Pig Examples & Lab

Word Counting:

```
lines = LOAD '/user/maria_dev/g.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

Drivers & timesheets:

```
drivers = LOAD 'drivers.csv' USING PigStorage(',');
raw_drivers = FILTER drivers BY $0>1;
drivers_details = FOREACH raw_drivers GENERATE $0 AS driverId, $1 AS name;
timesheet = LOAD 'timesheet.csv' USING PigStorage(',');
raw_timesheet = FILTER timesheet by $0>1;
timesheet_logged = FOREACH raw_timesheet GENERATE $0 AS driverId, $2 AS hours_logged,
$3 AS miles_logged;
grp_logged = GROUP timesheet_logged by driverId;
sum_logged = FOREACH grp_logged GENERATE group as driverId,
SUM(timesheet_logged.hours_logged) as sum_hourslogged,
SUM(timesheet_logged.miles_logged) as sum_mileslogged;
join_sum_logged = JOIN sum_logged by driverId, drivers_details by driverId;
join_data = FOREACH join_sum_logged GENERATE $0 as driverId, $4 as name, $1 as
hours_logged, $2 as miles_logged;
dump join_data;
```

Drivers & Timesheets Lab

Start by transferring the files we will be working with to the linux filesystem of our vm

```
[maria_dev@sandbox-hdp ~] $ wget https://cse.sc.edu/~rose/587/CSV/drivers.csv
[maria_dev@sandbox-hdp ~] $ wget https://cse.sc.edu/~rose/587/CSV/timesheet.csv
```

Next transfer the files to HDFS:

```
[maria_dev@sandbox ~]$ hadoop fs -put drivers.csv /user/maria_dev
[mara_dev@sandbox ~]$ hadoop fs -put time_sheet.csv /user/maria_dev
```

Invoke grunt the pig interactive environment:

```
[maria_dev@sandbox-hdp ~]$ pig
18/04/12 11:44:51 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
18/04/12 11:44:51 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
18/04/12 11:44:51 INFO pig.ExecTypeProvider: Trying ExecType : TEZ_LOCAL
18/04/12 11:44:51 INFO pig.ExecTypeProvider: Trying ExecType : TEZ
18/04/12 11:44:51 INFO pig.ExecTypeProvider: Picked TEZ as the ExecType
2018-04-12 11:44:51,825 [main] INFO  org.apache.pig.Main - Apache Pig version 0.16.0.2.6.3.0-235
(rexported) compiled Oct 30 2017, 02:55:15
```

2018-04-12 11:44:51,825 [main] INFO org.apache.pig.Main - Logging error messages to: /home/maria_dev/pig_1523533491824.log
2018-04-12 11:44:51,846 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/maria_dev/.pigbootup not found
2018-04-12 11:44:52,376 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://sandbox-hdp.horto
nworks.com:8020
2018-04-12 11:44:53,001 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-caa28e7b-7ba7-4aa8-94da-3e6d4519f2d1
2018-04-12 11:44:53,373 [main] INFO org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://sandbox-hdp.hortonworks.com:8188/
ws/v1/timeline/
2018-04-12 11:44:53,483 [main] INFO org.apache.pig.backend.hadoop.PigATSClient - Created ATS Hook
grunt>

Determine the current working directory:

```
grunt> pwd
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev
```

Verify that the files we will be working with are in the current working directory:

```
grunt> ls
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/.Trash   <dir>
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/.staging <dir>
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/g.txt<r 1>       1746
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/nfldata  <dir>
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/output   <dir>
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/outputwc <dir>
hdfs://sandbox-hdp.hortonworks.com:8020/user/maria_dev/test1    <dir>
grunt>
```

Ok. We are ready to start work! Load the first file into “drivers”:

```
grunt> drivers = LOAD 'drivers.csv' USING PigStorage(',');
```

Let’s take a peek at this data set. The following command will take a while, but will eventually dump “`drivers`” to the console.

```
grunt> dump drivers
```

Let’s add the next line of the code. This has the effect of removing the first line which has column heading information. (All lines except for the first line start with a integer representing a driver ID.)

```
grunt> raw_drivers = FILTER drivers BY $0>1;
```

If you like, you could also dump `raw_drivers` to the console to verify that the first line has been filtered out.

```
grunt> dump raw_drivers
```

Next we create drivers_details, a new object consisting of the first two columns from raw_drivers

```
grunt> drivers_details = FOREACH raw_drivers GENERATE $0 AS driverId, $1 AS name;
```

Let's look at the resulting table; column one is the driver ID and column two is the driver's name:

```
grunt> dump drivers_details
```

Now, load the second data set, the timesheets:

```
grunt> timesheet = LOAD 'timesheet.csv' USING PigStorage(',');
```

Let's take a peek at this data set. The following command will take a while, but will eventually dump "`drivers`" to the console.

```
grunt> dump timesheet
```

As with the first dataset, we need to filter out the first row which contains column headings:

```
grunt> raw_timesheet = FILTER timesheet by $0>1;
```

We now select a subset of columns from `raw_timesheet` to create `timesheet_logged`:
Create column 1 from the first column of `raw_timesheets` and label it `driverId`
Create column 2 from the third column of `raw_timesheets` and label it `hours_logged`
Create column 3 from the second column of `raw_timesheets` and label it `miles_logged`

```
grunt> timesheet_logged = FOREACH raw_timesheet GENERATE $0 AS driverId, $2 AS
hours_logged, $3 AS miles_logged;
```

Let's take a peek at timesheet_logged. We should have a row for every week of every drivers timesheet data:

```
grunt> dump timesheet_logged
```

Create a new structure from `timesheet_logged` in which we group all of the rows for a given `driverId` as a single tuple. We will end up with one tuple per `driverId`.

```
grunt> grp_logged = GROUP timesheet_logged by driverId;
```

Let's take a peek at `grp_logged` to verify our expectation:

```
grunt> dump grp_logged
```

Now we are ready to sum up the hours and miles in each tuple to create `sum_logged` which has as column 1 the `driverId`, column 2 the sum of hours logged, and column 3 the sum of miles logged.

```
sum_logged = FOREACH grp_logged GENERATE group as driverId,
SUM(timesheet_logged.hours_logged) as sum_hourslogged,
SUM(timesheet_logged.miles_logged) as sum_mileslogged;
```

Let's take a peek at `sum_logged` to verify our expectation:

```
grunt> dump sum_logged
```

Now we create a new table by joining the `sum_logged` and `drivers_details` tables by matching rows that have the same `driverId`.

```
grunt> join_sum_logged = JOIN sum_logged by driverId, drivers_details by driverId;
```

Let's take a peek at `join_sum_logged` to verify our expectation:

```
grunt> dump join_sum_logged
```

Finally, create a new table called `join_data`, by selecting columns 1, 5, 2, and 3 from the table `join_sum_logged`.

```
grunt> join_data = FOREACH join_sum_logged GENERATE $0 as driverId, $4 as name, $1 as
hours_logged, $2 as miles_logged;
```

Finally, we have the overall solution, a table with a row for each driver listing the driver Id, driver name, total hours, and total miles. Let's take a peek at `join_data` to verify our expectation:

```
grunt> dump join_data
```