Excercise 1

**Q1: Load the file and print the results.**

To load the file, we can use the "LOAD" operator: *"LOAD 'data' [USING function] [AS schema];"*

To print out the contents, we can use the "DUMP" diagnostic operator: "*DUMP alias;*"

I've only put three columns in the schema below, but it can be extended as needed for however many columns there are in the imaginary dataset.

data = LOAD 'data' USING PigStorage(',')

AS (date:datetime, name:chararray, rated\_points:int);

DUMP data;

'data' is the path to the file. If the script is in a file called "e1q1" in the current directory, and the data is in the current directory, then one can run the script using pig in "local" mode:

pig -x local e1q1

This produces a lot of useless information on the error output, even if everything is correct. That can be ignored using a redirection:

pig -x local e1q1 2> /dev/null

**Q2: Group the list of movies by ratings.**

Assuming we have the data loaded in from Q1:

grouped\_movies = GROUP data BY rated\_points;

**Q3: Group the list of movies by ratings.**

"DESCRIBE" prints the schema of the given relation. If "DUMP" is replaced with "DESCRIBE" in the script in Q1, we get the following output:



**Q4: Process clickstream data into user sessions.**

The dataset described in Exercise 1 is already processed into user sessions!

Each record has a login\_time and a logout\_time.

**Q5: Use a FOREACH statement.**

There are any number of uses. We could, for example, retrieve the usernames in the dataset.

grouped\_movies = FOREACH data GENERATE user\_name;

**Q6: Get top ratings in rach group.**

Assuming we have some group column to group by - I'll call it "groupID" for now - the following foreach with nested block will do the job:

groups = GROUP data BY groupID;

highest\_ratings = FOREACH groups {

ordered = ORDER data BY rating DESC;

highest = LIMIT ordered 1;

GENERATE FLATTEN(highest);

};

**Q7: Select clicks corresponding to starting, browsing, completing, or purchasing.**

Each row in the dataset above correspond to a *preference*, not a click.

However, imagining that there is a column "action", we could do something like:

selected = FILTER data BY (action == 'start') OR (action == 'browse') OR (action == 'complete') OR (action == 'purchase');

Alternatively, a tutor in the practical suggested that what is actually asked for here are columns, not rows. In that case, we could use a foreach statement:

selected = FOREACH data GENERATE (login\_time, rated\_movie, completed\_movie, purchased\_movie)

Excercise 2

**Q1: Describe the student data.**

This is the schema I've chosen in Pig Latin schema:

(student\_id:int, first\_name:chararray, last\_name:chararray, age:int);

**Q2: Assign names to the columns in the data.**

Write out a data file with headers seems to be something that Pig is not meant to do by default. We can send the information to console.

data = LOAD 'students.csv' USING PigStorage(',')

AS (student\_id:int, first\_name:chararray, last\_name:chararray, age:int);

DUMP data

And then use a redirection to write the data to a file:

pig -x local e1q1 2> /dev/null 1> student\_details.csv

But obviously that isn't scalable, and in the local case one might as well use the terminal to perform this task.

**Q3: Perform 3 operations.**

These 3 operations find a set of unique first names among the students who are 24 or older.

a = FILTER data BY age > 23

b = FOREACH a GENERATE first\_name

result = DISTINCT b

**Q4: Sum student hours.**

Assuming that there is a field "hours" in the attendance.csv dataset, and that attendance.csv has been loaded same as for the other datasets:

student = GROUP attendance\_data BY student\_id

student\_hours = FOREACH students GENERATE group, SUM(data.hours)

**Q5: Join the datasets.**

Assuming the two csvs are loaded as student\_data and attendance\_data, a join would look like:

joined = JOIN student\_data BY student\_id, attendance\_data BY student\_id

**Q6: Print the result.**

We can print the result using DUMP, same as before:

DUMP joined;

DUMP student\_hours;

-- etc