CSC 643 – Big Data & Web Intelligence

$sudo Team1 -Ahmad/Sean

*Project 2*

Submission Date: October 28, 2019

Table of Contents

[**Task Ranking** 3](#_Toc23495455)

[**Introduction** 4](#_Toc23495456)

[**1) Data Description** 5](#_Toc23495457)

[**2) Rearrange Data Tuples** 6](#_Toc23495458)

[**3) Using Queries to Find the Result** 7](#_Toc23495459)

[3.1) On average, find the state that has highest precipitation in September 2013 7](#_Toc23495460)

[3.2) find the state that has lowest precipitation in 2013 8](#_Toc23495461)

[3.3) Find the state had more than an inch of precipitation on July 4, 2013 10](#_Toc23495462)

[**Appendix - Command List** 12](#_Toc23495463)

[**Solution 3.1** 12](#_Toc23495464)

[**Solution 3.2** 13](#_Toc23495465)

[**Solution 3.3** 14](#_Toc23495466)

[**References** 15](#_Toc23495467)

# **Task Ranking**

* Member 1: **Ahmad Alshawaf**

Handled 50% of the project from start to complete

* + - * Task 1: analyzing data
      * Task 2: solving problem 1
      * Task 3: solving problem 2
      * Task 4: solving problem 3
      * Task 5: organizing report
* Member 2: **Sean Sothey**

Handled 50% of the project from start to complete

* + - * Task 1: analyzing data
      * Task 2: solving problem 1
      * Task 3: solving problem 2
      * Task 4: solving problem 3
      * Task 5: organizing report

# **Introduction**

Data science has become an extremely rewarding career choice for people interested in extracting, manipulating, and generating insights out of large volumes of data. To fully leverage the power of data science, scientists often need to obtain skills in databases, statistical programming tools, and data visualizations. Companies surely need data scientists to help them empower their analytics processes, build a numbers-based strategy that will boost their bottom line, and ensure that enormous amounts of data are translated into actionable insights. In this project, we would like to introduce one of the most popular platform tools to analyze the data.

Apache Pig programs are written in a query language known as Pig Latin that is similar to the SQL query language. To execute the query, there is a need for an execution engine. The Pig engine converts the queries into MapReduce jobs and thus MapReduce acts as the execution engine and is needed to run the programs.

Apache Pig can be started either in cluster mode or in local mode. To start Apache Pig in local mode, the developers should use the option “-x local”. If no option is specified, then Pig by default is started in the cluster mode. The cluster mode allows Pig to access data file present on HDFS, whereas in local mode only files within the local file system can be accessed.

Therefore, we are going to run the Apache Pig on HDFS as a model in this study course to fulfill the assigned “Project 2”.

**Analyzing Data**

## **1) Data Description**

Referring to the reference at the end of the report, in **2. Element Names and Definitions**, we can see that there are 12 data fields in this data set.

Here is the summary table of these data fields:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FIELD** | **START POS** | **END POS** | **ELEMENT NAME** | **VALUE** |
| 001 | 1 | 3 | Record-Type | HPD |
| 002 | 4 | 11 | Station-ID | 13136306 |
|  | 4 | 5 | State-Code | 13 |
|  | 6 | 9 | Cooperative Network | 1363 |
|  | 10 | 11 | Cooperative Network Division Number | 06 |
| 003 | 12 | 15 | Element-Type | HPCP |
| 004 | 16 | 17 | Element-Units | HT |
| 005 | 18 | 21 | Year | 2013 |
| 006 | 22 | 23 | Month | 09 |
| 007 | 24 | 27 | Day | 0015 |
| 008 | 28 | 30 | Number-Reported-values | 004 |
| 009 | 31 | 34 | Time-of-Value | 0100 |
| 010 | 35 | 40 | Data-Value | 00160 |
| 011 | 41 | 41 | FLAG1 |  |
| 012 | 42 | 42 | FLAG2 |  |

Table : Data Identification

Now, let’s observe this first three data values formats.

HPD01419307HPCPHT20130900240031700 00150 1800 00010 2500 00160

HPD01419307HPCPHT20130900300020400 00010 2500 00010

HPD01420904HPCPHT20130900010030100 00000g 1500 00010 2500 00010

We can see that the format of the data is combined together in each row, and the data value is counted after 2500, also including FLAG1 and FLAG2. In this case, we are going to chunk this data format into different column, using substring method.

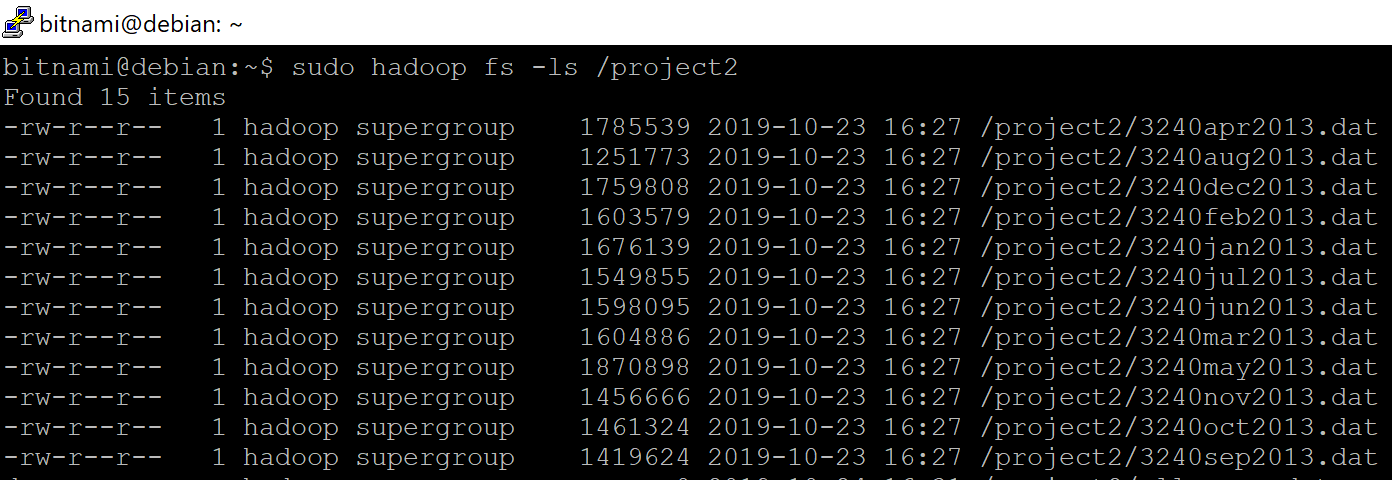
## **2) Rearrange Data Tuples**

* **Step 1**: There are two locations to load the data in this course, one is from local machine, and the other one is from HDFS. For this project, as mentioned from the beginning, we chose to work on HDFS. Therefore, first we copy all the data files from local machine (Bitnami) to HDFS, into a directory named “project2”, using the following command:

**$sudo hadoop fs -put '/home/bitnami/project2/' '/project2'**

Check if the copied files exist in HDFS directory, using the following command:

**$sudo hadoop fs -ls /project2**



* **Step 2**: Loading data (only September data) into Pig and store as “chararray”, using the following command:

**lines = LOAD 'hdfs://localhost:8020/project2/3240sep2013.dat’ AS (line:chararray);**

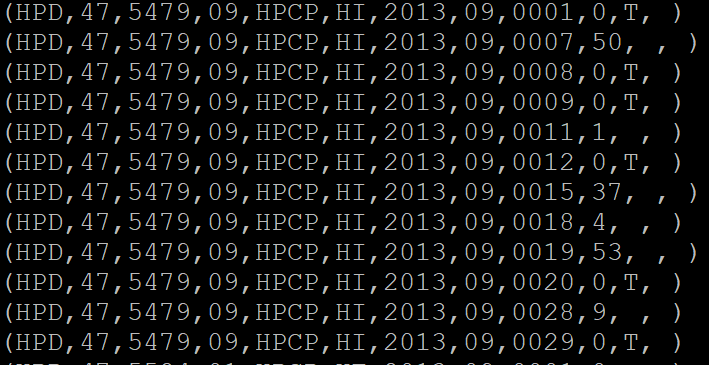
* **Step 3**: Chunking data fields into different column, separated by “tap” value according to Table 1, using the following command:

**sept\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division, SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit, SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(int) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value ,SUBSTRING(line,(int)SIZE(line)-2,(int)SIZE(line)-1) AS flag1, SUBSTRING(line,(int)SIZE(line)-1,(int)SIZE(line)) AS flag2;**

* **Step 4**: Filtering data value, using following command**:**

**filtered\_sept\_data = FILTER sept\_data BY flag1 matches '[^I]';**

**new tuples output:**



## **3) Using Queries to Find the Result**

### 3.1) On average, find the state that has highest precipitation in September 2013

Using the new tuples from **filtered\_sept\_data** from step 4, section 2.

* **Step 1**: Group the filtered data by state, using the following command:

**grouped\_by\_state = GROUP filtered\_sept\_data BY state;**

* **Step 2**: Find the average of each state, using the following command:

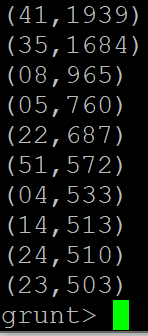
**avg\_of\_state = FOREACH grouped\_by\_state GENERATE group as state,**

**SUM(filtered\_sept\_data.data\_value)/30 as avg\_value;**

* **Step 3**: Show the average order by descending and limit 10 results only, use the following commands**:**

**order\_by\_avg = ORDER avg\_of\_state BY avg\_value DESC;**

**max\_avg\_10 = LIMIT order\_by\_avg 10;**

**dump max\_avg\_10;**

Therefore, the highest average

precipitation in September 2013

goes with state ID 41, which is **Texas**

**,** had **19.39** inches**.**

### 3.2) find the state that has lowest precipitation in 2013

Using same process in section 2, except loading all files instead of only the “sept file”.

* **Step 1**: Loading all data in 2013 into Pig and store as “chararray”, using command:

**lines = LOAD 'hdfs://localhost:8020/project2/\*.dat' AS (line:chararray);**

* **Step 2**: Do the same as **Step 3** in section **2.**

**all\_year\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division,SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(double) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value;**

* **Step 3**: Group the filtered data by state, using command:

**grouped\_by\_state = GROUP all\_year\_data BY state;**

* **Step 5**: Sum all the precipitation data value by each state, using command:

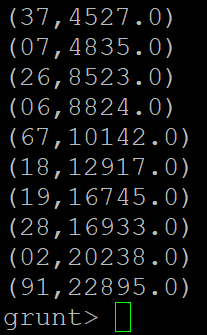
**sum\_of\_state = FOREACH grouped\_by\_state GENERATE group as state, SUM(all\_year\_data.data\_value) as sum\_value;**

* **Step 6**: Show the sum order by ascending and limit 10 results only, use commands**:**

**order\_by\_sum\_of\_state = ORDER sum\_of\_state BY sum\_value ASC;**

**lowest\_sum\_of\_state\_10 = LIMIT order\_by\_sum\_of\_state 10;**

**dump lowest\_sum\_of\_state\_10;**

Therefore, the lowest precipitation in 2013

goes with state ID 37, which is **Rhode Island**,

had only **45.27** inches.

### 3.3) Find the state had more than an inch of precipitation on July 4, 2013

Using same process in section 2, by loading only the “july file”.

* **Step 1**: Loading data (only July) into Pig and store as “chararray”, using command:

**lines = LOAD 'hdfs://localhost:8020/project2/3240jul2013.dat’ AS (line:chararray);**

* **Step 2**: Chunk data fields into different column, using the following command:

**jul\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division,SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(double) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value;**

* **Step 3**: Filter out by keeping only the day is 4th, using the following command:

**july\_4\_data = FILTER jul\_data BY day == '0004';**

* **Step 4**: Group the filtered data by state, using command:

**grouped\_by\_state = GROUP july\_4\_data BY state;**

* **Step 5**: Sum all the precipitation data value by each state, using command:

**sum\_of\_state = FOREACH grouped\_by\_state GENERATE group as state, SUM(filtered\_jul\_data.data\_value)/100 as sum\_value;**

*\*\*\* Note: the value is divided by 100 because the original data value is scaled 1/100 (inch)*

*Example: if the data value is 00030, the precipitation value is 0.3 inches.*

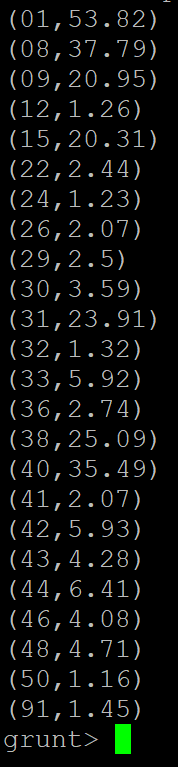
* **Step 6**: Show the states that had more than 1 inch, using the following command:

**states\_has\_more\_inch\_pres = FILTER sum\_of\_state BY sum\_value > 1; dump states\_has\_more\_inch\_pres;**

`

**Therefore, the states that had precipitation**

**more than 1 inch are:**

**state ID:01 is Alabama, 53.82 inches**

**state ID:08 is Delaware, 37.79 inches**

**state ID:09 is Georgia, 20.95 inches**

**state ID:15 is Kentucky, 20.31 inches**

**state ID:22 is Mississippi, 2.44 inches**

**state ID:24 is Montana, 1.23 inches**

**state ID:26 is Nevada, 2.07 inches**

**state ID:29 is New Mexico, 2.5 inches**

**state ID:30 is New York, 3.59 inches**

**state ID:31 is North Carolina, 23.91 inches**

**state ID:32 is North Dakota, 1.32 inches**

**state ID:33 is Ohio, 5.92 inches**

**state ID:36 is Pennsylvania, 2.74 inches**

**state ID:38 is South Carolina, 25.09 inches**

**state ID:40 is Tennessee, 35.49 inches**

**state ID:41 is Texas, 2.07 inches**

**state ID:42 is Utah, 5.93 inches**

**state ID:43 is Vermont, 4.28 inches**

**state ID:44 is Virginia, 6.41 inches**

**state ID:46 is West Virginia, 4.08 inches**

**state ID:48 is Wyoming, 4.71 inches**

**state ID:50 is Alaska, 1.16 inches**

**state ID:91 is Pacific Islands, 1.45 inches**

# **Appendix - Command List**

## **Solution 3.1**

* $sudo hadoop fs -put '/home/bitnami/project2/' '/project2'
* $sudo hadoop fs -ls /project2
* lines = LOAD 'hdfs://localhost:8020/project2/ 3240sep2013.dat’ AS (line:chararray);
* sept\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division, SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit, SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(int) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value ,SUBSTRING(line,(int)SIZE(line)-2,(int)SIZE(line)-1) AS flag1, SUBSTRING(line,(int)SIZE(line)-1,(int)SIZE(line)) AS flag2;
* filtered\_sept\_data = FILTER sept\_data BY flag1 matches '[^I]';
* grouped\_by\_state = GROUP filtered\_sept\_data BY state;
* avg\_of\_state = FOREACH grouped\_by\_state GENERATE group as state,

SUM(filtered\_sept\_data.data\_value)/30 as avg\_value;

* order\_by\_avg = ORDER avg\_of\_state BY avg\_value DESC;

max\_avg\_10 = LIMIT order\_by\_avg 10;

* dump max\_avg\_10;

## **Solution 3.2**

* lines = LOAD 'hdfs://localhost:8020/project2/\*.dat' AS (line:chararray);
* all\_year\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division,SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(double) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value;
* grouped\_by\_state = GROUP all\_year\_data BY state;
* sum\_of\_state = FOREACH grouped\_by\_state GENERATE group as state, SUM(all\_year\_data.data\_value) as sum\_value;
* order\_by\_sum\_of\_state = ORDER sum\_of\_state BY sum\_value ASC;
* lowest\_sum\_of\_state\_10 = LIMIT order\_by\_sum\_of\_state 10;
* dump lowest\_sum\_of\_state\_10;

## **Solution 3.3**

* lines = LOAD 'hdfs://localhost:8020/project2/ 3240sep2013.dat’ AS (line:chararray);
* jul\_data = FOREACH lines GENERATE SUBSTRING(line,0,3) AS record\_type,SUBSTRING(line,3,5) AS state,SUBSTRING(line,5,9) AS co\_network,SUBSTRING(line,9,11) AS co\_division,SUBSTRING(line,11,15) AS element\_type,SUBSTRING(line,15,17) AS element\_unit,SUBSTRING(line,17,21) AS year,SUBSTRING(line,21,23) AS month,SUBSTRING(line,23,27) AS day,(double) SUBSTRING(line,(int)SIZE(line)-8,(int)SIZE(line)-2) as data\_value;
* july\_4\_data = FILTER jul\_data BY day == '0004';
* grouped\_by\_state = GROUP july\_4\_data BY state;
* sum\_of\_state = FOREACH grouped\_by\_state GENERATE group as state, SUM(filtered\_all\_year\_data.data\_value) as sum\_value;
* order\_by\_sum\_of\_state = ORDER sum\_of\_state BY sum\_value ASC;

lowest\_sum\_of\_state\_10 = LIMIT order\_by\_sum\_of\_state 10;

* dump lowest\_sum\_of\_state\_10;

# **References**

Data Description

<ftp://ftp.ncdc.noaa.gov/pub/data/hourly_precip-3240/dsi3240.pdf>

Data of Each Month in 2013

<ftp://ftp.ncdc.noaa.gov/pub/data/hourly_precip-3240/by_month2013/>

Data Provider

<https://www.data.gov/>