ASSIGNMENT 2 REPORT

Yelp Dataset Challenge

Raghul Somineni Raghupathy
Rsr379
N11371498

Question 1:

To get the results for the given constraints, I had to make use of two of the given datasets. The datasets I used were business and reviews datasets. To accommodate nested data loading, I used the twitter's elephant-bird JsonLoader to load the data on to Pig. The necessary .jar files were added to the resources under the properties tab of the pig editor. For the sake of analysis, I have taken specific attributes of both the datasets. The attributes being categories, city, business, state, latitude, longitude from business dataset and the attributes business_id and review_id from the reviews dataset. Once we have both the data, we filter it with the constraints as latitude and longitude limit for USA. After this, both the data is joined together based on their common field business_id. Once they both are joined, I generated the city and categories for each of the records in joined. Since the categories in the business dataset are nested and each business can be classified under different categories, I had to flatten the categories so that each category can be identified with business individually. Once the categories have been flattened, I had to group the variable flattened by city and categories, in order to get the results grouped respectively. Also, if we group any field, it's schema changes. So, in order to get the desired result, for each record in the grouped variable, I have flattened the grouping done previously, as city and categories and then generated the count of reviews associated with it. Eventually the results are obtained by ordering it by city, so that the final output can be arranged by showing the number of reviews for each business category within each city in the dataset. The final result is then stored in a '.tsv' file.

The Pig script for the given question is below:

Question1A

```
A = LOAD '/user/rsr379/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (yelp:map[]);

business = FOREACH A GENERATE yelp#'categories' as categories, yelp#'business_id' as business_id, yelp#'city' as city, yelp#'state' as state,
(float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude;

business_coordinates = FILTER business BY (latitude<49.384472) AND (latitude > 24.520833) AND (longitude<-66.950) AND(longitude>-124.766667);


B = FOREACH business_coordinates GENERATE categories,business_id, city;

C = LOAD '/user/rsr379/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (review: map[]);

reviews = FOREACH C GENERATE review#'business_id' as business_id, review#'review_id' as review_id;

combine = JOIN B by business_id, reviews by business_id;

flattened = FOREACH combine GENERATE city, FLATTEN(categories);

grouped = GROUP flattened by (city,categories);

result = FOREACH grouped GENERATE FLATTEN(group) AS (city,categories), COUNT(flattened);

final_result = ORDER result by city;

STORE final_result into 'Q1ResultSet.tsv';
```

The Partial result:

```
Ahwahtukee      Home Cleaning   12
Ahwahtukee      Home Services   12
Ahwahtukee      Local Services  12
Ahwahtukee      Carpet Cleaning 12
Ahwahtukee      Professional Services   12
Ahwahtukee      Office Cleaning 12
Ahwatukee       Pets    10
Ahwatukee       Pizza   191
Ahwatukee       Movers  6
Ahwatukee       Burgers 7
Ahwatukee       Doctors 18
Ahwatukee       Grocery 4
Ahwatukee       Italian 191
Ahwatukee       Dentists        18
Ahwatukee       Creperies       236
Ahwatukee       Fast Food       7
Ahwatukee       Nightlife       158
Ahwatukee       Skin Care       4
Ahwatukee       Wine Bars       158
Ahwatukee       Automotive      6
Ahwatukee       Car Rental      26
Ahwatukee       Gastropubs      158
```

Question2:

For this question I had to use the same datasets as before, i.e., review and business dataset. As before, the datasets were loaded using twitter's elephant-bird JsonLoader. This time I have generated the fields categories, city, business_id from the business dataset and the fields business_id and stars from the reviews dataset. The two datasets are joined by their common attribute business_id. Once that has been done, I generated the city, stars, categories for each of the records in the joined variable. As the categories given in the business dataset are nested and each business can be classified under various different categories, I flattened the categories so that we can identify each category associated with the business individually.

Once the categories have been flattened, I then grouped the variable flattened by city and categories, so that we can see the results grouped respectively. Since the schema changes if we group a field, I flattened the previous grouping.

The Pig Script for the question is:

Question2

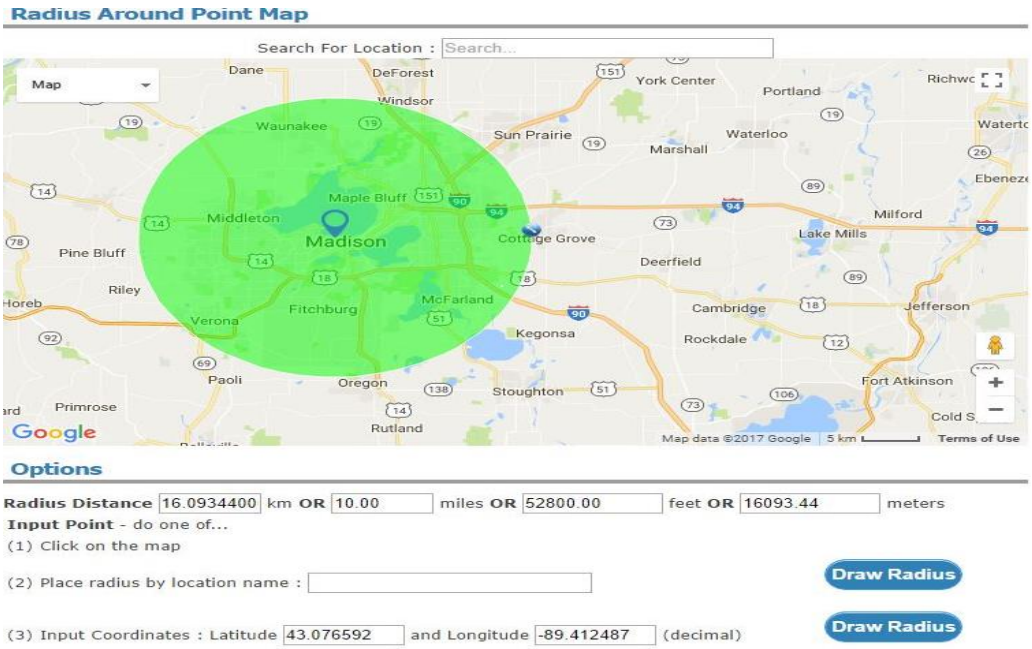
```
A = LOAD '/user/rsr379/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (yelp:map[]);

business = FOREACH A GENERATE yelp#'categories' as categories, yelp#'city' as city, yelp#'business_id' as business_id;

 B = LOAD '/user/rsr379/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (review: map[]);

reviews = FOREACH B GENERATE review#'business_id' as business_id, (INT)review#'stars' as stars;

combine = JOIN business by business_id, reviews by business_id;

flattened = FOREACH combine GENERATE city,stars, FLATTEN(categories);

grouped = GROUP flattened by (categories,city);

result = FOREACH grouped GENERATE FLATTEN(group) AS (categories,city), AVG(flattened.stars) AS ranking;

final_result = ORDER result BY categories,ranking DESC;

STORE final_result into '/user/rsr379/Q2Result.tsv';
```

The partial result set is:

```
Stuttgart       5.0
Mentor  4.833333333333333
Berea   4.333333333333333
Saint-Jean-sur-Richelieu        4.2
Chagrin Falls   4.0
Fountain Hills  4.0
Tolleson        3.0
Surprise        2.8
Carefree        2.75
Gilbert 2.7333333333333334
East York       2.6666666666666665
Urbana  2.5714285714285716
Beloeil 2.5
Sun City        2.5
Allison Park    2.5
Euclid  2.5
Laveen  2.5
Morin-Heights   2.5
Montréal        2.441860465116279
Markham 2.4285714285714284
Willowick       2.3333333333333335
Cheswick        2.3333333333333335
```

Question 3:

Datasets used in this question were business and reviews dataset. Both of them were loaded using twitter's elephant-bird JsonLoader. I then generated the fields categories, latitude, longitude, business_id from business dataset and the fields business_id and stars from the reviews dataset. In order to obtain business's within a 10 mile radius of University of Wisconsin-Madison, I put the limits of latitude and longitude which is to be satisfied for the business to be in the specified radius. The limits were got from the web using the image below:



The two are joined together by their common attribute business_id. Once they were joined I generated the stars, categories for each of the records in the joined variable. Since each business in the dataset can be classified under different categories, I had to flatten the categories so that we can identify the category associated with each business. Once the categories have been flattened, I grouped the variable flattened by categories, so that we can see the results grouped accordingly. Since the schema changes on grouping, I have flattened the previous grouping as categories and generated the average value of the stars within that group and renamed the calculated field as rankings.

The Pig Script for the question is:

Question3

```
A = LOAD '/user/rsr379/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (yelp: map[]);

business = FOREACH A GENERATE yelp#'categories' as categories, yelp#'business_id' as business_id,(float)yelp#'latitude' as latitude, (float)yelp#'longitude'

business_radius = FILTER business BY (latitude<43.2192) AND (latitude>42.9398) AND (longitude<-89.2461) AND (longitude>-89.6024);

B = LOAD '/user/rsr379/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (review: map[]);

reviews = FOREACH B GENERATE review#'business_id' as business_id, (int)review#'stars' as stars;

combine = JOIN business_radius by business_id, reviews by business_id;

flattened = FOREACH combine GENERATE stars, FLATTEN(categories);

grouped = GROUP flattened by categories;

result = FOREACH grouped GENERATE FLATTEN(group) AS categories, AVG(flattened.stars) AS rankings;

q2result = ORDER result BY categories;

STORE q2result into '/user/rsr379/Q3Result.tsv';
```

The Partial Result is :

```
Accessories     3.8029556650246303
Accountants     4.411764705882353
Active Life     4.091328886608517
Acupuncture     4.487179487179487
Adult   4.214285714285714
Adult Education 4.214285714285714
Advertising     5.0
Afghan  3.668918918918919
African 3.6292134831460676
Air Duct Cleaning       4.933333333333334
Aircraft Dealers        4.5
Aircraft Repairs        4.5
Airport Shuttles        3.317073170731707
Airports        4.059210526315789
Allergists      2.1666666666666665
Amateur Sports Teams    4.0
American (New)  3.8094785979903474
American (Traditional)  3.540144727773949
Amusement Parks 3.5
Animal Shelters 3.4210526315789473
Antiques        4.1192660550458715
```

Question 4:

For this question, I have made use of three datasets from the Yelp Academic Dataset. The datasets being Business, Users and Reviews dataset. All the dataset were loaded in Pig using Twitter's Elephant-bird JsonLoader. The '.jar' files needed to use the JsonLoader were added under resources the properties tab of the pig editor. The first part of the question involves finding out the top reviewers based on their review count. For this, we use the user dataset and get the attributes user_id, review_count. Using the ORDER BY and DESC we can sort the data based on the field necessary, in this case the review_count. For the second part, I have ordered the user's information by review count and limited the set to only the users with the 10 highest number of reviews. I then joined this with the reviews dataset. The is then joined to the business dataset. After this, they are flattened according to their names and the categories they belong to along with the average ratings for each category they have reviewed.

The Pig Script is:

Question4

```
A = LOAD '/user/rsr379/yelp_academic_dataset_user.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (yelp: map[]);

users = FOREACH A GENERATE yelp#'user_id' as user_id, yelp#'name' as name, (int)yelp#'review_count' as review_count;

B = LOAD '/user/rsr379/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (review: map[]);

reviews = FOREACH B GENERATE review#'business_id' as business_id, (int)review#'stars' as stars, review#'user_id' as user_id;

C = LOAD '/user/rsr379/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (busines: map[]);

business = FOREACH C GENERATE busines#'business_id' as business_id, busines#'categories' as categories;

ranked_users = ORDER users by review_count DESC;

STORE ranked_users INTO '/user/rsr379/RankedUsers.tsv';

ordered = ORDER users by review_count DESC ;

top_users = LIMIT ordered 10;

combine = JOIN top_users by user_id, reviews by user_id ;

tab1 = FOREACH combine GENERATE name as Name, stars as Stars, business_id as Business_ID;

combine2 = JOIN tab1 by Business_ID, business by business_id;

flattened = FOREACH combine2 GENERATE Stars, Name, FLATTEN(categories);

grouped = GROUP flattened by (Name,categories);

results = FOREACH grouped GENERATE FLATTEN(group) AS (Name, categories) , AVG(flattened.Stars) AS rankings;

STORE results INTO '/user/rsr379/Q4ResultSet.tsv';
```

The Partial Output is:

```
Dan     Bars    2.5
Dan     Food    3.7333333333333334
Dan     Thai    3.0
Dan     Used    3.0
Dan     Zoos    4.0
Dan     Cafes   4.0
Dan     Greek   2.0
Dan     Irish   3.0
Dan     Parks   4.0
Dan     Cinema  3.0
Dan     Tennis  4.0
Dan     Burgers 3.0
Dan     Fashion 3.0
Dan     Framing 4.0
Dan     Grocery 4.0
Dan     Italian 3.0
Dan     Jewelry 4.0
Dan     Museums 4.5
Dan     Seafood 2.0
Dan     Bakeries        4.0
Dan     Barbeque        4.0
Dan     Desserts        4.0
Dan     Shopping        4.0
Dan     Southern        4.0
Dan     Aquariums       5.0
```

Question 5:

For this, I first loaded the business dataset using Twitter's Elephant-bird JsonLoader and generated the attributes name, categories, business_id, latitude, longitude, stars. Then I filtered the data according to the location specification of University of Wisconsin-Madison. After this I generated the attributes other than latitude and longitude since they won't be needed anymore. At the same time, I have converted the categories field to bag using the TOBAG operator. The big was then converted to a string in the same step using the BagToString function. This is to easily filter the categories based on the constraint that they have food in them. Once the categories have been filtered, the businesses were ordered in the ascending order based on their stars and limited to 10 businesses. For the bottom 10 part of the question, the same thing was done but the ordering is in the descending order of stars.

The Pig Script is:

q5withmonth

```
A = LOAD '/user/rsr379/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (yelp: map[]);

business = FOREACH A GENERATE yelp#'categories' as categories, (float)yelp#'stars' as stars, yelp#'name' as name, yelp#'business_id' as business_id,(float)yelp#'latitude' as latitude, (float)yelp#'longitude' as

business_radius = FILTER business BY (latitude<43.2192) AND (latitude>42.9398) AND (longitude<-89.2461) AND (longitude>-89.6024);

processedtable = FOREACH business_radius GENERATE name, stars, business_id, org.apache.pig.builtin.BagToString(TOBAG(categories)) as category;

filtered = FILTER processedtable BY category matches '.*Food.*';

ordered = ORDER filtered by stars DESC;

top_10 = limit ordered 10;

B = LOAD '/user/rsr379/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true ') AS (review: map[]);

reviews = FOREACH B GENERATE (float)review#'rating' as ratings, review#'business_id' as business_id, review#'date' as date;

combine = JOIN top_10 by business_id, reviews by business_id;

tab2 = FOREACH combine GENERATE $0 as name, $1 as ratings, $5 as business_id, (int)SUBSTRING($6, 5, 7) as month;

filtered2 = FILTER tab2 BY (month>0) AND (month<6);

grouped = GROUP filtered2 by (business_id, name, month);

flattened = FOREACH grouped GENERATE FLATTEN (group) as (business_id, name, month), AVG(filtered2.ratings);

STORE flattened INTO '/user/rsr379/TOP_10_FOODS_month.tsv';

ordered_desc = ORDER filtered by stars;

bottom_10 = limit ordered_desc 10;

combineb = JOIN bottom_10 by business_id, reviews by business_id;

tab3 = FOREACH combineb GENERATE $0 as name, $1 as ratings, $5 as business_id, (int)SUBSTRING($6, 5, 7) as month;

filtered3 = FILTER tab3 BY (month>0) AND (month<6);

groupedb = GROUP filtered3 by (business_id, name, month);

flattenedb = FOREACH groupedb GENERATE FLATTEN (group) as (business_id, name, month), AVG(filtered3.ratings);

STORE flattenedb INTO '/user/rsr379/BOTTOM_10_FOODS_month.tsv';
```

The Partial result:

Top 10 Business:

```
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  5      5.0
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  6      5.0
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  7      5.0
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  8      5.0
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  9      5.0
4y8KM5HqQHOm6M0w7O-m4g  Madison Food Explorers  10     5.0
7e1rkqyJpXNAQfcFsOQRYw  Windsor Breads Bakery & Coffeehouse    2      5.0
7e1rkqyJpXNAQfcFsOQRYw  Windsor Breads Bakery & Coffeehouse    3      5.0
7e1rkqyJpXNAQfcFsOQRYw  Windsor Breads Bakery & Coffeehouse    5      5.0
7e1rkqyJpXNAQfcFsOQRYw  Windsor Breads Bakery & Coffeehouse    7      5.0
7e1rkqyJpXNAQfcFsOQRYw  Windsor Breads Bakery & Coffeehouse    8      5.0
DUUY9UpEbA69xhketnsdLw  Macha Tea       1      5.0
DUUY9UpEbA69xhketnsdLw  Macha Tea       4      5.0
DUUY9UpEbA69xhketnsdLw  Macha Tea       5      5.0
DUUY9UpEbA69xhketnsdLw  Macha Tea       9      5.0
DUUY9UpEbA69xhketnsdLw  Macha Tea       12     5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   1      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   2      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   3      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   4      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   5      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   6      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   7      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   8      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   9      5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   10     5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   11     5.0
HOMqym6ceItYao90zNPrkg  La Michoacana   12     5.0
KBWpa4uN2ke1t2cp27r9tw  Zen Sushi       2      5.0
KBWpa4uN2ke1t2cp27r9tw  Zen Sushi       5      5.0
KBWpa4uN2ke1t2cp27r9tw  Zen Sushi       11     5.0
WGg8UkesbPhi4HYIzHzxfw  Growlers To Go-Go       5      5.0
```

Bottom 10 Business:

8RbNzNeaUh5e8PJOnp1e4Q	McDonald's	6	1.0
8RbNzNeaUh5e8PJOnp1e4Q	McDonald's	9	1.0
8yaCjxIqYsPhiu6ZgD4ZlA	Sushi Hut	1	1.5
8yaCjxIqYsPhiu6ZgD4ZlA	Sushi Hut	2	1.5
8yaCjxIqYsPhiu6ZgD4ZlA	Sushi Hut	8	1.5
8yaCjxIqYsPhiu6ZgD4ZlA	Sushi Hut	9	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	1	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	4	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	5	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	7	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	9	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	11	1.5
Aua2Vi32K-G4LE0JCtAwiw	McDonald's	12	1.5
LtIwF6HuA2dGWJ7OpvLHog	Capitol Café.	5	1.0
LtIwF6HuA2dGWJ7OpvLHog	Capitol Café.	7	1.0
LtIwF6HuA2dGWJ7OpvLHog	Capitol Café.	10	1.0
_VG3IAIbXdends4ATxlEiA	McDonald's	1	1.0
_VG3IAIbXdends4ATxlEiA	McDonald's	5	1.0
_VG3IAIbXdends4ATxlEiA	McDonald's	6	1.0
_VG3IAIbXdends4ATxlEiA	McDonald's	10	1.0
d2VdAB1-AasPxxLPL6QE4A	McDonald's	2	1.0
d2VdAB1-AasPxxLPL6QE4A	McDonald's	7	1.0
d2VdAB1-AasPxxLPL6QE4A	McDonald's	8	1.0
d2VdAB1-AasPxxLPL6QE4A	McDonald's	10	1.0
d2VdAB1-AasPxxLPL6QE4A	McDonald's	11	1.0
gKJFQd2l1CRmeSR5eMButg	Walgreens	1	1.0
gKJFQd2l1CRmeSR5eMButg	Walgreens	10	1.0
gKJFQd2l1CRmeSR5eMButg	Walgreens	11	1.0
pNgDhHaeFkWQW71NzT9wzg	KFC	6	1.0
pNgDhHaeFkWQW71NzT9wzg	KFC	8	1.0
pNgDhHaeFkWQW71NzT9wzg	KFC	10	1.0
s8sAe29mA8q1eQwdC1qAKw	Wendy's	2	1.5