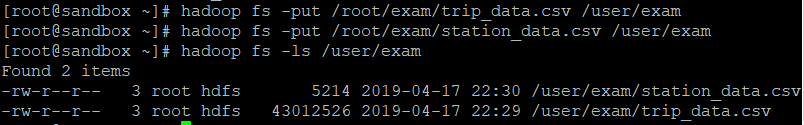
CIND 719 Final Exam Chantal Sylvestre

**First the data is loaded into Hadoop:**

hadoop fs -mkdir /user/exam

hadoop fs -put /root/exam/trip\_data.csv /user/exam

hadoop fs -put /root/exam/station\_data.csv /user/exam

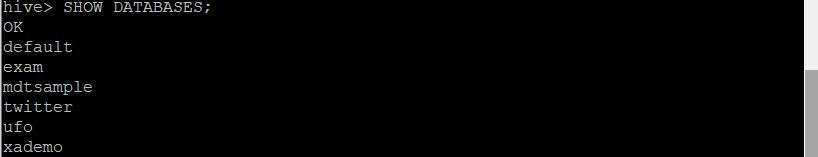


**QUESTION 1**

**Then a table for the bike trip data is created and the data is loaded into hive:**

CREATE DATABASE exam;

SHOW DATABASES;



USE exam;

**--Create the tables**

CREATE TABLE bike.trip(

trip\_id int,

duration int,

start\_date string,

start\_station string,

start\_terminal string,

end\_date string,

end\_station string,

end\_terminal string,

bike\_num string,

sub\_type string,

zip int)

row format delimited

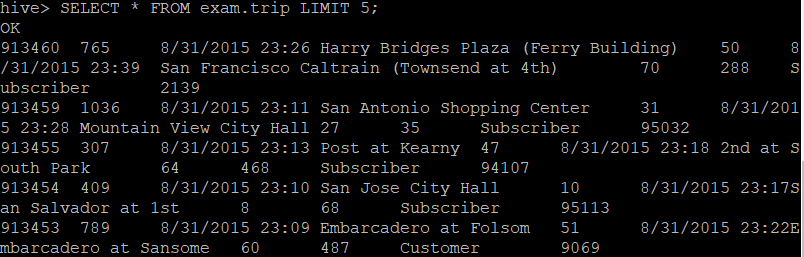
fields terminated by ',' ;

**--Load data into tables**

LOAD DATA INPATH '/user/lab/trip\_data.csv'

OVERWRITE INTO TABLE exam.trip;

SELECT \* FROM exam.trip LMIT 5;



The data is successfully loaded into Hive, now a table to represent a directed weighted graph (network) that shows which stations are connected will be created. The trip count between those stations is used as the weights. I’m going to create an edge list, like below:

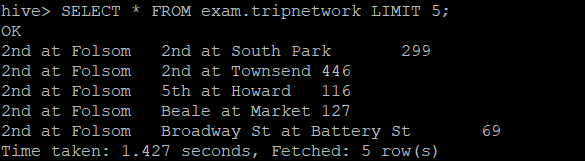
|  |  |  |
| --- | --- | --- |
| Start Station | End Station | # of Trips |
|  |  |  |
|  |  |  |

Since the bike trips are only one way and there are no loops this is adequate to explain the relationship between the points. This way it we can answer the 2nd question about the traffic index easier since we will have a number of trips which can be divided by the mean and in the final question we can look at the start and end station in order to see the in degree and out degrees. One thing I will do is remove the trips that start and end at the same station, since this would be a node and not an edge.

**We want the number of trips between stations, so we group by station and count the number of trips where the start and end stations are not the same:**

CREATE TABLE exam.tripnetwork AS SELECT start\_station, end\_station, COUNT(trip\_id) AS cnt FROM exam.trip WHERE start\_station <> end\_station GROUP BY start\_station, end\_station;

**This is the output:**



**Now I will save to external table in ‘/user/lab/q1’**

**Create the external table first:**

CREATE EXTERNAL TABLE exam.tripnetwork1(

Start\_station string

End\_station string

Trips int)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

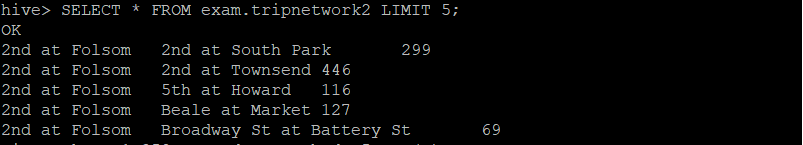
STORED AS TEXTFILE

LOCATION ‘/user/lab/q1’;

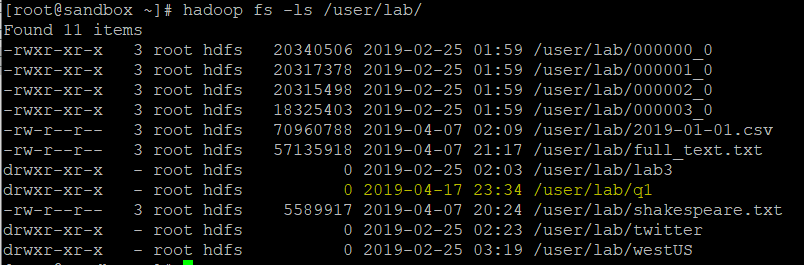
**Then load in data:**

INSERT INTO TABLE exam.tripnetwork1 SELECT \* FROM exam.tripnetwork;

**First 5 lines:**



**Directory output:**



**QUESTION 2**

**Load data into pig, then apply the index formula to all inputs. I had found the mean in an earlier calculation in Hive that I didn’t screen shot using SELECT AVG(trips) FROM exam.tripnetwork2; and that ouput is 212, so I used that in my calculations for the mean of all routes:**

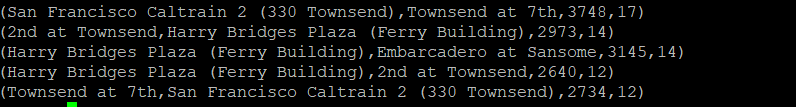
a = LOAD ‘/user/lab/q1/000000\_0’ using PigStorage(‘,’) AS (t\_start:chararray, t\_end:chararray, t\_trips:int);

b = foreach a generate t\_start, t\_end, t\_trips, t\_trips/212 AS index;

c = order b by index desc;

d = limit c 5;

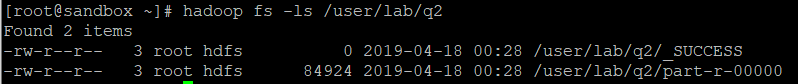
DUMP d



**Now I will save the results on HDFS:**

store c into '/user/lab/q2';

Directory on HDFS:

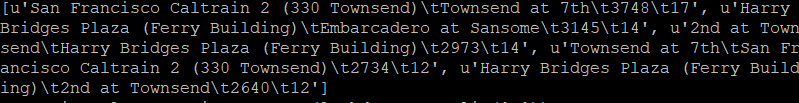


**QUESTION 3**

**Loading data into Pyspark:**

trips\_raw = sc.textFile(“/user/lab/q2/part-r-00000”)

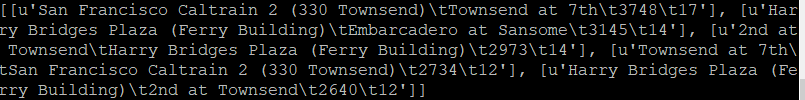
trips\_raw.take(5)



**Separating by Comma:**

trips\_clean = trips\_raw.map(lambda x :x.split(‘,’))

trips\_clean.take(5)

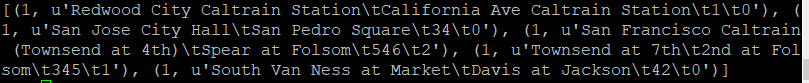


**Here I try to reduce by Key so that I can sort by key after:**

trips\_key = trips\_clean.map(lambda x: (x[0], 1)).reduceByKey(lambda a,b: a+b)

**Here I try to sort by key but I don’t think it worked :**

trips\_key.map(lambda x:(x[1], x[0])).sortByKey(1).take(3)



I think what I should have done is used the station table csv file to group all the trips relating to the start station together, so regardless of the end station (similar to what we did in the assignment with the timeblocks). That way I could see the highest number of out degrees by the start station, and then do the same with the end station and then get the highest number of in degrees.

If my code worked properly, I would sort by key and then take the top 3 results. The start station would show the top three stations with the highest out degrees, and the top three end stations would show the highest out degrees.

The highest in degree would represent the stations with the most incoming traffic connections. This would show where most people are looking to end up from many different connections. So for example, I would assume perhaps a station near a work place would have high incoming traffic from many different connections since people are commuting from many locations depending on where they live.

The highest out degree would represent the stations with the most outgoing traffic connections. So these are stations with a lot of people leaving that area to many different locations, so perhaps this would be leaving a central work location outwards to many different places.

If anything, the highest in degrees and out degrees show very important stations, they could be important for connecting to other stations, and they are used very frequently since they have a high number of routes.



Thank you! ☺

I would suggest if you base most of the final off graphs perhaps go through an example first with the class, like how to make an adjacency list in spark. It was really challenging for me to sort that out, on top of using 3 different languages (hive pig and spark). It was a bit too much. Thanks for considering my feedback.