Instructions

For each answer, please include your answer as text, and any screenshot(s) which demonstrate your answer was executed. Most importantly, make sure to include evidence your answer is correct. This will most likely be a screenshot. If you had issues, problems, or had to make assumptions include them in your answer.

Your Answers:

1. Load the comma-delimited HDFS dataset at **clickstream/iplookup** into a relation with an explicit schema. Use filter logic to remove the first row (which contains a header) then sort the output by IP and dump a comma-delimited data set to **clickstream/iplookup_noheader.** Record all of your Pig commands required to complete your transformation.

   Load the iplookup file into a relation
   >grunt clickstream = LOAD '/user/cloudera/clickstream/iplookup/ip_lookup.csv' USING PigStorage(',') as (IP: chararray, Country: chararray, State: chararray, City: chararray, ApproxLat: float, ApproxLng float);

   Create a new relation with IP sorted
   > grunt clickstream_sorted = ORDER clickstream BY IP ASC;

   Store the comma-delimited data in a new folder
   >grunt STORE clickstream_sorted INTO '/user/cloudera/clickstream/iplookup_noheader' USING PigStorage(',');

   Check that the file got stored there
   >grunt fs -ls clickstream/iplookup_noheader;

   Cat out the data to show that its there
   >grunt fs -cat clickstream/iplookup_noheader/part-r-00000;

```
grunt> fs -cat clickstream/iplookup_noheader/part-r-00000
128.122.140.238,USA,NY,New York,40.712784,-74.00594
128.230.122.180,USA,NY,Syracuse,43.048122,-76.14742
155.100.169.152,USA,UT,Salt Lake City,40.76078,-111.891045
172.189.252.8,USA,VA,Dulles,38.955856,-77.44782
215.82.23.2,USA,OH,Columbus,39.961178,-82.998795
38.68.15.223,USA,TX,Dallas,32.776665,-96.79699
54.114.107.209,USA,NJ,Jersey City,40.728157,-74.077644
56.216.127.219,USA,NC,Raleigh,35.77959,-78.638176
68.199.40.156,USA,NY,Freeport,40.6576,-73.58318
70.209.14.54,USA,FL,Tampa,27.950575,-82.45718
74.111.18.59,USA,NY,Syracuse,43.048122,-76.14742
74.111.6.173,USA,VA,Arlington,38.87997,-77.10677
8.37.70.112,USA,CA,Los Angeles,34.052235,-118.24368
8.37.70.170,USA,CA,Los Angeles,34.052235,-118.24368
8.37.70.226,USA,CA,Los Angeles,34.052235,-118.24368
8.37.70.77,USA,CA,Los Angeles,34.052235,-118.24368
8.37.70.99,USA,CA,Los Angeles,34.052235,-118.24368
8.37.71.25,USA,CA,Los Angeles,34.052235,-118.24368
8.37.71.43,USA,CA,Los Angeles,34.052235,-118.24368
8.37.71.57,USA,CA,Los Angeles,34.052235,-118.24368
8.37.71.69,USA,CA,Los Angeles,34.052235,-118.24368
8.37.71.9,USA,CA,Los Angeles,34.052235,-118.24368
98.29.25.44,USA,OH,Cleveland,41.49932,-81.69436
IP,Country,State,City,,
grunt>
```

2. Write Pig commands to produce a count of IP Addresses by state codes, sorted by the count with highest values first, like this:
   (CA, 10)
   (NY, 4)
   (VA, 2)
   Etc…
   Record all your Pig commands required to complete your transformation.

   Create an initial relation to group the states
   grunt> GROUP clickstream BY State;

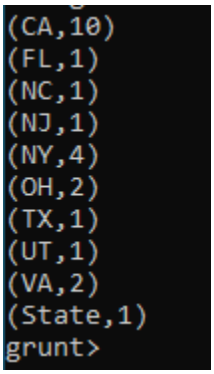
   Rename so that the column is not called group
   grunt> state_rename = FOREACH states GENERATE group as state, clickstream;

   Show the state_rename relation
   grunt> describe state_rename

```
grunt> describe state_rename
state_rename: {state: chararray,clickstream: {(IP: chararray,Country: char
array,State: chararray,City: chararray,ApproxLat: float,ApproxLng: float)}
```

   Create the relation with the counts by state
   grunt> state_count = FOREACH state_rename GENERATE state, COUNT(clickstream.State) as count;

Run the map reduce job and dump the relation
grunt> dump state_count



3. Use pig to load the web log files from **clickstream/logs** using the following schema: reqdate:chararray, reqtime:chararray, x1:int, method:chararray, uri:chararray, x2:int ,x3:int, x4:int ,ipaddress:chararray, useragent:chararray, filter any rows which begin with a "#" (these are header rows and should be removed, then writes out the reqdate, reqtime, method, uri, ipaddress and useragent columns to a tab-delimited data set in HDFS **clickstream/logs_noheader**. HINT: The data is space delimited.

   Take a look at were working with in one of the log files
   grunt> fs -cat clickstream/logs/u_ex160211.log

   Load the web logs into a relation
   grunt> weblogs = LOAD '/user/cloudera/clickstream/logs/*.log' USING PigStorage(' ') as (reqdate: chararray, reqtime: chararray, x1: int, method: chararray, uri: chararray, x2: int, x4: int, ipaddress: chararray, useragent: chararray);

   check out the relation schema
   >grunt describe weblogs

   Couldn't figure out how to get rid of #, tried (plus various other similar attempts switching the syntax around a little bit but couldn't get it to work)
   >grunt weblogs_filtered = FILTER weblogs by reqdate not like '#%'

   Create a new relation with the selected columns
   grunt> weblogs_new = FOREACH weblogs GENERATE reqdate, reqtime, method, uri, ipaddress, useragent;

   store the relation into a tab delimited folder in HDFS
   grunt> store weblogs_new into '/user/cloudera/clickstream/logs_noheader' USING PigStorage('\t');

Make sure that the file is there
grunt> fs -ls clickstream/logs_noheader

```
grunt> fs -ls clickstream/logs_noheader
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-08-17 03:58 clickstream/logs_noheader/_SUCCESS
-rw-r--r--   1 cloudera cloudera     100135 2021-08-17 03:58 clickstream/logs_noheader/part-m-00000
grunt>
```

4. Use hive to create two external tables for the **clickstream/logs_noheader** and **clickstream/iplookup_noheader** files you created in the previous steps. These tables should be named **weblogs** and **iplookup** respectively and should be placed in the **clickstream** database. Be sure to record all HQL steps to complete the operations.

   connect to the hive client
   beeline -u jdbc:hive2://localhost:10000/default -n cloudera -p cloudera --silent=true
   not sure why but it couldn't seem to connect

```
0: jdbc:hive2://localhost:10000/default (closed)> show databases;
No current connection
0: jdbc:hive2://localhost:10000/default (closed)>
```

   show a list of databases in the metastore
   > show databases;

   * I am not sure why it wouldn't let me connect but here are the commands that I would have ran if it did

   create the first external table
   > create external table weblogs (reqdate string, reqtime string, method string, uri string, ipaddress string, useragent string) row format delimited fields terminated by '\t' location '/user/cloudera/clickstream';

   create the second external table
   > create external table iplookup (state string, count int) row format delimited fields terminated by '\t' location '/user/cloudera/clickstream';

5. Write an HQL query to display the name of the city and the number of HTTP requests from that city (NOTE: each row in the web logs is an HTTP request). Order the output so cities with the most requests are at the top. If you complete the query correctly, you should see Syracuse has 272-page requests and Los Angeles has 24.

* Again could not connect to hive but here are the commands that I would run

Run the query
> select state, count(state) as count from(
(select * from weblogs) Q1 join on (select * from iplookup Q2)
) sub where Q1.IP = Q2.ipaddress
group by state