Lab: Exploring Data with Pig

About this Lab

Objective:	Use Pig to navigate through HDFS and explore a dataset.
File locations:	whitehouse/visits.txt in HDFS
Successful outcome:	You will have written several Pig scripts that analyze and query the White House visitors' data, including a list of people who visited the President.
Before you begin:	At a minimum, complete steps 1 and 2 of the Getting Started with Pig lab.
Related lesson:	***Introduction to Pig***

Lab Steps

1) Load the White House Visitor Data

If not already done, open a Terminal window in git bash or putty. You will use the TextLoader to load the visits.txt file. From the Pig Grunt shell, define the following `LOAD` relation:

```
# pig

grunt> A = LOAD '/user/root/whitehouse/' USING TextLoader();
```

2) Count the Number of Lines

Define a new relation named `B` that is a group of all the records in `A`:

```
grunt> B = GROUP A ALL;
```

Use `DESCRIBE` to view the schema of `B`.

```
grunt> DESCRIBE B;
```

What is the datatype of the group field? ________________________

Where did this datatype come from? ___________________________

Answer **:** The group field is a chararray because it is just the string "all" and is a result of performing a `GROUP ALL`.

Why does the A field of B contain no schema? ______________________

Answer: The A field of B contains no schema because the A relation has no schema.

How many groups are in the relation B? ______________

Answer: The B relation can only contain one group because it a grouping of every single record. Note that the A field is a bag, and A will contain any number of tuples.

The A field of the B tuple is a bag of all of the records in visits.txt. Use the COUNT function on this bag to determine how many lines of text are in visits.txt:

```
grunt> A_count = FOREACH B GENERATE 'rowcount', COUNT(A);
```

Note
The 'rowcount' string in the FOREACH statement is simply to demonstrate that you can have constant values in a GENERATE clause. It is certainly not necessary; it just makes the output nicer to read.
Use DUMP on A_count to view the result. The output should look like:

```
grunt> DUMP A_count;

(rowcount,447598)
```

We can now conclude that there are 447,598 rows of text in visits.txt. 3

) Analyze the Data's Contents

We now know how many records are in the data, but we still do not have a clear picture of what the records look like. Let's start by looking at the fields of each record. Load the data using PigStorage(',') instead of TextLoader():

```
grunt> visits = LOAD '/user/root/whitehouse/'
USING PigStorage(',');
```

This will split up the fields by comma.

Use a FOREACH...GENERATE command to define a relation that is a projection of the first 10 fields of the visits relation.

```
grunt> firstten = FOREACH visits GENERATE $0..$9;
```

Use `LIMIT` to display only 50 records then `DUMP` the result.

The output should be 50 tuples that represent the first 10 fields of visits:

```
grunt> firstten_limit = LIMIT firstten

50;

grunt> DUMP firstten_limit;

(PARK,ANNE,C,U51510,0,VA,10/24/2010 14:53,B0402,,)
(PARK,RYAN,C,U51510,0,VA,10/24/2010 14:53,B0402,,)
(PARK,MAGGIE,E,U51510,0,VA,10/24/2010 14:53,B0402,,)
(PARK,SIDNEY,R,U51510,0,VA,10/24/2010 14:53,B0402,,)
(RYAN,MARGUERITE,,U82926,0,VA,2/13/2011 17:14,B0402,,)
(WILE,DAVID,J,U44328,,VA,,,,)
(YANG,EILENE,D,U82921,,VA,,,,)
(ADAMS,SCHUYLER,N,U51772,,VA,,,,)
(ADAMS,CHRISTINE,M,U51772,,VA,,,,)
(BERRY,STACEY,,U49494,79029,VA,10/15/2010 12:24,D0101,10/15/2010
14:06,D1S)
```

Note

Because `LIMIT` uses an arbitrary sample of the data, your output will be different names but the format should look similar.

Notice from the output that the first three fields are the person's name. The next seven fields are a unique ID, badge number, access type, time of arrival, post of arrival, time of departure, and post of departure.

4) Locate the POTUS (President of the United States of America)

There are 26 fields in each record, and one of them represents the visitee (the person being visited in the White House). Your goal now is to locate this column and determine who has visited the President of the United States. Define a relation that is a projection of the last seven fields ($19 to $25) of visits. Use LIMIT to only output 500 records. The output should look like:

```
grunt> lastfields = FOREACH visits GENERATE $19..$25;
grunt> lastfields_limit = LIMIT lastfields 500;
grunt> DUMP lastfields_limit;

(OFFICE,VISITORS,WH,RESIDENCE,OFFICE,VISITORS,HOLIDAY OPEN
HOUSE/) (OFFICE,VISITORS,WH,RESIDENCE,OFFICE,VISITORS,HOLIDAY
OPEN HOUSES/)
(OFFICE,VISITORS,WH,RESIDENCE,OFFICE,VISITORS,HOLIDAY OPEN HOUSE/)
(CARNEY,FRANCIS,WH,WW,ALAM,SYED,WW TOUR)
(CARNEY,FRANCIS,WH,WW,ALAM,SYED,WW TOUR)
(CARNEY,FRANCIS,WH,WW,ALAM,SYED,WW TOUR)
(CHANDLER,DANIEL,NEOB,6104,AGCAOILI,KARL,)
```

It is not necessarily obvious from the output, but field $19 in the visits relation represents the visitee. Even though you selected 500 records in

the previous step, you may or may not see POTUS in the output above. (The White House has thousands of visitors each day, but only a few meet the President.)
Use `FILTER` to define a relation that only contains records of visits where field $19 matches POTUS. Limit the output to 500 records.

The output should include only visitors who met with the President. For example:

```
grunt> potus = FILTER visits BY $19 MATCHES
'POTUS';
grunt> potus_limit = LIMIT potus 500;
grunt> DUMP potus_limit;
(ARGOW,KEITH,A,U83268,,VA,,,,,2/14/2011 18:42,2/16/2011
16:00,2/16/2011 23:59,,154,LC,WIN,2/14/2011
18:42,LC,POTUS,,WH,EAST ROOM,THOMPSON,MARGRETTE,,AMERICA'S GREAT
OUTDOORS ROLLOUT EVENT
,5/27/2011,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,)
(AYERS,JOHNATHAN,T,U84307,,VA,,,,,2/18/2011 19:11,2/25/2011
17:00,2/25/2011 23:59,,619,SL,WIN,2/18/2011
19:11,SL,POTUS,,WH,STATE FLOO,GALLAGHER,CLARE,,RECEPTION
,5/27/2011,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,)
```

5) Count the POTUS Visitors

a. Let’s discover how many people have visited the President. To do this, we need to count the number of records in visits where field $19 matches `POTUS`. See if you can write a Pig script to accomplish this. Use the `potus` relation from the previous step as a starting point. You will need to use `GROUP ALL` and then a `FOREACH` projection that uses the `COUNT` function.

If successful, you should get 21,819 as the number of visitors to the White House who visited the President.

Solution:

```
grunt> potus = FILTER visits BY $19 MATCHES
'POTUS';
grunt> potus_group = GROUP potus ALL;
grunt> potus_count = FOREACH potus_group GENERATE COUNT(potus);
grunt> DUMP potus_count;
```

6) Finding People Who Visited the President

So far you have used `DUMP` to view the results of your Pig scripts. In this step, you will save the output to a file using the `STORE` command.
Now `FILTER` the relation by visitors who met with the President:

```
grunt> potus = FILTER visits BY $19 MATCHES 'POTUS';
```

Define a projection of the `potus` relationship that contains the name and time of arrival of the visitor:

```
grunt> potus_details = FOREACH potus GENERATE
(chararray) $0 AS lname:chararray,
(chararray) $1 AS fname:chararray,
(chararray) $6 AS arrival_time:chararray,
(chararray) $19 AS visitee:chararray;
```

d. Order the `potus_details` projection by last name:

```
grunt> potus_details_ordered = ORDER potus_details BY lname ASC;
```

Store the records of `potus_details_ordered` into a folder named `potus` and using a comma delimiter:

```
grunt> STORE potus_details_ordered INTO 'potus' USING
PigStorage(',');
```

f. View the contents of the `potus` folder:

```
grunt> ls potus
hdfs://sandbox.hortonworks.com:8020/user/root/potus/_SUCCESS<r 1>0
hdfs://sandbox.hortonworks.com:8020/user/root/potus/ part-v003-o000-r-
00000 <r
1>501378
```

Notice that there is a single output file, so the Pig job was executed with one reducer. View the contents of the output file using `cat`:

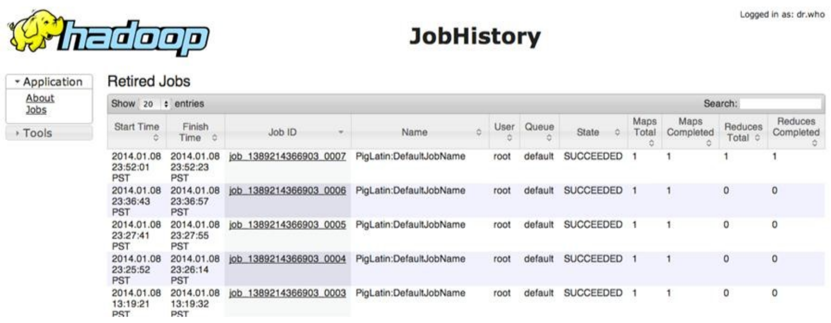
```
grunt> cat potus/part-r-00000
```

The output should be in a comma-delimited format and should contain the last name, first name, time of arrival (if available), and the string `POTUS`:

```
CLINTON,WILLIAM,,POTUS
CLINTON,HILLARY,,POTUS
CLINTON,HILLARY,,POTUS
CLINTON,HILLARY,,POTUS
CLONAN,JEANETTE,,POTUS
CLOOBECK,STEPHEN,,POTUS
CLOOBECK,CHANTAL,,POTUS
CLOOBECK,STEPHEN,,POTUS
CLOONEY,GEORGE,10/12/2010 14:47,POTUS
```

7) View the Pig Log Files

Each time you executed a `DUMP` or `STORE` command, a MapReduce job is executed on your cluster.
You can view the log files of these jobs in the JobHistory UI. Point your browser to `http://<sandbox-ip>:8088/`:



Retired Jobs

Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2014.01.08 23:52:01 PST	2014.01.08 23:52:23 PST	job_1389214366903_0007	PigLatin:DefaultJobName	root	default	SUCCEEDED	1	1	1	1
2014.01.08 23:36:43 PST	2014.01.08 23:36:57 PST	job_1389214366903_0006	PigLatin:DefaultJobName	root	default	SUCCEEDED	1	1	0	0
2014.01.08 23:27:41 PST	2014.01.08 23:27:55 PST	job_1389214366903_0005	PigLatin:DefaultJobName	root	default	SUCCEEDED	1	1	0	0
2014.01.08 23:25:52 PST	2014.01.08 23:26:14 PST	job_1389214366903_0004	PigLatin:DefaultJobName	root	default	SUCCEEDED	1	1	0	0
2014.01.08 13:19:21 PST	2014.01.08 13:19:32 PST	job_1389214366903_0003	PigLatin:DefaultJobName	root	default	SUCCEEDED	1	1	0	0

b. Click on the job's ID to view the details of the job and its log files.

Result

You have written several Pig scripts to analyze and query the data in the White House visitors' log. You should now be comfortable with writing Pig scripts with the Grunt shell and using common Pig commands like `LOAD`, `GROUP`, `FOREACH`, `FILTER`, `LIMIT`, `DUMP`, and `STORE`.