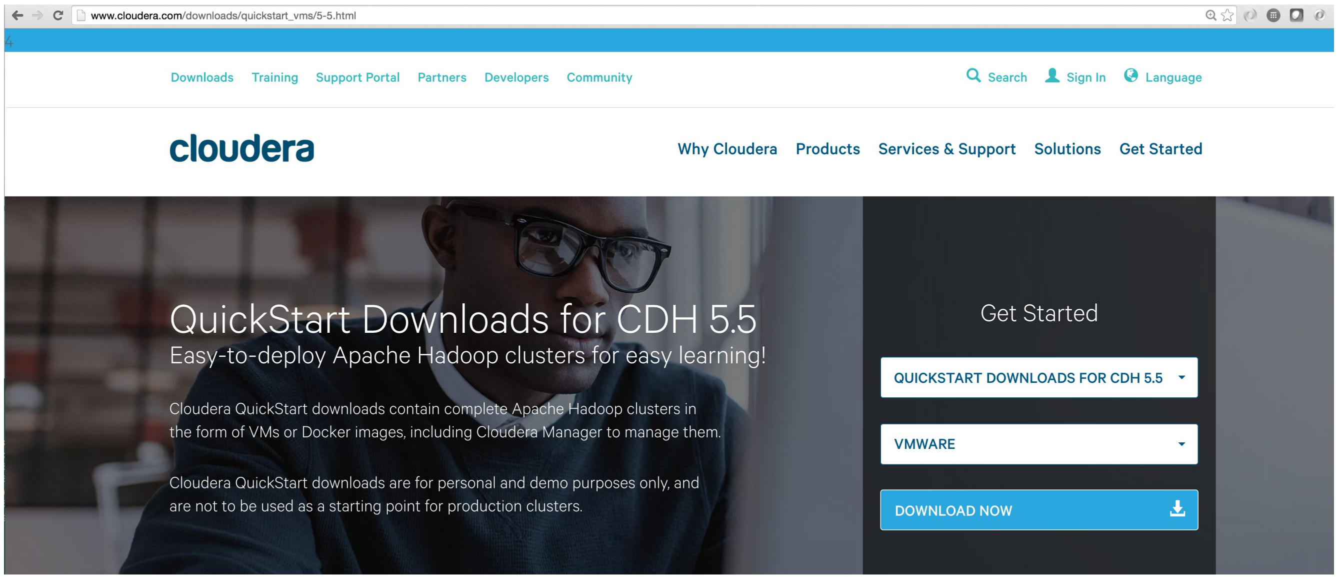
CS9223 Big Data - Midterm
Jiann-Liang Tsay (jlt245)

I use the platform downloaded from cloudera.com, the QuickStart VM of vmware version for this project.



QuickStart Downloads for CDH 5.5

Easy-to-deploy Apache Hadoop clusters for easy learning!

Cloudera QuickStart downloads contain complete Apache Hadoop clusters in the form of VMs or Docker images, including Cloudera Manager to manage them.

Cloudera QuickStart downloads are for personal and demo purposes only, and are not to be used as a starting point for production clusters.

Before I can use this installed QuickStart VM, I run a startup script to start a few services:

```
sudo service oozie start
sudo service hive-metastore start
sudo service hive-server2 start
sudo service impala-state-store start
sudo service impala-server start
sudo service hue start
sudo service impala-catalog start
sudo service hadoop-yarn-nodemanager start
sudo service hadoop-mapreduce-historyserver start
sudo service hadoop-yarn-resourcemanager start
sudo service spark-master start
sudo service spark-worker start
sudo service hbase-master start
sudo service hbase-regionserver start
```

I have started more services than I actually needed, for example, originally I plan to use HBase to store the dataset but end up to use HDFS only. I did not use HIVE since PIG seems good enough to work out the project.

After inspection the json data from yelp dataset, I decide to use Pig's Jython UDF to do data conversion and work on the converted data instead of the original dataset. This gives me flexibility to create the data schema I want in order to run Pig scripts.

1. Summarize the number of reviews by US city, by business category:

a. The Jython UDF python script converter.py has the method "business" to convert dataset into the “business.csv” dataset in hdfs.

```
def business(line):
    b_json = json.loads(line)
    biz = [b_json["business_id"],
           b_json["name"],
           b_json["city"],
           b_json["type"],
           str(b_json["review_count"]),
           str(b_json["stars"]),
           str(b_json["longitude"]),
           str(b_json["latitude"]) ]

    return ",".join(biz).encode('utf-8')
```

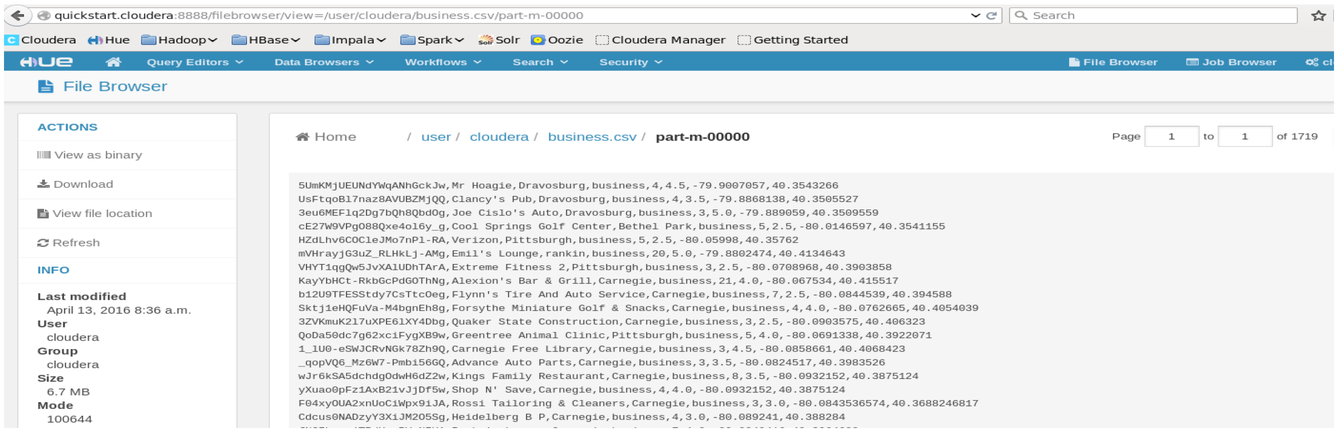
From python script, we can see that I only retrieve the fields that I need only from yelp_academic_dataset_business.json. Also I can reformat the data, this gives us to correct errors or incompatibility of the original dataset.

The new dataset creation in Pig is as below:

```
REGISTER '/home/cloudera/data/jyson-1.0.2/lib/jyson-1.0.2.jar'

-- business_id, city, review_count, type, stars, longitude, latitude
REGISTER '/home/cloudera/workspace/py_midterm/bin/converter.py' USING jython AS converter;
biz = LOAD '/user/cloudera/data/yelp_academic_dataset_business.json' AS (line:CHARARRAY);
csv = FOREACH biz GENERATE converter.business(line);
STORE csv INTO '/user/cloudera/business.csv';
```

You may inspect the created file in Hue's “File browser” pages as below:



Last modified
April 13, 2016 8:36 a.m.
User
cloudera
Group
cloudera
Size
6.7 MB
Mode
100644

```
5UmKMjUEUNdYWqANhGckJw,Mr Hoagie,Dravosburg,business,4,4.5,-79.9007057,40.3543266
UsFtqoBl7naz8AVUBZMjQQ,Clancy's Pub,Dravosburg,business,4,3.5,-79.8868138,40.3505527
3eu6MEFlq2Dg7bQh8QbdOg,Joe Cislo's Auto,Dravosburg,business,3,5.0,-79.889059,40.3509559
cE27W9VPgO88Qxe4ol6y_g,Cool Springs Golf Center,Bethel Park,business,5,2.5,-80.0146597,40.3541155
HZdLhv6COCleJMo7nPl-RA,Verizon,Pittsburgh,business,5,2.5,-80.05998,40.35762
mVHrayjG3uZ_RLHkLj-AMg,Emil's Lounge,rankin,business,20,5.0,-79.8802474,40.4134643
VHYT1qgQw5JvXAlUDhTArA,Extreme Fitness 2,Pittsburgh,business,3,2.5,-80.0708968,40.3903858
KayYbHCt-RkbGcPdGOThNg,Alexion's Bar & Grill,Carnegie,business,21,4.0,-80.067534,40.415517
b12U9TFESStdy7CsTtcOeg,Flynn's Tire And Auto Service,Carnegie,business,7,2.5,-80.0844539,40.394588
Sktj1eHQFuVa-M4bgnEh8g,Forsythe Miniature Golf & Snacks,Carnegie,business,4,4.0,-80.0762665,40.4054039
3ZVKmuK2l7uXPE6lXY4Dbg,Quaker State Construction,Carnegie,business,3,2.5,-80.0903575,40.406323
QoDa50dc7g62xciFygXB9w,Greentree Animal Clinic,Pittsburgh,business,5,4.0,-80.0691338,40.3922071
1_lU0-eSWJCRvNGk78Zh9Q,Carnegie Free Library,Carnegie,business,3,4.5,-80.0858661,40.4068423
_qopVQ6_Mz6W7-Pmbi56GQ,Advance Auto Parts,Carnegie,business,3,3.5,-80.0824517,40.3983526
wJr6kSA5dchdgOdwH6dZ2w,Kings Family Restaurant,Carnegie,business,8,3.5,-80.0932152,40.3875124
yXuao0pFz1AxB21vJjDf5w,Shop N' Save,Carnegie,business,4,4.0,-80.0932152,40.3875124
F04xyOUA2xnUoCiWpx9iJA,Rossi Tailoring & Cleaners,Carnegie,business,3,3.0,-80.0843536574,40.3688246817
Cdcus0NADzyY3XiJM2O5Sg,Heidelberg B P,Carnegie,business,4,3.0,-80.089241,40.388284
```

With this “business.csv” dataset, we can now run the below Pig script to get the answer for this question:

```
R = LOAD '/user/cloudera/business.csv' USING PigStorage(',') AS (business_id:chararray, name:chararray, city:chararray,
type:chararray, review_count:int, stars:float, longitude:float, latitude:float);

F = FILTER R BY type=='business';
```

```
-- generate only the fields we need.
G = FOREACH F GENERATE business_id, city, review_count;
C = GROUP G BY city;

X = FOREACH C GENERATE group, SUM(G.review_count);

DUMP X;
```

The results are as below (partial list):

```
(Dane,10)
(Etna,36)
(MMRP,16)
(Mesa,63897)
(Meza,11)
(Anjou,53)
(Clark,193)
(Eagan,4)
(Laval,1417)
(Leith,39)
(Mesa ,35)
(Ratho,17)
(Savoy,128)
(Tempe,95650)
….
```

<u>2. Rank the cities by # of stars for each category, by city.</u>

In this case, I used two customized dataset, one is from the above “business.csv”, the other is the category v.s. business_id, since for each business, it can be categorized into multiple categories.

The method of Jython UDF python script converter.py is called “categories” as below:

```
@outputSchema("categories:bag{t:(business_id:chararray,category:chararray)}")
def categories(line):
    b_json = json.loads(line)
    cats = []
    for x in b_json["categories"]:
        cats.append((b_json["business_id"], x))

    return cats
```

It converts the data into Pig's “bag schema” dataset.

The below pig script to run the ranking for this question is:

```
R = LOAD '/user/cloudera/business.csv' USING PigStorage(',') AS (business_id:chararray, city:chararray, type:chararray,
review_count:int, stars:float, longitude:float, latitude:float);
C = LOAD '/user/cloudera/categories.bag' USING PigStorage() AS b:{t:(business_id:chararray,category:chararray)};

-- Flattening the bag structure into list.
B = FOREACH C GENERATE FLATTEN(b);
```

```
X = JOIN R BY business_id, B BY business_id;

-- Only generate the fields we need for ranking.
Y = FOREACH X GENERATE category, city, stars;

G = GROUP Y by (category, city);

-- Round the average stars.
Z = FOREACH G GENERATE FLATTEN(group), ROUND(AVG(Y.stars)*10f)/10f AS stars;

K = RANK Z by stars DESC;

DUMP K;
```

Here R, C are representing two datasets and X is the join of them with the common key "business_id". Y is the dataset with the only fields we need in this ranking operation. Z is the calculation of stars before ranking. The final results are as below:

The fields are "rank", "category", "city", "stars(average)".

```
(1,Property Management,Gilbert,5.2)
(2,Fast Food,Bonnyrigg,5.0)
(2,Bowling,Fountain Hills,5.0)
(2,Fertility,Chandler,5.0)
(2,Gardeners,Glendale,5.0)
(2,Gardeners,Apache Junction,5.0)
(2,Hospitals,Peoria,5.0)
(2,Insurance,Anthem,5.0)
(2,Insurance,Carefree,5.0)
(2,Insurance,Florence,5.0)
(2,Insurance,Champaign,5.0)
(2,Insurance,Casa Grande,5.0)
(2,Insurance,Boulder City,5.0)
(2,Insurance,Apache Junction,5.0)
...
```

3. What is the average rank(stars) for businesses within 5 miles of CMU by type.

First of all I used this site: http://www.csgnetwork.com/degreelenllavcalc.html to calculate the degrees of latitude and longitude degrees for 5 mile ranges:

Lo5 = 5/52.72 = 0.0948
La5 = 5/69.00 = 0.0725

so the range of latitude is [40.3687,40.5137]
the range of longitude is [-80.8376,-79.848]

The pig script is as below:

```
R = LOAD '/user/cloudera/business.csv' USING PigStorage(',') AS (business_id:chararray, name:chararray, city:chararray,
type:chararray, review_count:int, stars:float, longitude:float, latitude:float);
```

```
F = FILTER R BY ((longitude>=-80.8376) AND (longitude<=-79.848) AND (latitude>=40.3687) AND
(latitude<=40.5137));

-- generate only the fields we need.
G = FOREACH F GENERATE business_id, type, stars;

C = GROUP G BY type;

X = FOREACH C GENERATE group, ROUND(10f*AVG(G.stars))/10f AS stars;

DUMP X;
```

I used the dataset “business.csv” that has been created for question 1. Filter the dataset by checking the range of longitude and latitude near CMU. Then GROUP BY “type” and calculate the average of stars. The final result is:

```
(business,3.7)
```

<u>4. Rank reviewers by number of reviews. Show the average number of starts by category for the top 10 reviewers.</u>

From the similar approach in the previous questions, I created new datasets “reviewers.csv” and “users.csv” from yelp's yelp_academic_dataset_review.json and yelp_academic_dataset_user.json. The UDF python scripts are:

```
def reviewers(line):
    r_json = json.loads(line)
    revs = [r_json["business_id"], r_json["user_id"], str(r_json["stars"]) ]

    return ",".join(revs).encode('utf-8')

def users(line):
    u_json = json.loads(line)
    users = [u_json["user_id"], u_json["name"], str(u_json["average_stars"]), str(u_json["review_count"]) ]

    return ",".join(users).encode('utf-8')
```

And the pig script for generating datasets “users.csv” and “reviewers.csv” are:

```
REGISTER '/home/cloudera/data/jyson-1.0.2/lib/jyson-1.0.2.jar'

REGISTER '/home/cloudera/workspace/py_midterm/bin/converter.py' USING jython AS converter;

revs = LOAD '/user/cloudera/data/yelp_academic_dataset_review.json' AS (r_line:CHARARRAY);
r_csv = FOREACH revs GENERATE converter.reviewers(r_line);

STORE r_csv INTO '/user/cloudera/reviewers.csv';

usrs = LOAD '/user/cloudera/data/yelp_academic_dataset_user.json' AS (u_line:CHARARRAY);
u_csv = FOREACH usrs GENERATE converter.users(u_line);

STORE u_csv INTO '/user/cloudera/users.csv';
```

Again, I just retrieved the fields I need for the project from the original yelp's datasets.

For ranking reviews by number of reviews, the Pig script is:

```
U = LOAD '/user/cloudera/users.csv' USING PigStorage(',') AS (user_id:chararray, name:chararray, average_stars:float,
review_count:int);

G = GROUP U by (user_id, name);

Z = FOREACH G GENERATE FLATTEN(group), SUM(U.review_count) AS reviews;

K = RANK Z by reviews DESC;

DUMP K;
```

And the final results are (rank, user_id, name, review_count):

```
(1,JLM36sYWmouJAZ2knzst7A,Victor,10320)
(2,1p3d__fuRRCDXfbS1Tq0wA,Shila,8529)
(3,3zBKfA8-_fJRagWSTMLVvg,Kenneth,5833)
(4,VhI6xyylcAxi0wOy2HOX3w,Bruce,5648)
(5,22-6yC05pgWbLupHZTjQig,Jennifer,5050)
(6,pz97SxRe1Vk-5_K6EB9OSA,Anita,3953)
(7,7FuLnS_-b79GG-33mwLaMg,Andrew,3934)
(8,EY2M507w8246POHv37TLyA,Neal,3750)
(9,sKtmrRvPMn5GRTnHYd41CA,Nijole,3653)
(10,utwvWcQYQrXBUSpzRR-21Q,Tina,3621)
(11,6HBnx7fTfFlpWyez_P55xA,Karen,3539)
(12,C3xoIPpa7O6pjX9rZOaeHA,Jason,3480)
(13,SjocZ3uwUAKl-vBZ0UYmuw,Ian,3473)
(14,JfHEExxD3iBXK5ZVWcc9SA,David,3336)
(15,OMB8Y_DOaB21vCXa-cruwg,Ed,3241)
(16,5lq4LkrviYgQ4LJNsBYHcA,Michael,3225)
(17,5JBz60Xz2dXoHvUbgtZkUQ,Stefan,3155)
(18,uI1HFHjEQrtR_vqMJTXmsQ,Greg,3146)
(19,pW91HUnVz6ssLZ4dY-ztyQ,Miriam,3069)
(20,pEVf8GRshP9HUkSpizc9LA,Jennifer,2977)
...
```

Let's get the average stars for the top 10 reviewers by category. The categories are from business_id. Therefore, I am going to do a couple joins.

```
R = LOAD '/user/cloudera/reviewers.csv' USING PigStorage(',') AS (business_id:chararray, user_id:chararray, stars:int);
U = LOAD '/user/cloudera/users.csv' USING PigStorage(',') AS (user_id:chararray, name:chararray, average_stars:float,
review_count:int);

G = GROUP U by (user_id, name);

Z = FOREACH G GENERATE FLATTEN(group), SUM(U.review_count) AS reviews;

K = RANK Z by reviews DESC;

-- DUMP K;

-- The below is for showing average stars of the top 10 reviewers.
```

```
Top10 = LIMIT K 10;

T = JOIN Top10 BY user_id, U BY user_id;
D = FOREACH T GENERATE U::user_id AS user_id, U::name AS name, U::average_stars AS average_stars;

X = JOIN R BY user_id, D BY user_id;

E = FOREACH X GENERATE business_id, D::user_id AS user_id, D::name AS name, D::average_stars AS
average_stars;

C = LOAD '/user/cloudera/categories.bag' USING PigStorage() AS b:{t:(business_id:chararray,category:chararray)};

B = FOREACH C GENERATE FLATTEN(b);

Y = JOIN B BY business_id, E BY business_id;

J = FOREACH Y GENERATE category, E::user_id AS user_id, E::name AS name, E::average_stars AS average_stars;

W = GROUP J BY (category, user_id, name);

A = FOREACH W GENERATE FLATTEN(group), FLATTEN(J.average_stars);

AA = DISTINCT A;

DUMP AA;
```

Here I used three dataset to do join and retrieve the fields we need:
X is the reviewers and users dataset join that we can have review data for users. Y is the categories and business join that give us a flat dataset with category v.s. Business. Unlike in the original yelp's business dataset, it includes a list of categories for each business, that is hard to do search by category operations. T is giving the top 10 users with their "average_stars" data field. The final dataset A and AA that we need to do "DISTINCT" because in the business category join, we generate "replications".

The final results are ("category", "user_id", "name", "average_stars"):

```
(Bars,22-6yC05pgWbLupHZTjQig,Jennifer,3.26)
(Bars,3zBKfA8-_fJRagWSTMLVvg,Kenneth,3.36)
(Bars,7FuLnS_-b79GG-33mwLaMg,Andrew,3.69)
(Bars,EY2M507w8246POHv37TLyA,Neal,3.92)
(Bars,VhI6xyylcAxi0wOy2HOX3w,Bruce,3.48)
(Bars,pz97SxRe1Vk-5_K6EB9OSA,Anita,3.43)
(Bars,sKtmrRvPMn5GRTnHYd41CA,Nijole,3.73)
(Bars,utwvWcQYQrXBUSpzRR-21Q,Tina,3.27)
(Food,1p3d__fuRRCDXfbS1Tq0wA,Shila,3.74)
(Food,22-6yC05pgWbLupHZTjQig,Jennifer,3.26)
...
```

5. For the top 10 and bottom 10 food business near CMU, summarize star rating by of month.

The Pig script is as below:

```
-- food business near CMU

R = LOAD '/user/cloudera/business.csv' USING PigStorage(',') AS (business_id:chararray, name:chararray, city:chararray,
type:chararray, review_count:int, stars:float, longitude:float, latitude:float);
```

```
F = FILTER R BY ((longitude>=-80.8376) AND (longitude<=-79.848) AND (latitude>=40.3687) AND
(latitude<=40.5137));

C = LOAD '/user/cloudera/categories.bag' USING PigStorage() AS b:{t:(business_id:chararray,category:chararray)};

B = FOREACH C GENERATE FLATTEN(b);

X = JOIN B BY business_id, F BY business_id;
Y = FILTER X BY (category=='Food');

-- those are business_id for Food business near CMU.
O = FOREACH Y GENERATE F::business_id as business_id, F::name as name;

D = DISTINCT O;

M = LOAD '/user/cloudera/reviewbymonth.csv' USING PigStorage(',') AS (business_id:chararray, user_id:chararray,
stars:int, month:chararray);

Z = JOIN M BY business_id, D BY business_id;

Q = FOREACH Z GENERATE M::business_id as business_id, D::name AS name, M::month AS month, M::stars AS
stars;

A = GROUP Q BY (month, business_id, name);

P = FOREACH A GENERATE FLATTEN(group), SUM(Q.stars) AS stars;

K = RANK P BY stars DESC;
L = RANK P BY stars ASC;

K10 = LIMIT K 10;
L10 = LIMIT L 10;

DUMP K10;
DUMP L10;
```

Again, here we need a couple joins to retrieve fields that exist in other dataset. F is borrowed from question 3 to find the local businesses neary CMU. We need to do “by category”, that is where B and X come from and Yto filter “Food” business only. I did not count “Restaurant”, “... Food” etc categories. In order to reduce the efforts of joins operations, I double check on “DISTINCT” datasets before joins if needed. M is the new dataset that comes with month. Here can prove the advantage of UDF again that the original dataset is with “date”, but we can just simply convert it to “month” before using it. Below is the conversion UDF python:

```
def revwbymonth(line):
    r_json = json.loads(line)
    # date in yyyy-mm-dd format
    month = r_json["date"].split("-")[1]

    revs = [r_json["business_id"], r_json["user_id"], str(r_json["stars"]), month ]

    return ",".join(revs).encode('utf-8')
```

The final results for top 10 as “rank #”, “month”, “business_id”, “name”, “stars”.

```
(1,10,NjWAVjMxUwOqnpOOQL71QA,La Gourmandine Bakery & Pastry Shop,141)
(2,08,r9rub_ajcb-3yYsYYD5apw,Proper Brick Oven & Tap Room,131)
(3,07,oN552ZjtfJWV7CMtftGrfA,Dave & Andy's,120)
(4,08,NjWAVjMxUwOqnpOOQL71QA,La Gourmandine Bakery & Pastry Shop,116)
(5,09,r9rub_ajcb-3yYsYYD5apw,Proper Brick Oven & Tap Room,114)
(6,10,r9rub_ajcb-3yYsYYD5apw,Proper Brick Oven & Tap Room,113)
(7,06,RhOTBt7ISr2rhXisSMLTSw,Spice Island Tea House,112)
(7,07,ewN984Rvux04uuQFrGu7mw,Waffallonia,112)
(9,01,XY3aIm-0PZMXXkVKECD7uw,Penn Ave Fish Company,109)
(10,09,NjWAVjMxUwOqnpOOQL71QA,La Gourmandine Bakery & Pastry Shop,107)
```

and final results for bottom 10:

```
(1,01,d6iplekfa0MMII_Hrko63g,Taste Buds,1)
(1,01,dtIfIC_I-c_Oll0jw_1Acw,Dunkin' Donuts,1)
(1,01,eUeg7tdRFvNfXjmeFReSEg,Dunkin' Donuts,1)
(1,12,2VNHbRpwZn2tg0zrckydXQ,DeLuca's Bakery,1)
(1,12,AK8T92URfkOUoTAVd7l2EA,Giant Eagle Market District,1)
(1,12,D9sHwmqlNkrVxmX-057i1g,Katerbean,1)
(1,12,G325ssFBWP4vwyifUAAF_A,Walmart,1)
(1,12,GiI6H2Z76lxlr7hnqbhFtw,Sandpresso,1)
(1,12,Lgyc-00L5pAELhjwUjAz-w,Rite Aid,1)
(1,12,WY6FFxpa6ztwDH5OhS6WDw,Potomac Bakery,1)
```

In the submitted zip, I have included the UDF converter.py, startup.sh and Pig's scripts that I used to run hadoop (script1...script10).