# Pig Introduction

DS 730

## Overview

In this activity, you will be running some Pig code on the Hortonworks system. You must be very careful with capitalization. A file called Baseball.csv cannot be referenced to as baseball.csv nor can a variable called agged be used as Agged.

### Task 1: Practice Basic Pig Latin Statements

We will spend quite a bit of time going through the Pig Latin syntax. In order to start, go to your command line in Hortonworks (http://localhost:4200) and log in as **maria_dev**. Each step will contain a statement and an explanation of the statement. We will do a bit of command line processing in this activity.

1. Type in the following to get the Pig sample files onto your Hortonworks system[1]:
   **wget http://www.uwosh.edu/faculty_staff/krohne/ds730/pigFiles.tar.gz**
2. Decompress the file by using the following command:
   **tar xvzf pigFiles.tar.gz**
3. Move those files to the HDFS into **/user/maria_dev/pigtest/**. This could be accomplished via the Ambari Files View page if you extracted them locally using your local filesystem. Alternatively, we can use the command line. In order to do this, type in:

   **cd PigSamples**

   and then type in

   **ls**

   You should see 3 files listed, a Batting.csv, Master.csv and Complaints.csv. Type in the following command to create the folder on HDFS:

   **hdfs dfs -mkdir /user/maria_dev/pigtest**

   In order to copy the files to HDFS, use the following command:

   **hdfs dfs -copyFromLocal *.csv /user/maria_dev/pigtest/**

[1] Baseball datasets used with Creative Commons Attribution-ShareAlike 3.0 Unported License from http://www.seanlahman.com/baseball-archive/statistics/. Complaints dataset from http://catalog.data.gov/dataset/consumer-complaint-database

4. First, open up the csv files so you can get a handle on what is saved in each file. As we are solving each one of these problems, you should think about how difficult it would be to do this using MapReduce. Similarly, if your file size is on the order of terabytes or higher, Microsoft Excel or some other spreadsheet software would struggle to even open it.
5. Type **`pig`** and press **ENTER**. You will see some INFO lines that you can safely ignore. As long as you see **grunt>** at the end, you are ready to continue. In the next few steps, type in exactly what is written into the grunt shell (**grunt>**).
6. 
```
Complaints = LOAD
'hdfs:/user/maria_dev/pigtest/Complaints.csv'
USING PigStorage(',') AS (date:chararray,
product:chararray, subprod:chararray,
issue:chararray, subissue:chararray,
narrative:chararray, response:chararray,
company:chararray, state:chararray,
zip:chararray, submitted:chararray,
senttocomp:chararray, compresponse:chararray,
timelyresponse:chararray, disputed:chararray,
id:int);
```

   **Explanation:**

   - The first **Complaints** id is simply the name of the relation I am using. It can be called anything you want.
   - The **LOAD** call is a built-in command to load the data from the file. The name of the file follows the **LOAD** command.
   - Then we need to tell Pig how to parse it. Since it's comma delimited, we use the **USING PigStorage(',')** command.
   - Finally, we give a schema to the relation. Instead of having to reference columns by their position in the relation, it's easier to give them names and types.
   - The actual load will not happen until we need it.
7. 
```
DESCRIBE Complaints;
```

   **Explanation**: This will print out our schema. It should print out something like:

```
Complaints: {date: chararray,product: chararray,subprod:
chararray,issue: chararray,subissue: chararray,narrative:
chararray,response: chararray,company: chararray,state:
chararray,zip: chararray,submitted: chararray,senttocomp:
chararray,compresponse: chararray,timelyresponse:
chararray,disputed: chararray,id: int}
```

8. **`Oshkosh = FILTER Complaints BY state=='WI' AND zip=='54904';`**

   **Explanation**: This will filter all tuples where the state is equal to WI and the zip code is equal to 54904. Oshkosh is now a relation that only contains tuples where state is Wisconsin and the zip is 54904.

   **Note:** All of your favorite **AND**, **OR** and **NOT** work in the expressions. You can read about them here: http://pig.apache.org/docs/r0.17.0/basic.html#boolops

9. **`DUMP Oshkosh;`**

   **Explanation**: This command will print out the relation to the screen. It will start a MapReduce program to run through all of this for us. There should be 19 records that print out. Instead of having to count them, you can use the following commands (explained in the future) and it should print out (19):

   ```
   grouped = GROUP Oshkosh ALL;
   total = FOREACH grouped GENERATE COUNT(Oshkosh);
   DUMP total;
   ```

10. **`ContainsAnIOrE = FILTER Complaints BY (state MATCHES '.*I.*') OR (state MATCHES '.*E.*');`**

    **Explanation**: This will filter all tuples where the state abbreviation contains an I or if the state abbreviation contains an E. It is a way to do regular expressions to filter what you need.

11. **`STORE Oshkosh INTO 'hdfs:/user/maria_dev/pigtest/wisconsin' USING PigStorage(':', '-schema');`**

**Explanation**:

- We are creating a MapReduce program to write out our Oshkosh relation to some (possibly more than 1) files. As with MapReduce outputs, the folder must *not* have been created.
- We are using **`PigStorage`** as a way to simplify our code. We want to output our relation using a different delimiter this time. We will delimit with a colon instead of a comma.
- The last **`-schema`** tag will save our schema to a file called **.pig_schema**. This will be great for future loads because we won't have to specify a schema each time. If the **.pig_schema** file is already in the folder we are reading in, that will be the schema that is used. This lets us do away with everything after the **`AS`** command in step 1. Our Oshkosh relation will be stored in some number of **part-xxxxx** files. Mine turned out to only save into 1 file. Keep track of this wisconsin folder as you will be uploading it at the end of this activity.

12. **`readable = FOREACH Oshkosh GENERATE date, product, company, compresponse;`**

    **Explanation**: We may not be interested in all of the data from the original relation. If so, we can only include the columns we want. For instance, I may only want the date, product, company and what the company responded with. We can run this and then run a **`DUMP`** readable to see what the relation is.

13. **`DUMP readable;`**
14. **`firstfilter = FOREACH Complaints GENERATE date, product, company, state, zip, submitted;`**

    **Explanation**: We may only care about some of the columns.

15. **`filtered = FILTER firstfilter BY state IS NOT NULL;`**

    **Explanation**: Cleans up some of the bogus data that gets read in. Ensures that all states are not an empty string.

16. **`groupedbystate = GROUP filtered BY state;`**

**Explanation**: This will group all of our tuples by the state value so that we can run aggregate functions on them. This **groupedbystate** relation contains two columns, a group column that is the name of the state and a bag of tuples.

> **Example:** If we look at a tuple where group==X, every tuple in the bag of tuples is a tuple in the original filtered relation where state==X. This will be explained a second time with the next command if it didn't make sense now.

17. **`DESCRIBE groupedbystate;`**

**Explanation:**

- You'll notice the first column is called group. This group is the same type as state because that is what we grouped by. We will use this group column in our aggregate functions a bit later.
- The second column is a subset of the filtered relation. The second column, which is called a bag, is all of the tuples of filtered that have "group" as its state column.

> **Example:** Assume you have a tuple in **`groupedbystate`** that has its group value set to WI. The second column of that **`groupedbystate`** tuple is going to be a bag of tuples such that each tuple (from filtered) has a state value of WI.
>
> Assuming this **`DESCRIBE`** made sense, the next statement should be rather straightforward. Don't move on to the next step unless this step is crystal clear. Look at the example here if this is not clear:
> http://pig.apache.org/docs/r0.17.0/basic.html#group
>
> You may have to scroll down a little bit but the example starts with "Suppose we have relation A" and it talks about names, ages and gpa. It is a very good example.

18. **`agged = FOREACH groupedbystate GENERATE group, COUNT(filtered) AS total;`**

**Explanation:** The **FOREACH** is straightforward; for every tuple in **groupedbystate**, do something… we want to create a new tuple. Remember, each tuple has a bag of tuples that all have the same state. For each tuple, generate a tuple in our new relation. The first column in our new relation is simply the name of the state since that is what we grouped by in the previous step.

**COUNT** is a built in function that simply counts all of the tuples in the second column (which is a bag of tuples).

19. **`sorted = ORDER agged BY total DESC;`**

    **Explanation:** Sort all of the states by the ones who have the most complaints. Sort them in descending order.

20. **`topten = LIMIT sorted 10;`**

    **Explanation:** Limit the number of states that end up in our output to 10.

21. **`DUMP topten;`**

    **Explanation:** Should end up printing California, followed by Florida, followed by Texas and the rest of the top ten.

22. **`wisc = FILTER agged BY group=='WI';`**
23. **`DUMP wisc;`**
24. **`webandphone = FILTER filtered BY (submitted=='Web' OR submitted=='Phone') AND state=='WI';`**
25. **`groupbymore = GROUP webandphone BY (state,submitted);`**

    **Explanation:** We can group tuples from our relation using multiple values. Here I am interested in not only how many complaints are from Wisconsin, but how many of them were submitted via the web and via phone. Because I know I only care about those two options, I filtered them out first (step 18) before working with them. The more filtering I can do early on, the better. You always want to filter first, group and aggregate second.

26. **`agged = FOREACH groupbymore GENERATE group, COUNT(webandphone) AS total;`**

    **Explanation:** Aggregate all of the results into a simple relation we can read.

27. **`dump agged;`**

## Task 2: Practice More Detailed Pig Latin Examples

The previous task covered the basic Pig Latin statements. In this task, we will go over some more detailed examples. We will look at a different example where we have separate csv files and we want to join them together in some fashion.

### Enter Commands at Pig Grunt Shell

1. **`batters = LOAD 'hdfs:/user/maria_dev/pigtest/Batting.csv' using PigStorage(',');`**

   **Explanation:** We will load up the data without specifying a schema as the number of columns may be huge and we may only need a subset of them.

2. **`realbatters = FILTER batters BY $1>0;`**

   **Explanation:** If we do not specify a schema, we can reference the columns in the relation by column numbers starting at 0. The year column is therefore referenced by **$1**. If a batter doesn't have a year, we want to filter them out.

3. **`run_data = FOREACH realbatters GENERATE $0 AS id, $1 AS year, $6 AS runs;`**

   **Explanation:** We are creating a new relation that only contains the player's id, the year and the total number of runs scored that year.

4. **`grouped_by_year = GROUP run_data BY year;`**
5. **`best_per_year = FOREACH grouped_by_year GENERATE group, MAX(run_data.runs) AS best;`**

   **Explanation:** Here we are looking at each tuple in our grouping. Each tuple has a group key of the year. Each tuple in our bag contains a person who played during that year. We are interested in the person who scored the most runs in that particular year. If two players had the same number of runs, both will show up in our relation.

6. **`DUMP best_per_year;`**
7. **`get_player_ids = JOIN best_per_year BY ($0, best), run_data BY (year,runs);`**

**Explanation:** In order to join the maximum runs back up with the player that got it, we need to do a join. We are joining our new relation that has the $0 (which is the year) and best (which is a number of runs) with our original relation with (year, runs). Every time the ($0, best) exactly equals the (year, runs), the tuples will combine. This will give us multiple rows if two players had the same number (see 1967).

**Example:** In our **`best_per_year`**, we have **`(2011, 136.0)`**. In our **`run_data`** relation, we have **`(grandcu01,2011,136)`**. Since 2011==2011 and 136==136, we get a new tuple in our joined relation consisting of **`(2011,136.0,grandcu01,2011,136)`**. However, we probably don't want them in this order nor do we want duplicate column values, so… (continue with next steps)

8. **`nicer_data = FOREACH get_player_ids GENERATE $0 AS year, $2 AS id, $4 AS runs;`**
9. **`DUMP nicer_data;`**

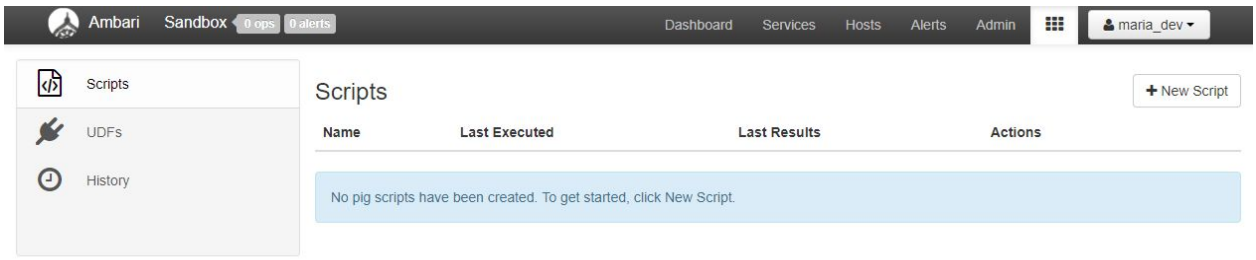
**Explanation**: The data looks good but the id may not mean anything to some people. Therefore, we need to join up with the master relation and get the names of the players instead of the ids. The following steps are everything we've already seen but they are done with a different file. Therefore, no explanation is given.

10. **`names = LOAD 'hdfs:/user/maria_dev/pigtest/Master.csv' using PigStorage(',');`**
11. **`master_data = FOREACH names GENERATE $0 AS id, $13 AS first, $14 AS last;`**
12. **`complete_data = JOIN nicer_data BY id, master_data BY id;`**
13. **`finished = FOREACH complete_data GENERATE $0 AS year, $4 AS first, $5 AS last, $2 AS runs;`**
14. **`sorted = ORDER finished BY year DESC;`**
15. **`DUMP sorted;`**

When you are finished, type in **quit;** and you will quit out of the grunt shell.

## Run Scripts

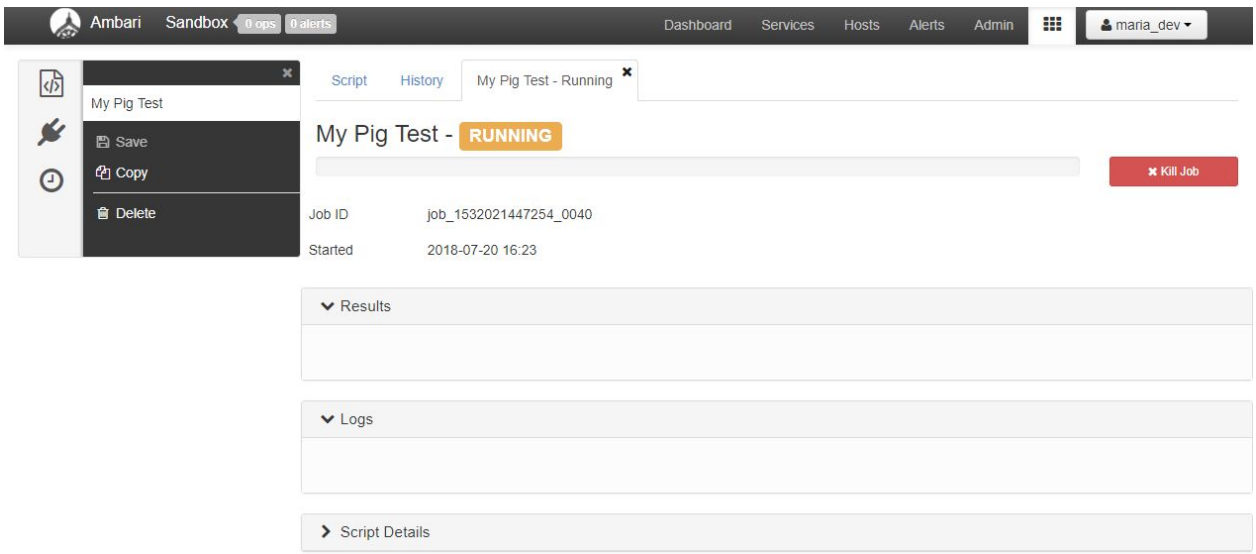
1. The grunt shell was nice for debugging/testing our code one step at a time. However, we will be creating pig scripts to simply run everything without any interaction from us. We will run our scripts from the Hortonworks Dashboard. You are also welcome to use the Dashboard for all of your Pig testing.
2. Open up the Dashboard by going to http://localhost:8080. Log in using **maria_dev**. Click on the 9 boxes in the upper right hand corner and select the **Pig View** option. You should see something similar to this:




3. Click on **+New Script**. Enter in anything you want for the script name. Leave the HDFS location part empty. Type in the following basic commands and then hit the Execute button:

```
batters = LOAD 'hdfs:/user/maria_dev/pigtest/Batting.csv'
using PigStorage(',');
DUMP batters;
```

4. You will see something that looks like this:



5. Eventually the script will complete and results will be displayed in the *Results* section.
6. Click on the **Script** tab and enter in the script on the top of the next page. The script will store data into a folder called baseballsorted and will also dump the results out so we can see them in the results section. The following script took about 90 seconds to run on my machine. Once the script is finished, look at the /user/maria_dev/pigtest/baseballsorted folder and you will see at least 1 part file with the answers stored in it/them.

```
batters = LOAD 'hdfs:/user/maria_dev/pigtest/Batting.csv'
using PigStorage(',');
realbatters = FILTER batters BY $1>0;
run_data = FOREACH realbatters GENERATE $0 AS id, $1 AS
year, $6 AS runs;
grouped_by_year = GROUP run_data BY year;
best_per_year = FOREACH grouped_by_year GENERATE group,
MAX(run_data.runs) AS best;
get_player_ids = JOIN best_per_year BY ($0, best), run_data
BY (year,runs);
nicer_data = FOREACH get_player_ids GENERATE $0 AS year, $2
AS id, $4 AS runs;
names = LOAD 'hdfs:/user/maria_dev/pigtest/Master.csv'
using PigStorage(',');
master_data = FOREACH names GENERATE $0 AS id, $13 AS
first, $14 AS last;
complete_data = JOIN nicer_data BY id, master_data BY id;
finished = FOREACH complete_data GENERATE $0 AS year, $4 AS
first, $5 AS last, $2 AS runs;
sorted = ORDER finished BY year DESC;
STORE sorted INTO
'hdfs:/user/maria_dev/pigtest/baseballsorted' USING
PigStorage(',');
DUMP sorted;
```

**Note:** We have seen a lot of the syntax that you will need for the majority of your pig scripts. For more information on anything else you may want to know about Pig, go to http://pig.apache.org/docs/r0.17.0/ and you will find a ton of information about Pig. I recommend starting here: http://pig.apache.org/docs/r0.17.0/basic.html.

## Task 3: Using a UDF (user defined function) in Python.

All of the code you have to write in this course can be done without a UDF. However, if you ever find yourself needing to create one, this is the process to get it working with Pig using Hortonworks.

1. Open up Notepad++ and create a UNIX file called **baseball.py**
2. Copy the following function into your baseball.py file. The function accepts 3 arguments, all of them numbers. Assuming they are ordered first, second and third, your function is going to return **first*.4 + second*.5 + third*.2**.

```
@outputSchema('value:double')
def getVal(first, second, third):
  return first*.4+second*.5+third*.2
```

3. Go to your Files View and upload the baseball.py file into the **/user/maria_dev/pigtest** folder.
4. Go back to the Pig View. Click on UDFs and click on the **+Create UDF** button. Call it BaseballTest and use the /user/maria_dev/pigtest/baseball.py path.
5. Now we will see how this UDF can be used with the baseball example. Many of these steps have no explanation as they have already been explained.
6. `REGISTER 'hdfs:/user/maria_dev/pigtest/baseball.py' USING jython AS myfuncs`
7. `batters = LOAD 'hdfs:/user/maria_dev/pigtest/Batting.csv' using PigStorage(',');`
8. `realbatters = FILTER batters BY $1>0;`
9. `data = FOREACH realbatters GENERATE $0 AS id, (int)$1 AS year:int, (int)$8 AS doubles:int, (int)$9 AS triples:int, (int)$10 AS hr:int;`
10. `filtereddata = FILTER data BY year==2011;`
11. `answer = FOREACH filtereddata GENERATE myfuncs.getVal(doubles, triples, hr), id;`
    - Here we are calling our UDF in the **baseball.py** file. The name of my function is called **getVal**. We are passing in the number of doubles, triples and hr from our **filtereddata** relation and getting back some calculated value. In our calculation, we value triples the most followed by doubles and finally followed by home runs. We can tweak that computation to however we want it. Along with that value, we are getting the user id of the person with that value.

12. `DUMP answer;`
13. `DESCRIBE answer;`
    - When doing this, we got the following back:
      **answer: {value: double,id: bytearray}**
      This value was created in the Python script. Recall from the Python code:
      `@outputSchema('value:double')`
      Therefore, the column is called *value*. This is used in the next step.
14. `best_player = ORDER answer BY value DESC;`
15. `top_ten = LIMIT best_player 10;`
16. `DUMP top_ten;`

## Task 4: Run Pig Scripts on AWS

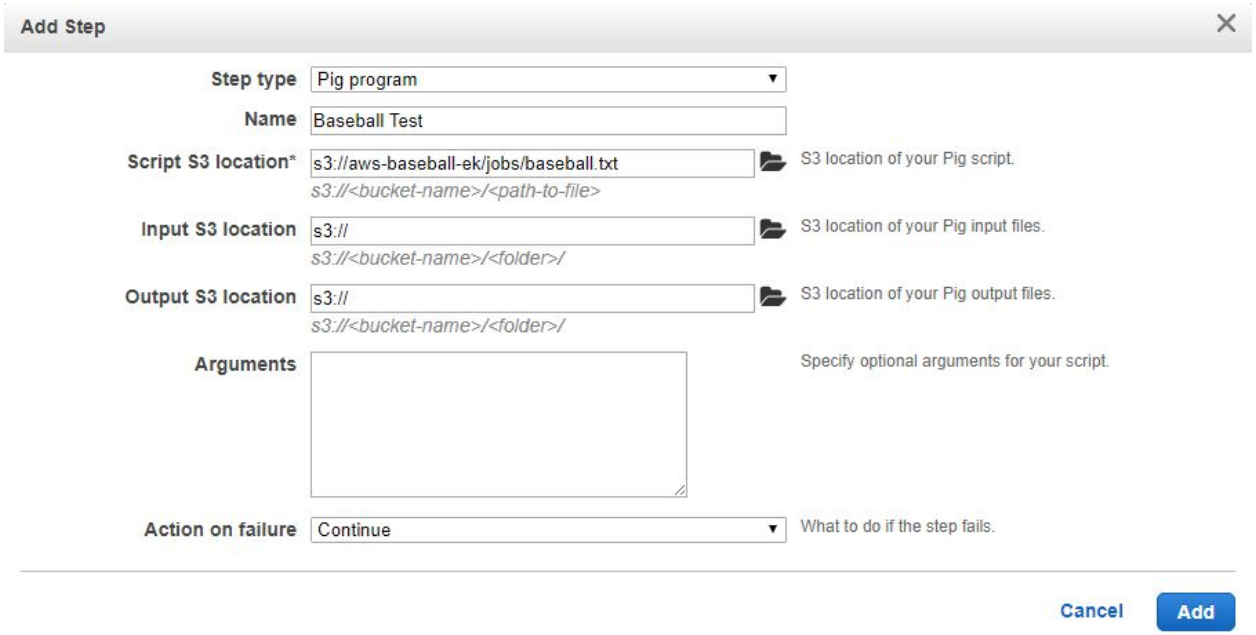
1. Go to your S3 manager on AWS and create a new bucket.
   - You can call the bucket whatever you want, but take note of the name, as you'll need it shortly.
2. Upload the **Batting.csv** and **Master.csv** files to your bucket, preferably in an input folder.
3. Modify the following code to change BUCKET to your bucket name, and then save the code in a UNIX file called **baseball.txt**:

```
batters = LOAD 's3://BUCKET/InputFolder/Batting.csv' using
PigStorage(',');
realbatters = FILTER batters BY $1>0;
run_data = FOREACH realbatters GENERATE $0 AS id, $1 AS year,
$6 AS runs;
grouped_by_year = GROUP run_data BY year;
best_per_year = FOREACH grouped_by_year GENERATE group,
MAX(run_data.runs) AS best;
get_player_ids = JOIN best_per_year BY ($0, best), run_data BY
(year,runs);
nicer_data = FOREACH get_player_ids GENERATE $0 AS year, $2 AS
id, $4 AS runs;
names = LOAD ' s3://BUCKET/InputFolder/Master.csv' using
PigStorage(',');
master_data = FOREACH names GENERATE $0 AS id, $13 AS first,
$14 AS last;
```

```
complete_data = JOIN nicer_data BY id, master_data BY id;
finished = FOREACH complete_data GENERATE $0 AS year, $4 AS
first, $5 AS last, $2 AS runs;
sorted = ORDER finished BY year DESC;
STORE sorted INTO 's3://BUCKET/baseballsorted' USING
PigStorage(',');
```

When you have everything uploaded to S3, do the following to run a Pig program:

- Log in to your AWS console and click on the EMR icon.
- Click on the **Create Cluster** button. Change the name of the cluster to be whatever you want it to be. The only thing you have to do is change the EC2 key pair to be the key you created in week 1 (or choose to proceed without a key pair). Everything else can stay at their default values. Click **Create Cluster**.
- At the top of the next page, click on the **Add Step** button.
- This time we are running a Pig program so change the **Step type** to be a Pig program. You can change the name to be whatever you would like. The only thing you need to do is change the **Script S3 location** to be the location you uploaded your baseball.txt file from the previous step. You do not need to put in an input or output because it is specified in your Pig script. Click on the **Add** button.



4. When the previous Pig script is done executing, you should have your results stored in your s3://BUCKET/baseballsorted folder. When you are finished, go back to the Cluster List and be sure to shut down (terminate) your cluster.

## Task 5: Solve Baseball Problems Using Pig Code

Solve the following problems using the batting and master table and create a script to solve each problem. You should assume that:

- **Batting.csv** will be stored in this **hdfs** folder: **/user/maria_dev/pigtest**
- **Master.csv** will be stored in this **hdfs** folder: **/user/maria_dev/pigtest**

Make sure to load your tables correctly in each script given that information. You should create 1 script per problem and call your script **PX.pig** where **X** is the question number you are solving. For each question, **print out only the playerID(s) of the player(s) who answer the question**. If there are any ties, report all players who tied. Do not print out extra information. For example, the code for number 1 should DUMP only the playerID. Do not output the weight or number of triples. You should dump your output to the terminal window for each question. Do not use the STORE command for any of these problems.

| Problem # | Description |
|---|---|
| **1** | Who was the heaviest player to hit more than 5 triples (3B) in 2005? |
| **2** | In the batting file, if a player played for more than 1 team in a season, that player will have his name show up in multiple tuples with the same year. For example, in 2011, Francisco Rodriguez (**rodrifr03**) played for the New York Mets and then played for the Milwaukee Brewers (see tuples **95279** and **95280**). The question you have to answer is this: what player played for the most teams in any single season? A player may have played for the same team twice in the same season at different times in the season. If this is the case, you should count this as two different teams. |
| **3** | What player had the most extra base hits during the entire 1980's (1980 to 1989)? Note that this question is not asking about any 1 specific year. It is asking about the entire 10 year span in the 80's. An extra base hit is a double, triple or home run (columns **2B**, **3B**, **HR**). |

| | |
|---|---|
| **4** | Of the right-handed batters who were born in October and died in 2011, which one had the most hits in his career? The column with the heading of **H** is the hits column. Do not consider switch hitters to be right-handed batters. |

## Submitting Your Work

1. Submit the `wisconsin` folder from Task 1, step 10
2. Submit the the answers to the four problems in Task 5 in a single text file called `answers.txt`
3. Submit the Pig scripts containing Pig code you used to get your answers stored in files called:
    - `P1.pig`
    - `P2.pig`
    - `P3.pig`
    - `P4.pig`

When you are finished, create a zip file containing all of the above folders/files and upload the resulting `a4.zip` file to the **Activity 4 dropbox**.