390063104      ["@ShreyVithalani","You","don't","wanna","miss","this","Go","follow","@acadgild","","To","#AchieveMore","and","play","in"
,"contest","&amp;","win","Flipkart","vouchers"]
689085696619974656      ["RT","@codespano:","Why","building","an","enterprise","#data","strategy","https://t.co/nLIurnaI4i","#Hadoop","#bigdata"]
689085710079537154      ["@SANGEETAAGRAWA","Tweethearts!","Follow","@acadgild","","To","#AchieveMore","and","participate","in","contest","and","w
in","Flipkart","vouchers!","Aye!"]
689085731420131328      ["Urgent","Need:","Hadoop","Developer_Sunnyvale,","CA_6+","Months","","https://t.co/s3HqDJmj0R","Need:","Hadoop","Develop
er_Sunnyvale,","CA_6+","Months"]
689085740374949888      ["@itzzmesush","Lets","make","ur","day","Fantastic!","Follow","@acadgild","","To","#AchieveMore","and","participate","in"
,"contest","Prizes","of","Rs.500"]
689085765154897920      ["Weblogic","Admin","with","Oracle","Strong-MI","&amp;","BA","Hadoop,","Teradata","+","healthcare","domain-MN,","NJ,","CT
","https://t.co/UyOtSBSP1n","https://t.co/Pm8VawKU17"]
689085776731213824      ["@AryanSarath","Contestest","freaks!","Follow","@acadgild","","To","#AchieveMore","and","play","in","contest","&amp;","w
in","shopping","vouchers","Wohooo!"]
689085781210722304      ["New","#WhitePaper:","#Virtualization","for","#Hadoop","and","#Spark","https://t.co/NZmVKcAOjZ","#OReilly"]
689085800215121920      ["@rashipandit","Tweeps","Follow","@acadgild","","To","#AchieveMore","and","participate","in","contest","&amp;","Win","Aw
esome","Prizes!","Prizes","of","Rs.500"]
689085820674945025      ["@RaviThakkar","Its","time","to","follow","@acadgild","","To","#AchieveMore","and","participate","in","contest","You","c
an","win","shopping","vouchers"]
689085845538762752      ["@iwant_us","Follow","@acadgild","","To","#AchieveMore","and","participate","in","contest","&amp;","Win","Flipkart","vou
chers!","Win","Rs.500","worth","vouchers"]
689085871522496518      ["@DeEp_","Its","Crackling!","The","Contest","will","blow","your","mind!","follow","@acadgild","","To","#AchieveMore","&a
mp;","win","Flipkart","vouchers"]
689085883069394945      ["Guys","Follow","▯"@acadgild,","&amp;","Participate","in","#AchieveMore","#contest","!"]
NULL    NULL
689085896730263553      ["@sutharsweta","MASSIVE","contest","FOLLOW","@acadgild","","To","#AchieveMore","and","play","in","contest","&amp;","win"
,"Flipkart","vouchers","Win","Rs.500","worth","vouchers"]
689085921854124032      ["@ygeshsni","Tweethearts!!","Follow","@acadgild","","To","#AchieveMore","and","participate","in","contest","Hurry","Up!"
,"&amp;","win","Flipkart","vouchers"]
689085951352680448      ["@adi_shah","Tweethearts!","Wanna","beat","your","day","blues!","Follow","@acadgild","","To","#AchieveMore","&amp;","p
y","in","contest","&amp;","win","Flipkart","vouchers"]
```

- Next, every word inside the array has been split as a new row.
  For this UDTF (User Defined Table Generating Function) is used.
    - A built-in UDTF called explode will help extract each element from an array and create a new row for that element.

- A table has been created that can store id and word.

```
create table tweet_word as select id as id,word from split_words LATERAL
VIEW explode(words) w as word;
```

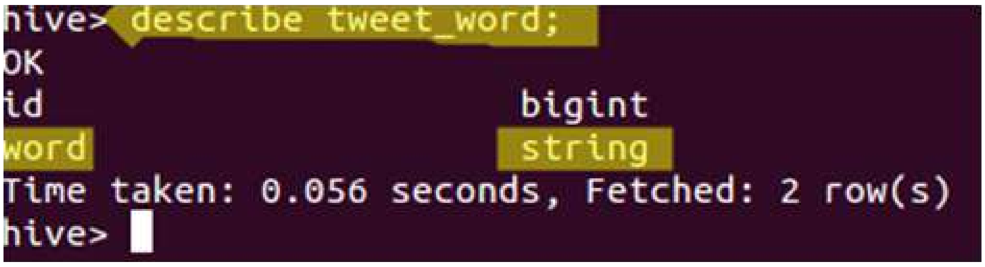
***Note**:Syntax for LATERAL VIEW explode UDTF is as follows:

```
lateralView: LATERAL VIEW udtf(expression) tableAlias AS columnAlias (','
columnAlias)*fromClause: FROM baseTable (lateralView)
```

In general, **explode** UDTF has some limitations; **explode** cannot be used with other columns in the same select relation. Hence, LATERAL VIEW is added in conjunction with explode, so that the explode function can be used in other columns as well.

The schema of the table can be seen using the 'describe' command.

In **Exhibit 26**, it is seen that the array of values has been converted into a string.



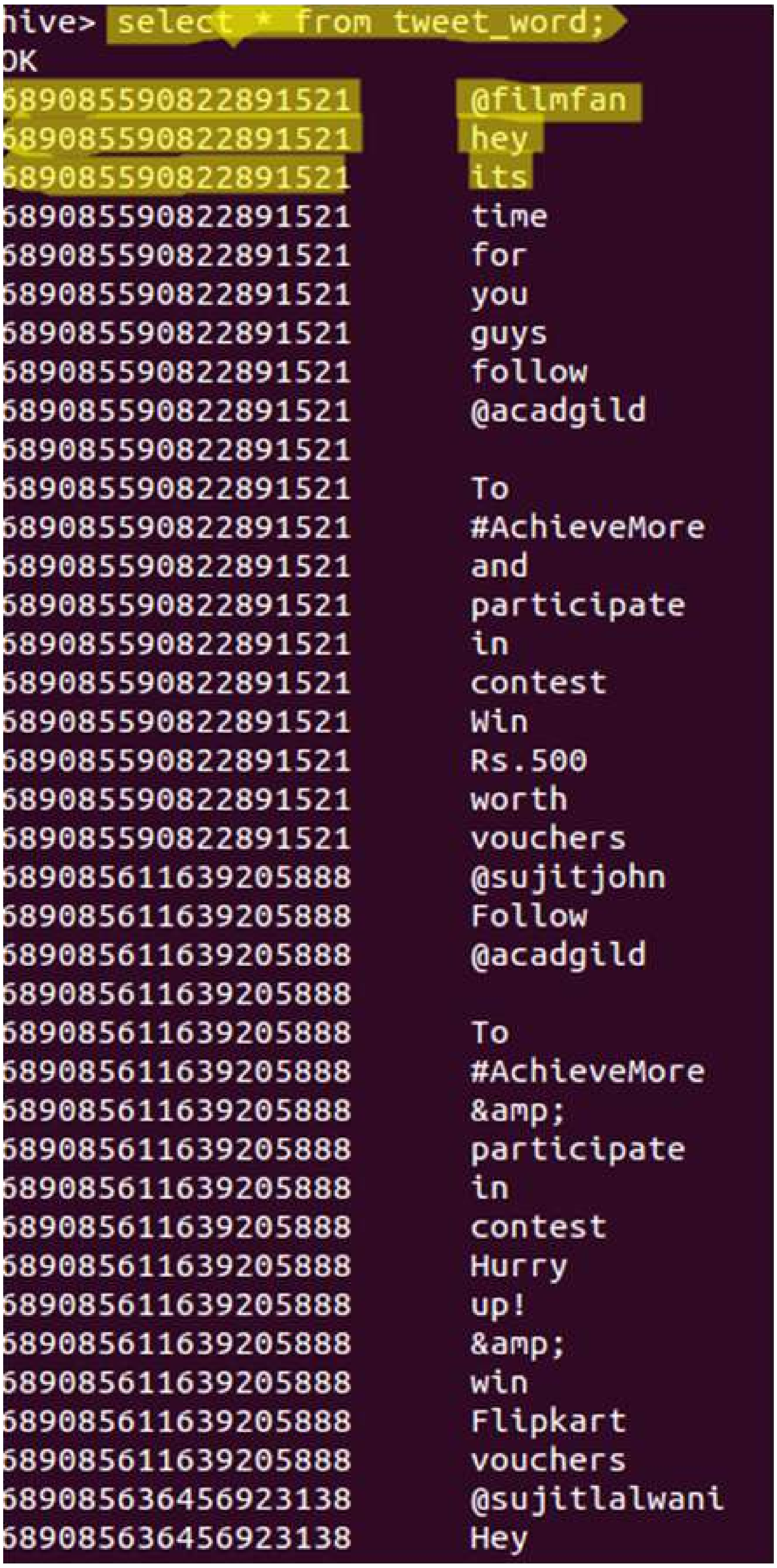
```
hive> describe tweet_word;
OK
id                              bigint
word                            string
Time taken: 0.056 seconds, Fetched: 2 row(s)
hive>
```

Exhibit 26: Conversion of an array of values into a string

- Contents of the table can be seen using the following command:

```
select * from tweet_word;
```

In **Exhibit 27**, it is seen that the array of words has been split as single words in a new row.



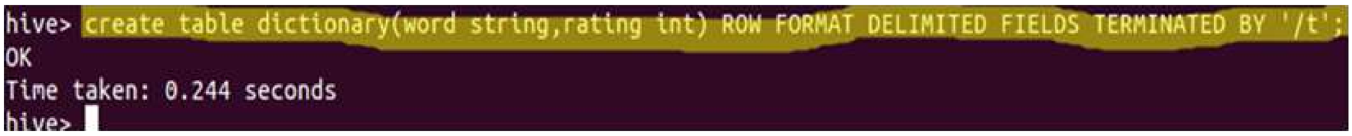
```
hive> select * from tweet_word;
OK
689085590822891521        @filmfan
689085590822891521        hey
689085590822891521        its
689085590822891521        time
689085590822891521        for
689085590822891521        you
689085590822891521        guys
689085590822891521        follow
689085590822891521        @acadgild
689085590822891521
689085590822891521        To
689085590822891521        #AchieveMore
689085590822891521        and
689085590822891521        participate
689085590822891521        in
689085590822891521        contest
689085590822891521        Win
689085590822891521        Rs.500
689085590822891521        worth
689085590822891521        vouchers
689085611639205888        @sujitjohn
689085611639205888        Follow
689085611639205888        @acadgild
689085611639205888
689085611639205888        To
689085611639205888        #AchieveMore
689085611639205888        &amp;
689085611639205888        participate
689085611639205888        in
689085611639205888        contest
689085611639205888        Hurry
689085611639205888        up!
689085611639205888        &amp;
689085611639205888        win
689085611639205888        Flipkart
689085611639205888        vouchers
689085636456923138        @sujitlalwani
689085636456923138        Hey
```

Exhibit 27: Splitting of array of wordsas single words in a new row

- The AFINN dictionary will be used to calculate the sentiments.
  - AFINN is a dictionary that consists of 2500 words rated from +5 to -5 depending on their meaning.
  - A table will be created to load the contents of an AFINN dictionary.

Download AFINN dictionary from here.

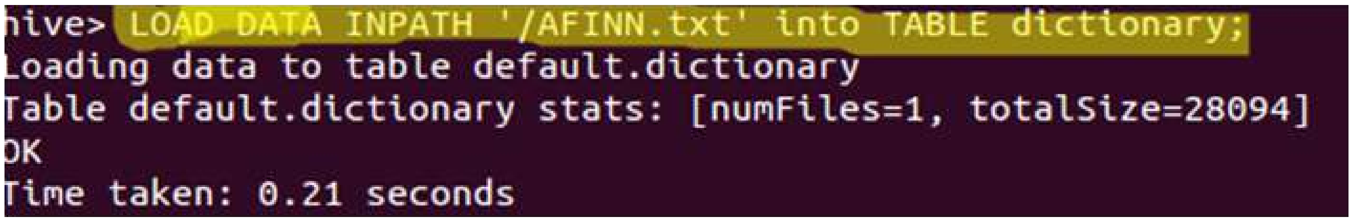
```
create table dictionary(word string,ratingint) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t';
```



```
hive> create table dictionary(word string,rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY '/t';
OK
Time taken: 0.244 seconds
hive>
```

- The AFINN dictionary will be loaded into the table by using the following command:

```
LOAD DATA INPATH '/AFINN.txt' into TABLE dictionary;
```

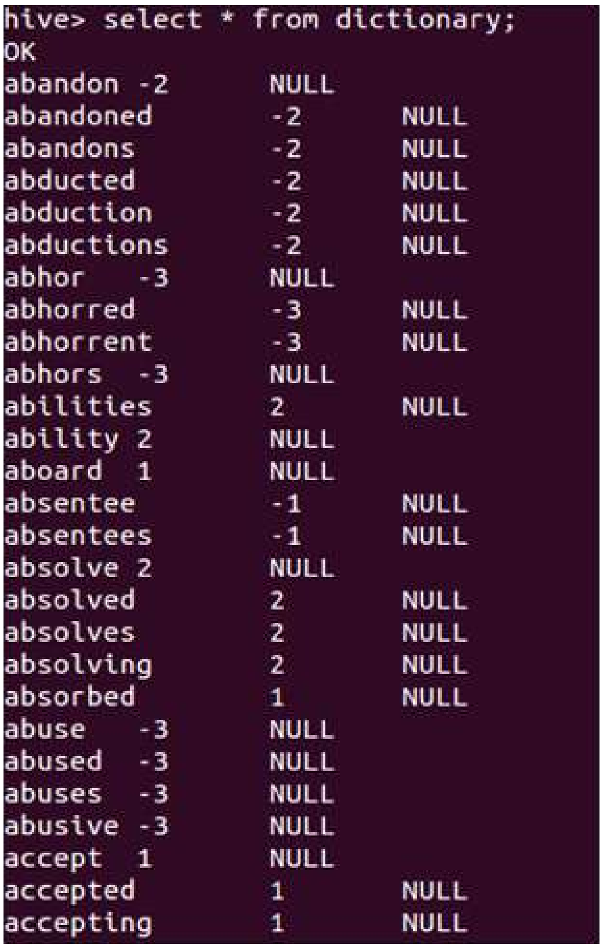


```
hive> LOAD DATA INPATH '/AFINN.txt' into TABLE dictionary;
Loading data to table default.dictionary
Table default.dictionary stats: [numFiles=1, totalSize=28094]
OK
Time taken: 0.21 seconds
```

We now have loaded the AFINN dictionary in the root directory of HDFS.

- The contents of the dictionary table can be used by using this command:

```
select * from dictionary;
```

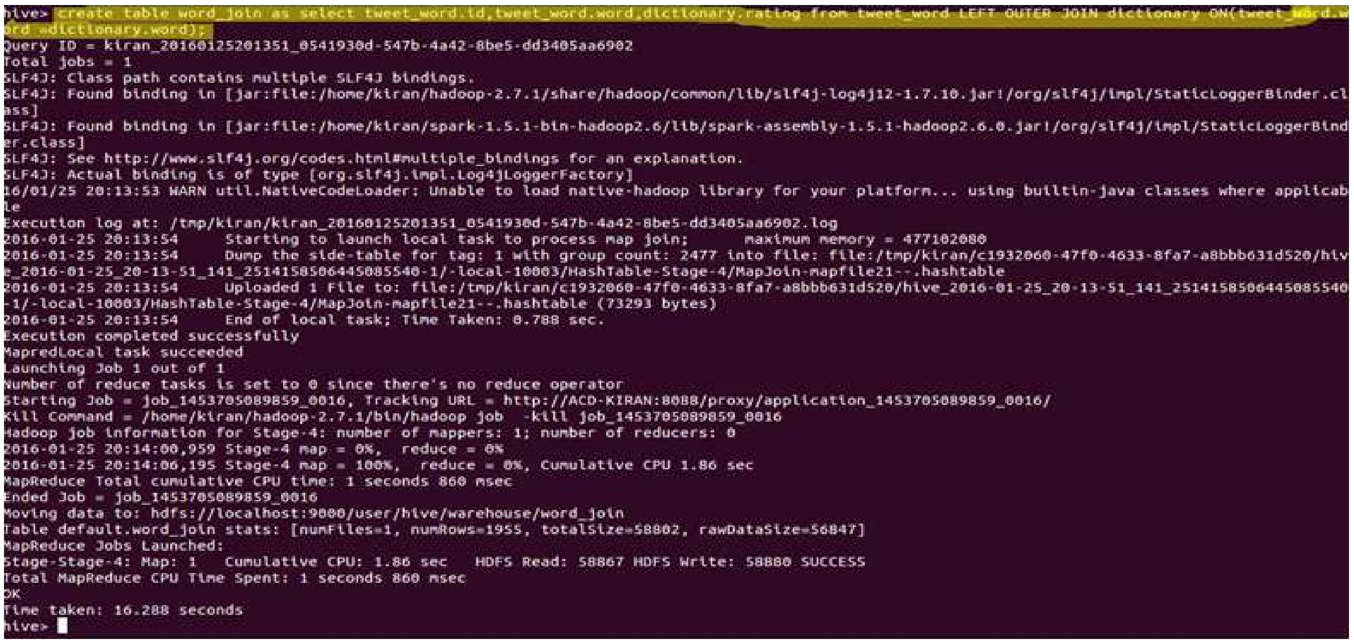


```
hive> select * from dictionary;
OK
abandon -2      NULL
abandoned       -2      NULL
abandons        -2      NULL
abducted        -2      NULL
abduction       -2      NULL
abductions      -2      NULL
abhor   -3      NULL
abhorred        -3      NULL
abhorrent       -3      NULL
abhors  -3      NULL
abilities       2       NULL
ability 2       NULL
aboard  1       NULL
absentee        -1      NULL
absentees       -1      NULL
absolve 2       NULL
absolved        2       NULL
absolves        2       NULL
absolving       2       NULL
absorbed        1       NULL
abuse   -3      NULL
abused  -3      NULL
abuses  -3      NULL
abusive -3      NULL
accept  1       NULL
accepted        1       NULL
accepting       1       NULL
```

- Now, the **tweet_word table** andthe**dictionary table**will be joined so that the rating of the words will be joined to the word by using the following command:

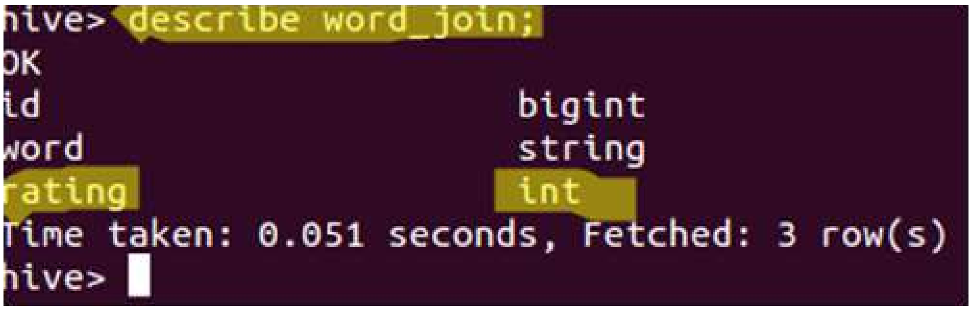
```
create table word_join as select
tweet_word.id,tweet_word.word,dictionary.rating from tweet_word LEFT OUTER
JOIN dictionary ON(tweet_word.word =dictionary.word);
```

Exhibit 28 shows the schema of the table by describing it, and in Exhibit 29, it is seen that the rating column has been added along with the id and the word.



```
hive> create table word_join as select tweet_word.id,tweet_word.word,dictionary.rating from tweet_word LEFT OUTER JOIN dictionary ON(tweet_word.w
ord =dictionary.word);
Query ID = kiran_20160125201351_0541930d-547b-4a42-8be5-dd3405aa6902
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/kiran/hadoop-2.7.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: Found binding in [jar:file:/home/kiran/spark-1.5.1-bin-hadoop2.6/lib/spark-assembly-1.5.1-hadoop2.6.0.jar!/org/slf4j/impl/StaticLoggerBind
er.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
16/01/25 20:13:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicab
le
Execution log at: /tmp/kiran/kiran_20160125201351_0541930d-547b-4a42-8be5-dd3405aa6902.log
2016-01-25 20:13:54     Starting to launch local task to process map join;      maximum memory = 477102080
2016-01-25 20:13:54     Dump the side-table for tag: 1 with group count: 2477 into file: file:/tmp/kiran/c1932060-47f0-4633-8fa7-a8bbb631d520/hiv
e_2016-01-25_20-13-51_141_2514158506445085540-1/-local-10003/HashTable-Stage-4/MapJoin-mapfile21--.hashtable
2016-01-25 20:13:54     Uploaded 1 File to: file:/tmp/kiran/c1932060-47f0-4633-8fa7-a8bbb631d520/hive_2016-01-25_20-13-51_141_2514158506445085540
-1/-local-10003/HashTable-Stage-4/MapJoin-mapfile21--.hashtable (73293 bytes)
2016-01-25 20:13:54     End of local task; Time Taken: 0.788 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1453705089859_0016, Tracking URL = http://ACD-KIRAN:8088/proxy/application_1453705089859_0016/
Kill Command = /home/kiran/hadoop-2.7.1/bin/hadoop job  -kill job_1453705089859_0016
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 0
2016-01-25 20:14:00,959 Stage-4 map = 0%,  reduce = 0%
2016-01-25 20:14:06,195 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 1.86 sec
MapReduce Total cumulative CPU time: 1 seconds 860 msec
Ended Job = job_1453705089859_0016
Moving data to: hdfs://localhost:9000/user/hive/warehouse/word_join
Table default.word_join stats: [numFiles=1, numRows=1955, totalSize=58802, rawDataSize=56847]
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1   Cumulative CPU: 1.86 sec   HDFS Read: 58867 HDFS Write: 58880 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 860 msec
OK
Time taken: 16.288 seconds
hive>
```

Exhibit 28: Schema of the table

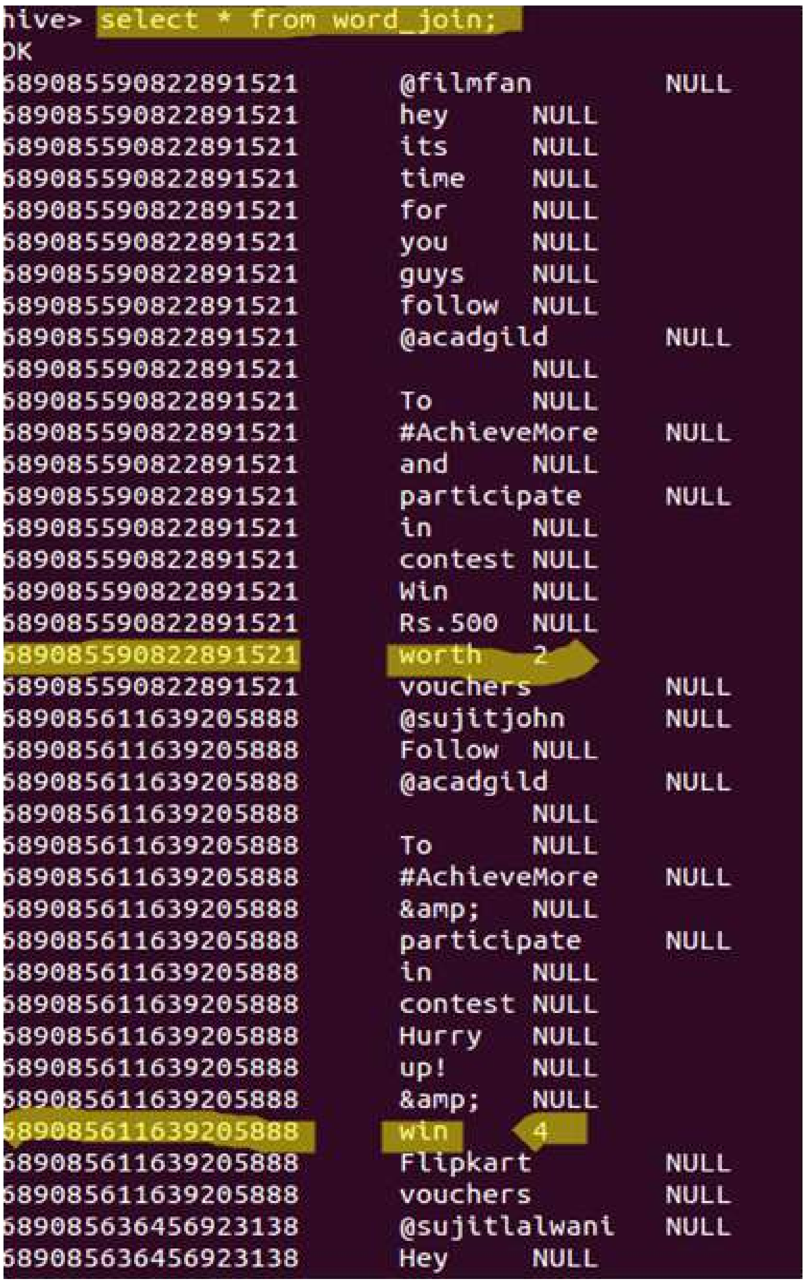


```
hive> describe word_join;
OK
id                              bigint
word                            string
rating                          int
Time taken: 0.051 seconds, Fetched: 3 row(s)
hive>
```

Exhibit 29: Rating column

Whenever there is a match with the word of the tweet in the dictionary, the rating will be given to that word else NULL will be present.

- The contents of the table can be viewed using the following command:



```
hive> select * from word_join;
OK
689085590822891521      @filmfan        NULL
689085590822891521      hey     NULL
689085590822891521      its     NULL
689085590822891521      time    NULL
689085590822891521      for     NULL
689085590822891521      you     NULL
689085590822891521      guys    NULL
689085590822891521      follow  NULL
689085590822891521      @acadgild       NULL
689085590822891521              NULL
689085590822891521      To      NULL
689085590822891521      #AchieveMore    NULL
689085590822891521      and     NULL
689085590822891521      participate     NULL
689085590822891521      in      NULL
689085590822891521      contest NULL
689085590822891521      Win     NULL
689085590822891521      Rs.500  NULL
689085590822891521      worth   2
689085590822891521      vouchers        NULL
689085611639205888      @sujitjohn      NULL
689085611639205888      Follow  NULL
689085611639205888      @acadgild       NULL
689085611639205888              NULL
689085611639205888      To      NULL
689085611639205888      #AchieveMore    NULL
689085611639205888      &amp;   NULL
689085611639205888      participate     NULL
689085611639205888      in      NULL
689085611639205888      contest NULL
689085611639205888      Hurry   NULL
689085611639205888      up!     NULL
689085611639205888      &amp;   NULL
689085611639205888      win     4
689085611639205888      Flipkart        NULL
689085611639205888      vouchers        NULL
689085636456923138      @sujitlalwani   NULL
689085636456923138      Hey     NULL
```

```
select * from word_join;
```

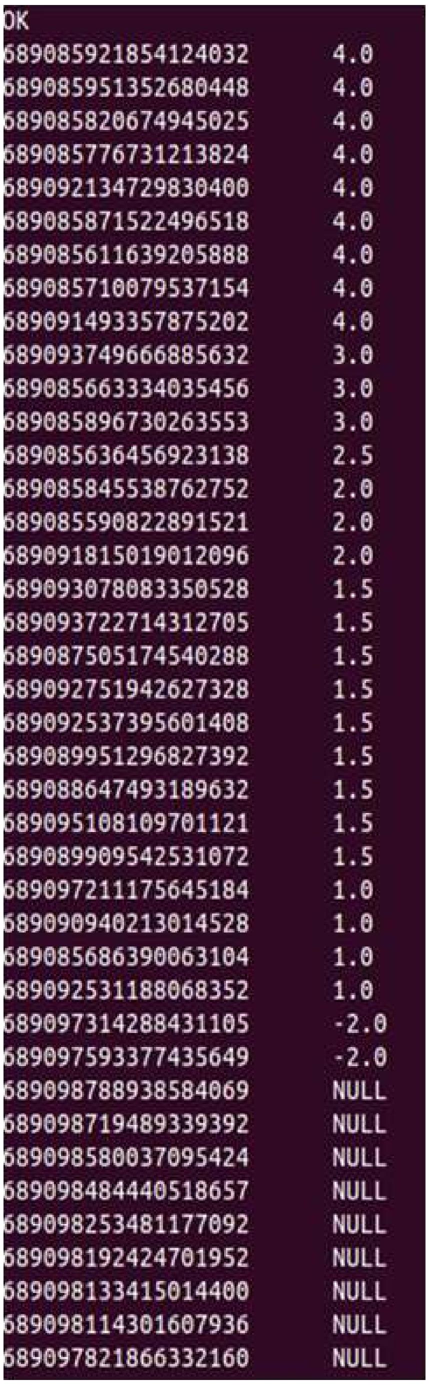
- The groupby operation needs to performed on the tweet_id so that all the words of one tweet will come to a single place. This should be followed by performing the average operation on the rating of the words of each tweet so that the average rating of each tweet can be found.

B) SAMPLE OUTPUT

```
select id,AVG(rating) as rating from word_join GROUP BY word_join.id order
by rating DESC;
```

By executing the above mentioned command, the average rating of each tweet is calculated by using each word of the tweet and arranging the tweets in the descending order as per their rating.

In **Exhibit 29**, the tweet_id and its rating has been displayed.



```
OK
689085921854124032      4.0
689085951352680448      4.0
689085820674945025      4.0
689085776731213824      4.0
689092134729830400      4.0
689085871522496518      4.0
689085611639205888      4.0
689085710079537154      4.0
689091493357875202      4.0
689093749666885632      3.0
689085663334035456      3.0
689085896730263553      3.0
689085636456923138      2.5
689085845538762752      2.0
689085590822891521      2.0
689091815019012096      2.0
689093078083350528      1.5
689093722714312705      1.5
689087505174540288      1.5
689092751942627328      1.5
689092537395601408      1.5
689089951296827392      1.5
689088647493189632      1.5
689095108109701121      1.5
689089909542531072      1.5
689097211175645184      1.0
689090940213014528      1.0
689085686390063104      1.0
689092531188068352      1.0
689097314288431105      -2.0
689097593377435649      -2.0
689098788938584069      NULL
689098719489339392      NULL
689098580037095424      NULL
689098484440518657      NULL
689098253481177092      NULL
689098192424701952      NULL
689098133415014400      NULL
689098114301607936      NULL
689097821866332160      NULL
```

Chapter - 10

HIVE USE CASE COUNTING HASHTAGS

01 Problem Statement 1:

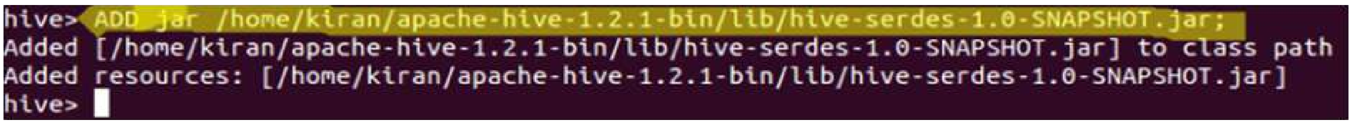
Count popular hashtags in tweets from Twitter

A) SOLUTION

In this part of the e-book, complex data types in Hive, like Struct and Arrays, will be discussed. Data from Twitter will be taken as an example to explain the complex types.

- As the tweets coming in from Twitter are in JSON format, the tweets need to be loaded into the Hive using the JSON input format. Cloudera Hive JSONSerDe have been used for this purpose.
  - Cloudera JSON SerDe can be downloaded using ***this link.***
- After downloading Cloudera's JSONSerDe, the JAR file needs to be copied into the lib directory of the already installed Hive folder. Next, the JAR file needs to be added into the Hive as shown below

```
ADD JAR 'path of the jar file';
```



```
hive> ADD jar /home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/kiran/apache-hive-1.2.1-bin/lib/hive-serdes-1.0-SNAPSHOT.jar]
hive>
```

- After successfully adding the JAR file, a Hive table needs to be created to store the Twitter data.

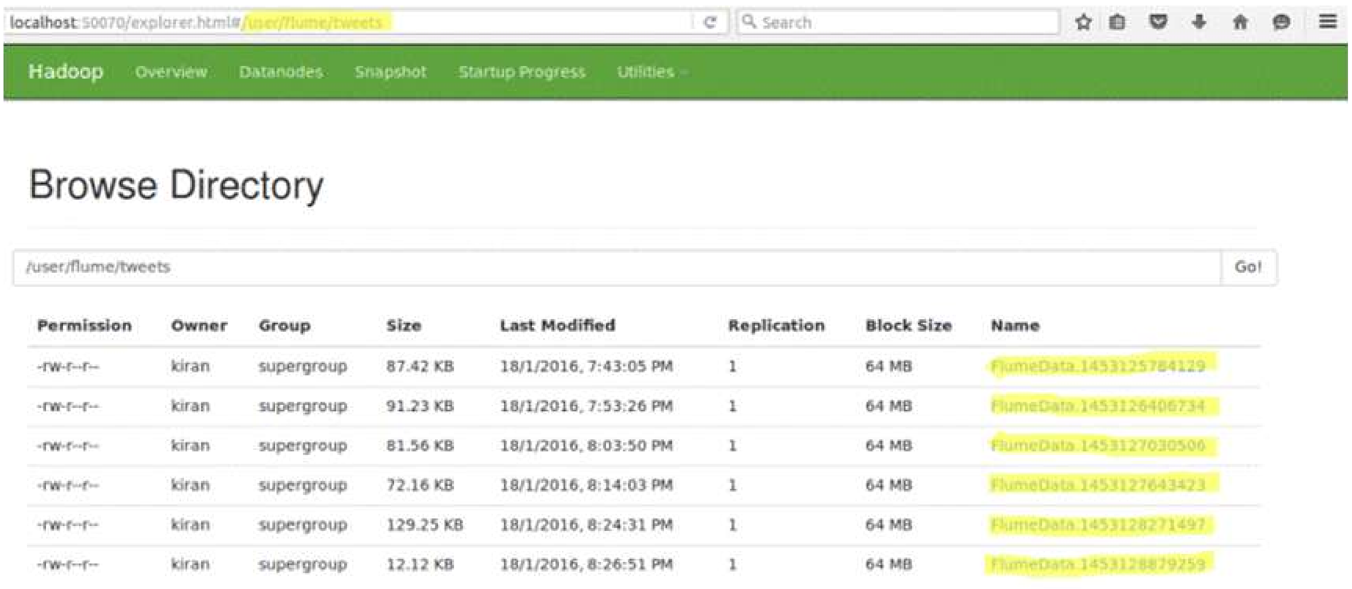
For calculating the hashtags, tweet_id and hashtag_text are needed, hence, a Hive table will be created that will help extract the id and hashtag_text from the tweets using the Cloudera JSONSerDe.

Below mentioned is the one of the tweets that has been collected.

```
{"filter_level":"low","retweeted":false,"in_reply_to_screen_name":"FilmFan"
,"truncated":false,"lang":"en","in_reply_to_status_id_str":null,"id":689085
590822891521,"in_reply_to_user_id_str":"6048122","timestamp_ms":"1453125782
100","in_reply_to_status_id":null,"created_at":"Mon Jan 18 14:03:02 +0000
2016","favorite_count":0,"place":null,"coordinates":null,"text":"@filmfan
hey its time for you guys follow @acadgild To #AchieveMore and participate
in contest Win Rs.500 worth
vouchers","contributors":null,"geo":null,"entities":{"symbols":[],"urls":[]
,"hashtags":[{"text":"AchieveMore","indices":[56,68]}],"user_mentions":[{"i
d":6048122,"name":"Tanya","indices":[0,8],"screen_name":"FilmFan","id_str":
"6048122"},{"id":2649945906,"name":"ACADGILD","indices":[42,51],"screen_nam
e":"acadgild","id_str":"2649945906"}]},"is_quote_status":false,"source":"<a
href=\"https://about.twitter.com/products/tweetdeck\"
rel=\"nofollow\">TweetDeck<\/a>","favorited":false,"in_reply_to_user_id":60
48122,"retweet_count":0,"id_str":"689085590822891521","user":{"location":"I
ndia
","default_profile":false,"profile_background_tile":false,"statuses_count":
86548,"lang":"en","profile_link_color":"94D487","profile_banner_url":"https
://pbs.twimg.com/profile_banners/197865769/1436198000","id":197865769,"foll
owing":null,"protected":false,"favourites_count":1002,"profile_text_color":
"000000","verified":false,"description":"Proud Indian, Digital Marketing
Consultant,Traveler, Foodie, Adventurer, Data Architect, Movie Lover, Namo
Fan","contributors_enabled":false,"profile_sidebar_border_color":"000000","
name":"Bahubali","profile_background_color":"000000","created_at":"Sat Oct
02 17:41:02 +0000
2010","default_profile_screenshot":false,"followers_count":4467,"profile_sc
reenshot_url_https":"https://pbs.twimg.com/profile_screenshots/664486535040
000000/GOjDUiuK_normal.jpg","geo_enabled":true,"profile_background_screensh
ot_url":"http://abs.twimg.com/screenshots/themes/theme1/bg.png","profile_ba
ckground_screenshot_url_https":"https://abs.twimg.com/screenshots/themes/th
eme1/bg.png","follow_request_sent":null,"url":null,"utc_offset":19800,"time
_zone":"Chennai","notifications":null,"profile_use_background_screenshot":f
alse,"friends_count":810,"profile_sidebar_fill_color":"000000","screen_name
":"Ashok_Uppuluri","id_str":"197865769","profile_screenshot_url":"http://pb
s.twimg.com/profile_screenshots/664486535040000000/GOjDUiuK_normal.jpg","li
sted_count":50,"is_translator":false}}
```

The tweet is in nested the JSON format. From this tweet, we will extract the id, which is the tweet_id and hashtag text, which is the hashtag.

The tweets are stored in the **'/user/flume/tweets/'** directory of HDFS as shown below.



Browse Directory

/user/flume/tweets Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	kiran	supergroup	87.42 KB	18/1/2016, 7:43:05 PM	1	64 MB	FlumeData.1453125784129
-rw-r--r--	kiran	supergroup	91.23 KB	18/1/2016, 7:53:26 PM	1	64 MB	FlumeData.1453126406734
-rw-r--r--	kiran	supergroup	81.56 KB	18/1/2016, 8:03:50 PM	1	64 MB	FlumeData.1453127030506
-rw-r--r--	kiran	supergroup	72.16 KB	18/1/2016, 8:14:03 PM	1	64 MB	FlumeData.1453127643423
-rw-r--r--	kiran	supergroup	129.25 KB	18/1/2016, 8:24:31 PM	1	64 MB	FlumeData.1453128271497
-rw-r--r--	kiran	supergroup	12.12 KB	18/1/2016, 8:26:51 PM	1	64 MB	FlumeData.1453128879259

- Next, an external table will be created in Hive in the same directory where our tweets are present i.e., "/user/flume/tweets/," so that the tweets present in this location will be automatically stored in the Hive table.

The command for creating a Hive table to store id and hashtag text of the tweets are as follows:

```
CREATE EXTERNAL TABLE tweets (id BIGINT,entities
STRUCT<hashtags:ARRAY<STRUCT<text:STRING>>>) ROW FORMAT SERDE
'com.cloudera.hive.serde.JSONSerDe' LOCATION '/user/flume/tweets';
```

- **Structure:**It is a collection of elements of different types.
- **Arrays:** It is an ordered collection of elements. The elements in an array must be of the same type.

- A tweet has been examined for greater clarity.

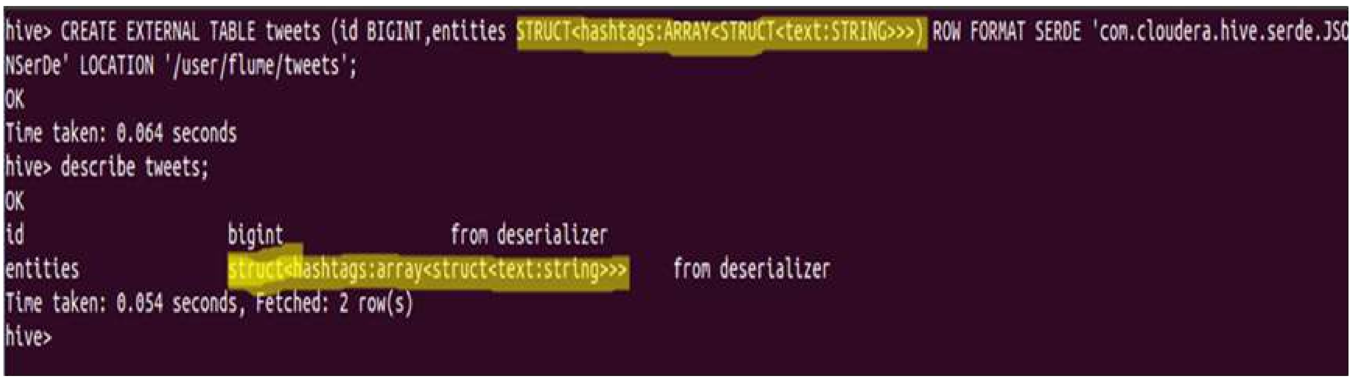
```
"entities":{"symbols":[],"urls":[],"hashtags":[{"text":"AchieveMore","indic
es":[56,68]}]
```

Here, "entities" is a structure in which hashtags are present in a nested structure.

Structures in Hive are enclosed within open and closed curly braces { }, and Arrays are enclosed within open and closed square brackets [ ].

- Schema of a table can be checked using the command,**describe tweets;**
- Contents of the table can be viewed using the command **select * from tweets;**

Exhibit 30 shows that the tweet_id and hashtags of the tweet examined have been loaded successfully.



```
hive> CREATE EXTERNAL TABLE tweets (id BIGINT,entities STRUCT<hashtags:ARRAY<STRUCT<text:STRING>>>) ROW FORMAT SERDE 'com.cloudera.hive.serde.JSO
NSerDe' LOCATION '/user/flume/tweets';
OK
Time taken: 0.064 seconds
hive> describe tweets;
OK
id                  bigint                  from deserializer
entities            struct<hashtags:array<struct<text:string>>>     from deserializer
Time taken: 0.054 seconds, Fetched: 2 row(s)
hive>
```

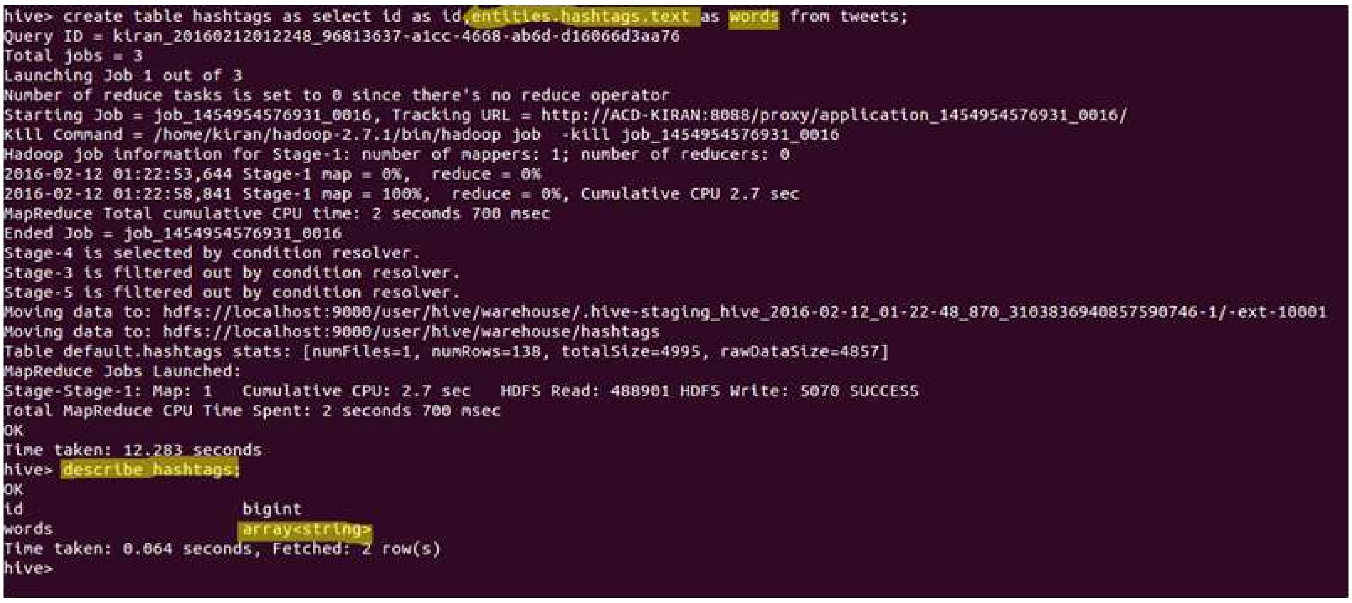
Exhibit 30: tweet_id and hashtags of the tweet have been loaded successfully

- Next, only the "hashtags" array need to be extracted from this structure that consists of the text.
  We can achieve this by using the following command:

```
create table hashtags as select id as id,entities.hashtags.text as words
from tweets;
```

- A table is created with the name “hashtags” and in that table the tweet_id and hashtags text array needs to be sorted.

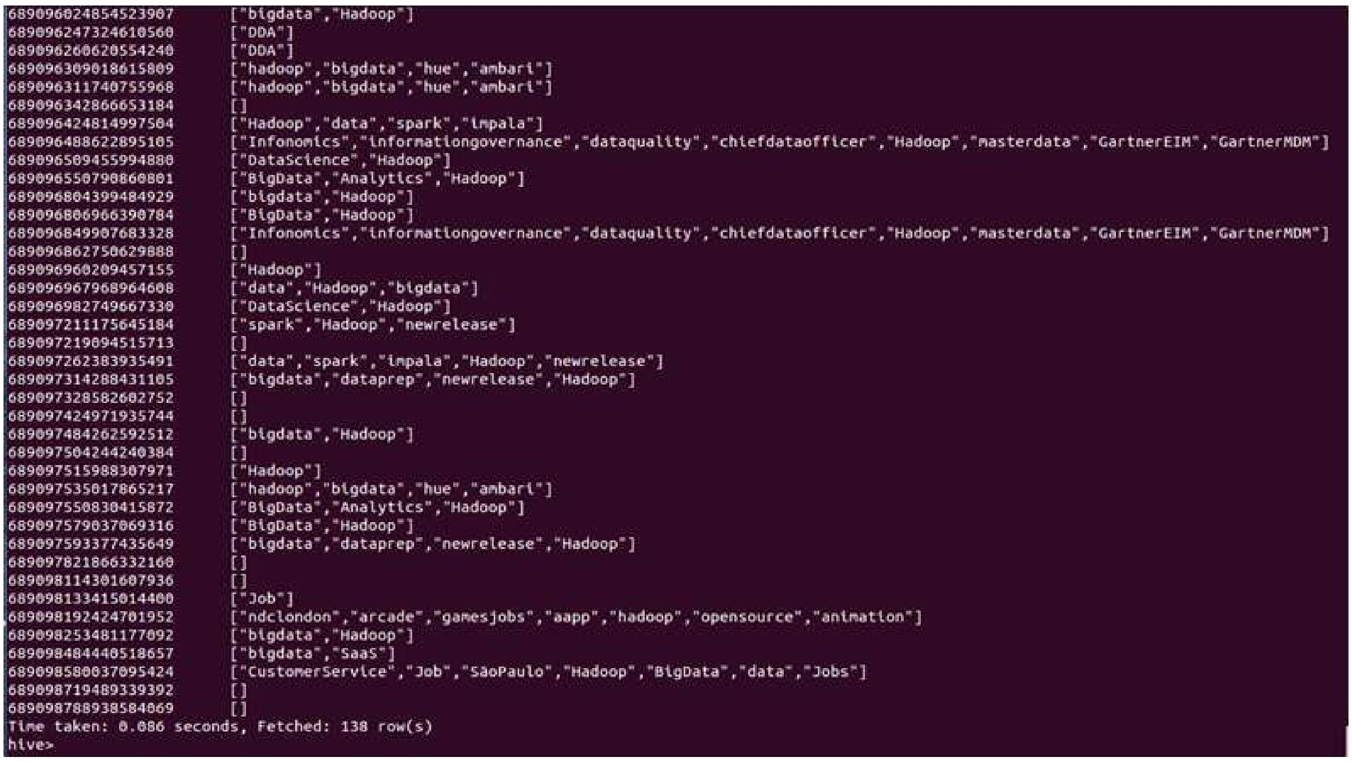
Exhibit 31 shows asuccessful extraction of the hashtags text.



```
hive> create table hashtags as select id as id,entities.hashtags.text as words from tweets;
Query ID = kiran_20160212012248_96813637-a1cc-4668-ab6d-d16066d3aa76
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454954576931_0016, Tracking URL = http://ACD-KIRAN:8088/proxy/application_1454954576931_0016/
Kill Command = /home/kiran/hadoop-2.7.1/bin/hadoop job  -kill job_1454954576931_0016
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-02-12 01:22:53,644 Stage-1 map = 0%,  reduce = 0%
2016-02-12 01:22:58,841 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.7 sec
MapReduce Total cumulative CPU time: 2 seconds 700 msec
Ended Job = job_1454954576931_0016
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/user/hive/warehouse/.hive-staging_hive_2016-02-12_01-22-48_870_3103836940857590746-1/-ext-10001
Moving data to: hdfs://localhost:9000/user/hive/warehouse/hashtags
Table default.hashtags stats: [numFiles=1, numRows=138, totalSize=4995, rawDataSize=4857]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.7 sec   HDFS Read: 488901 HDFS Write: 5070 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 700 msec
OK
Time taken: 12.283 seconds
hive> describe hashtags;
OK
id                      bigint
words                   array<string>
Time taken: 0.064 seconds, Fetched: 2 row(s)
hive>
```

Exhibit 31: Extracted hashtags text

- The contents can be viewed using the following command:



```
689096024854523907      ["bigdata","Hadoop"]
689096247324610560      ["DDA"]
689096260620554240      ["DDA"]
689096309018615809      ["hadoop","bigdata","hue","ambari"]
689096311740755968      ["hadoop","bigdata","hue","ambari"]
689096342866653184      []
689096424814997504      ["Hadoop","data","spark","impala"]
689096488622895105      ["Infonomics","informationgovernance","dataquality","chiefdataofficer","Hadoop","masterdata","GartnerEIM","GartnerMDM"]
689096509455994880      ["DataScience","Hadoop"]
689096550790860801      ["BigData","Analytics","Hadoop"]
689096804399484929      ["bigdata","Hadoop"]
689096806966390784      ["BigData","Hadoop"]
689096849907683328      ["Infonomics","informationgovernance","dataquality","chiefdataofficer","Hadoop","masterdata","GartnerEIM","GartnerMDM"]
689096862750629888      []
689096960209457155      ["Hadoop"]
689096967968964608      ["data","Hadoop","bigdata"]
689096982749667330      ["DataScience","Hadoop"]
689097211175645184      ["spark","Hadoop","newrelease"]
689097219094515713      []
689097262383935491      ["data","spark","impala","Hadoop","newrelease"]
689097314288431105      ["bigdata","dataprep","newrelease","Hadoop"]
689097328582602752      []
689097424971935744      []
689097484262592512      ["bigdata","Hadoop"]
689097504244240384      []
689097515988307971      ["Hadoop"]
689097535017865217      ["hadoop","bigdata","hue","ambari"]
689097550830415872      ["BigData","Analytics","Hadoop"]
689097579037069316      ["BigData","Hadoop"]
689097593377435649      ["bigdata","dataprep","newrelease","Hadoop"]
689097821866332160      []
689098114301607936      []
689098133415014400      ["Job"]
689098192424701952      ["ndclondon","arcade","gamesjobs","aapp","hadoop","opensource","animation"]
689098253481177092      ["bigdata","Hadoop"]
689098484440518657      ["bigdata","SaaS"]
689098580037095424      ["CustomerService","Job","SaoPaulo","Hadoop","BigData","data","Jobs"]
689098719489339392      []
689098788938584069      []
Time taken: 0.086 seconds, Fetched: 138 row(s)
hive>
```

```sql
select * from hashtags;
```

Here, it is seen that there are two or more hashtags for every tweet, hence, every hashtag needs to be extracted into a new row. For this, a UDTF needs to be used, which generates a new row for each of the value inside an array.

Next, every word is split inside the array as a new row. For this a UDTF (User Defined Table Generating Function) is used. A built-in UDTF called **explode** helps extract each element from an array and creates a new row for each element.

Another table is created that can store id and the hashtag text using the following command

```
create table hashtag_word as select id as id,hashtag from hashtags LATERAL
VIEW explode(words) w as hashtag;
```

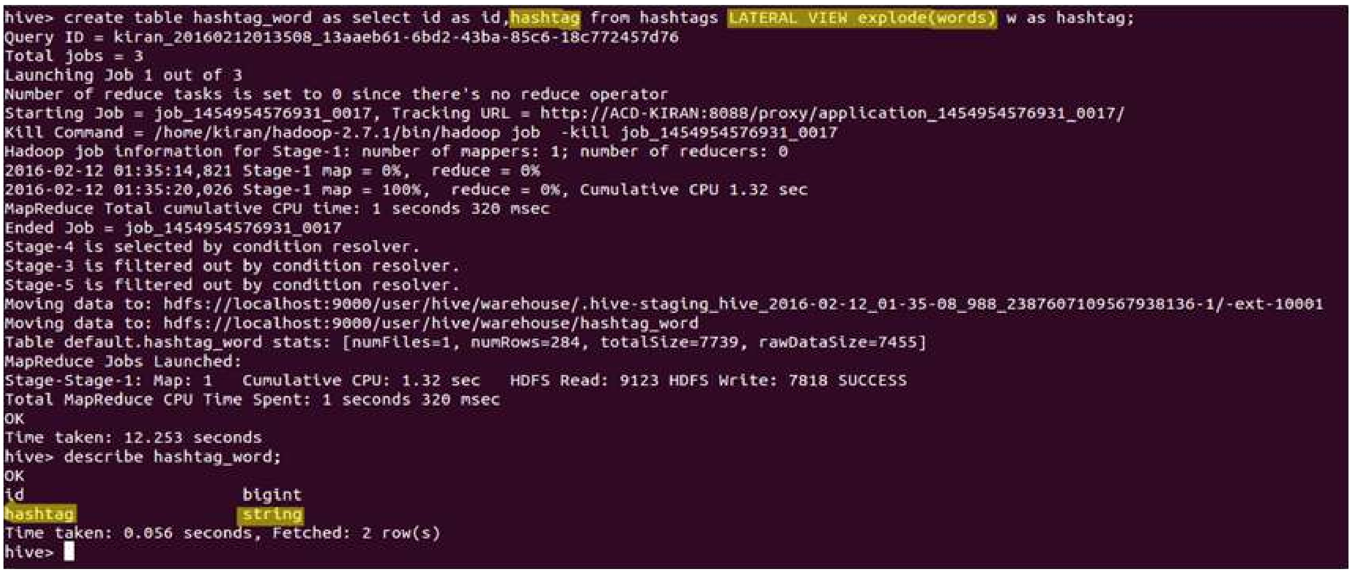
***Note:** Syntax for LATERAL VIEW explode UDTF is as follows:

```
lateralView: LATERAL VIEW udtf(expression) tableAlias AS columnAlias (','
columnAlias)*fromClause: FROM baseTable (lateralView)
```

In general, explode UDTF has some limitations; explode cannot be used with other columns in the same select relation. Hence, LATERAL VIEW needs to be added in conjunction with explode so that the explode function can be used in other columns as well.

The schema of the table can be viewed using the 'describe' command.

In **Exhibit 32**, it is seen that the array of values has been converted into a string. Contents of the table can be viewed by using the following command:

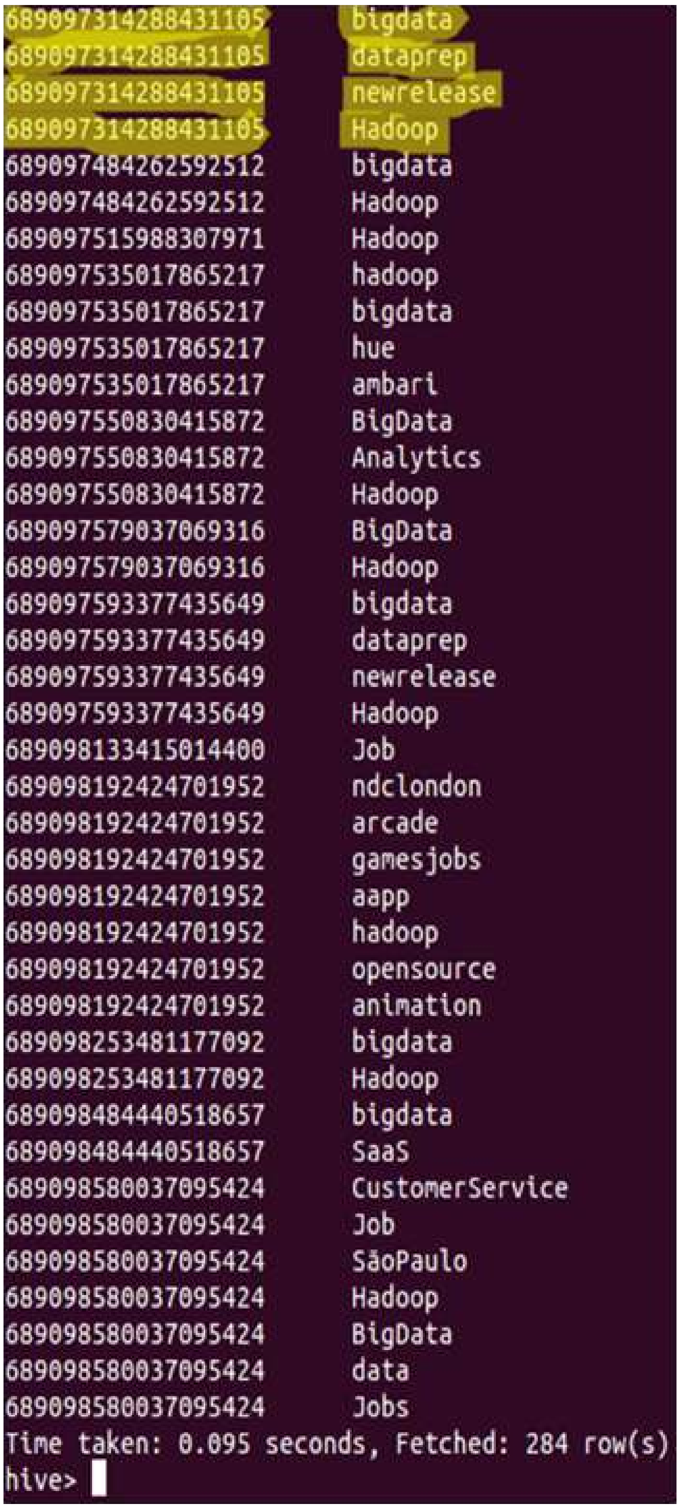


```
hive> create table hashtag_word as select id as id,hashtag from hashtags LATERAL VIEW explode(words) w as hashtag;
Query ID = kiran_20160212013508_13aaeb61-6bd2-43ba-85c6-18c772457d76
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454954576931_0017, Tracking URL = http://ACD-KIRAN:8088/proxy/application_1454954576931_0017/
Kill Command = /home/kiran/hadoop-2.7.1/bin/hadoop job  -kill job_1454954576931_0017
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-02-12 01:35:14,821 Stage-1 map = 0%,  reduce = 0%
2016-02-12 01:35:20,026 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1454954576931_0017
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/user/hive/warehouse/.hive-staging_hive_2016-02-12_01-35-08_988_2387607109567938136-1/-ext-10001
Moving data to: hdfs://localhost:9000/user/hive/warehouse/hashtag_word
Table default.hashtag_word stats: [numFiles=1, numRows=284, totalSize=7739, rawDataSize=7455]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.32 sec   HDFS Read: 9123 HDFS Write: 7818 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 320 msec
OK
Time taken: 12.253 seconds
hive> describe hashtag_word;
OK
id                      bigint
hashtag                 string
Time taken: 0.056 seconds, Fetched: 2 row(s)
hive>
```

Exhibit 32: Array of values has been converted into a string

```
select * from hashtag_word;
```

In **Exhibit 33,** it is seen that all the hashtags from one tweet are extracted as a separate row.



```
689097314288431105      bigdata
689097314288431105      dataprep
689097314288431105      newrelease
689097314288431105      Hadoop
689097484262592512      bigdata
689097484262592512      Hadoop
689097515988307971      Hadoop
689097535017865217      hadoop
689097535017865217      bigdata
689097535017865217      hue
689097535017865217      ambari
689097550830415872      BigData
689097550830415872      Analytics
689097550830415872      Hadoop
689097579037069316      BigData
689097579037069316      Hadoop
689097593377435649      bigdata
689097593377435649      dataprep
689097593377435649      newrelease
689097593377435649      Hadoop
689098133415014400      Job
689098192424701952      ndclondon
689098192424701952      arcade
689098192424701952      gamesjobs
689098192424701952      aapp
689098192424701952      hadoop
689098192424701952      opensource
689098192424701952      animation
689098253481177092      bigdata
689098253481177092      Hadoop
689098484440518657      bigdata
689098484440518657      SaaS
689098580037095424      CustomerService
689098580037095424      Job
689098580037095424      SãoPaulo
689098580037095424      Hadoop
689098580037095424      BigData
689098580037095424      data
689098580037095424      Jobs
Time taken: 0.095 seconds, Fetched: 284 row(s)
hive>
```

Exhibit 33: Extraction of hashtags into a separate row

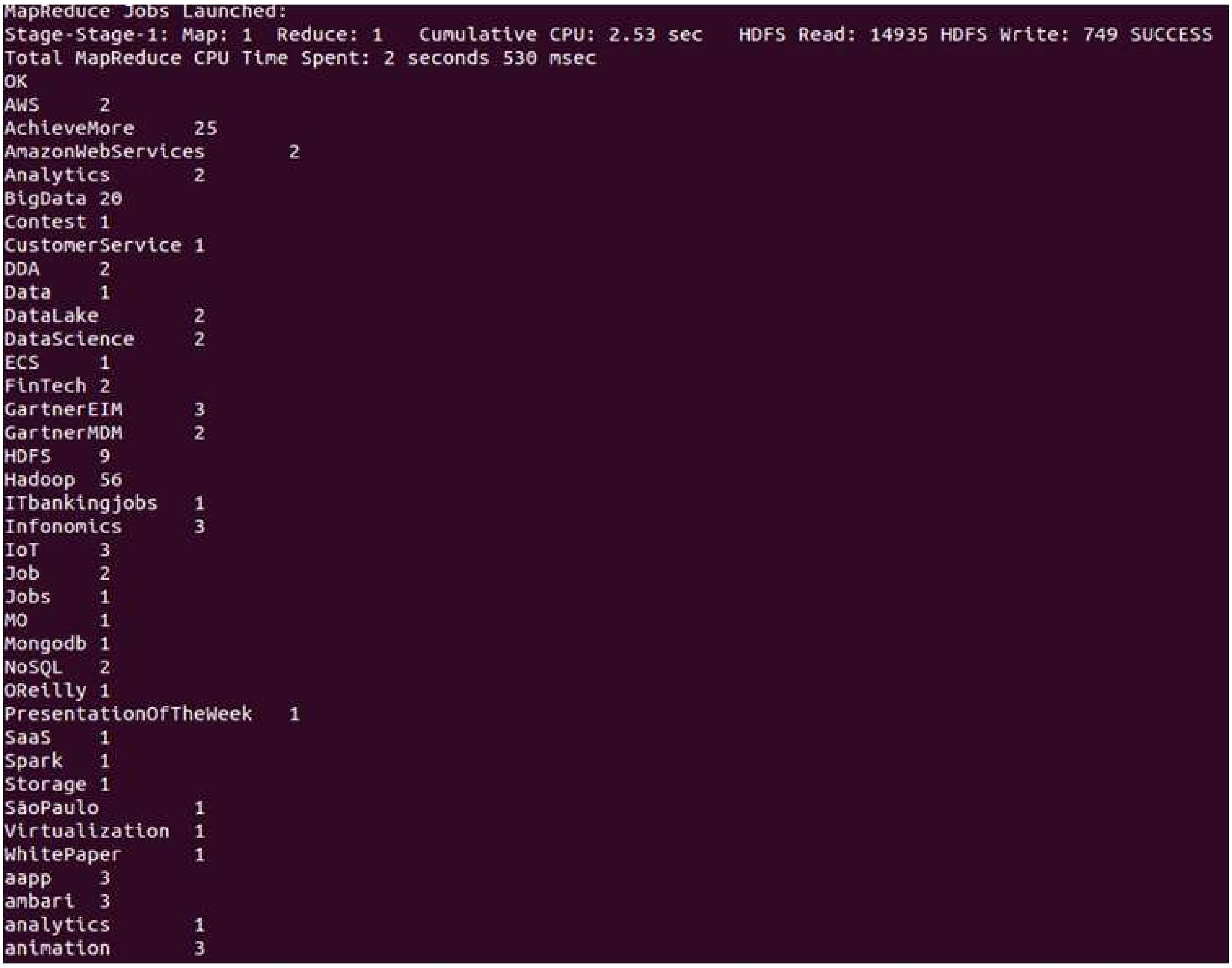
B) SAMPLE OUTPUT

A query has been used to calculate the number of times each hashtag has been repeated.

```
select hashtag, count(hashtag) from hashtag_word group by hashtag;
```

Exhibit 34 displays the number of times a hashtag has been repeated in the Twitter data.

Number of popular hashtags in Twitter using Hive have thus been counted.



```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.53 sec   HDFS Read: 14935 HDFS Write: 749 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 530 msec
OK
AWS     2
AchieveMore     25
AmazonWebServices       2
Analytics       2
BigData 20
Contest 1
CustomerService 1
DDA     2
Data    1
DataLake        2
DataScience     2
ECS     1
FinTech 2
GartnerEIM      3
GartnerMDM      2
HDFS    9
Hadoop  56
ITbankingjobs   1
Infonomics      3
IoT     3
Job     2
Jobs    1
MO      1
Mongodb 1
NoSQL   2
OReilly 1
PresentationOfTheWeek   1
SaaS    1
Spark   1
Storage 1
SãoPaulo        1
Virtualization  1
WhitePaper      1
aapp    3
ambari  3
analytics       1
animation       3
```

Exhibit 34: Number of repetitions of a hashtag in the Twitter data

In case you need any assistance or clarifications regarding the above use cases, feel free to drop a mail at support@acadgild.com.