Assignment-2

Course: CS 9223
Semester: Spring 2017
Advisor: Professor Juan Rodriguez
Submitted By: Shivaraj Shankaralinga Nesaragi
Net ID: ssn314
UniversityID: N1558346
Date of Submission:3/12/2017

INITIAL SETUP:

These are the initial things that need to be done before you start writing your Pig Script for yelp dataset.

1. Download the yelp dataset from the Yelp website and untar the file to extract 5 Json files.
2. Upload these 5 json files in the hpc. I uploaded these files in the path /user/ssn314/yelp/ folder.
3. Download the following jars: **elephant-bird-core-4.15-RC1.jar, elephant-bird-hadoop-compat-4.15-RC1.jar,elephant-bird-pig-4.15-RC1.jar.**
4. Upload these jars to your Pig Script Interface in the hpc. I uploaded these files in the path /user/ssn314/.

Q1) Summarize the number of reviews by US city, by business category.

Steps:

- Open up the pig script terminal and click on New Script. Click on the properties tab and add the 3 elephant jars.
- The first line of Pig Script should say **SET elephantbird.jsonloader.nestedLoad 'true'**.
- After this is done, load the yelp_academic_dataset_business.json into the workspace.
- The next step would be to select the fields city,review count and the flattened categories from the json File loaded in the previous step.
- We then need to group the data got from the previous step by city and categories.
- The next step would be to select the cities, categories and sum of the review count from the group data. This is done in Pig with the help of FOREACH and GENERATE commands.
- Once this is done, we use FILTER command to get the data for US cities.
- Then we ORDER by city in order to get the data in a proper sorted format.
- We then store the data got from previous step in the output file. The path of the output file is /user/ssn314/PigOutput/q1Output.

Script:

```
SET elephantbird.jsonloader.nestedLoad 'true';

yelp_business_data = LOAD '/user/ssn314/yelp/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as (json:map[]);

Yelp_business_data_category = FOREACH yelp_business_data GENERATE (int)json#'review_count' AS review_count, json#'city' as city, FLATTEN(json#'categories') AS categories;
```

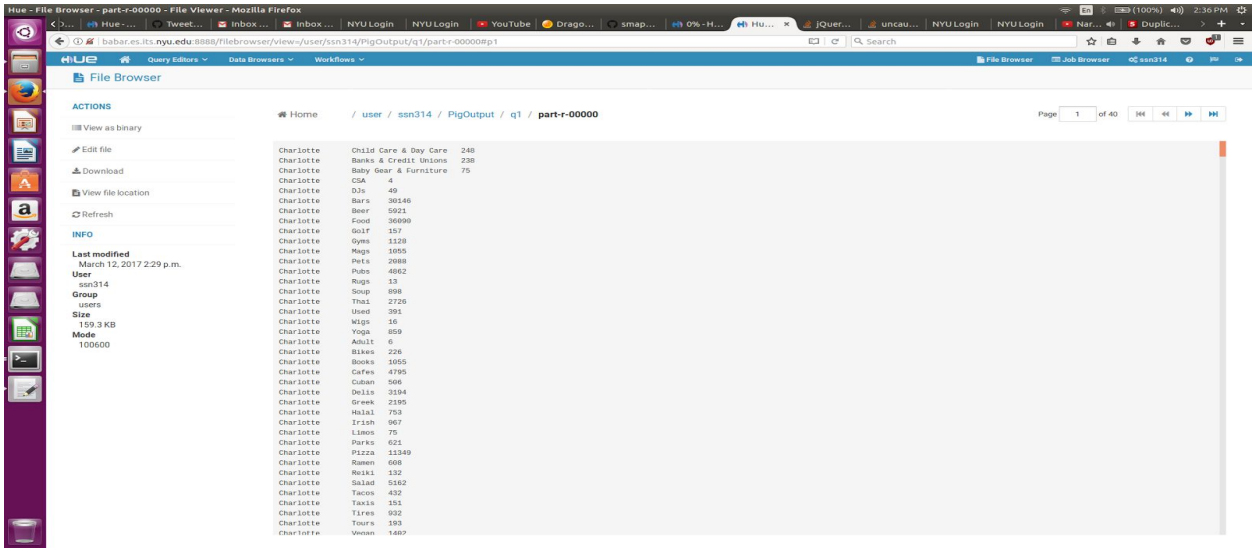
```
yelp_business_group= GROUP yelp_business_data_category BY (city,categories);

yelp_business_group_data= FOREACH yelp_business_group GENERATE group.city as
city,group.categories as category,SUM(yelp_business_data_category.review_count) AS
reviewCount;

yelp_business_us_cities= FILTER yelp_business_group_data BY city MATCHES
'.*(Charlotte|Cleveland|Madison|Pittsburgh|Phoenix|Urbana|Las Vegas).*';

 yelp_business_order= ORDER yelp_business_us_cities BY city;

STORE yelp_business_order INTO '/user/ssn314/PigOutput/q1Output';
```

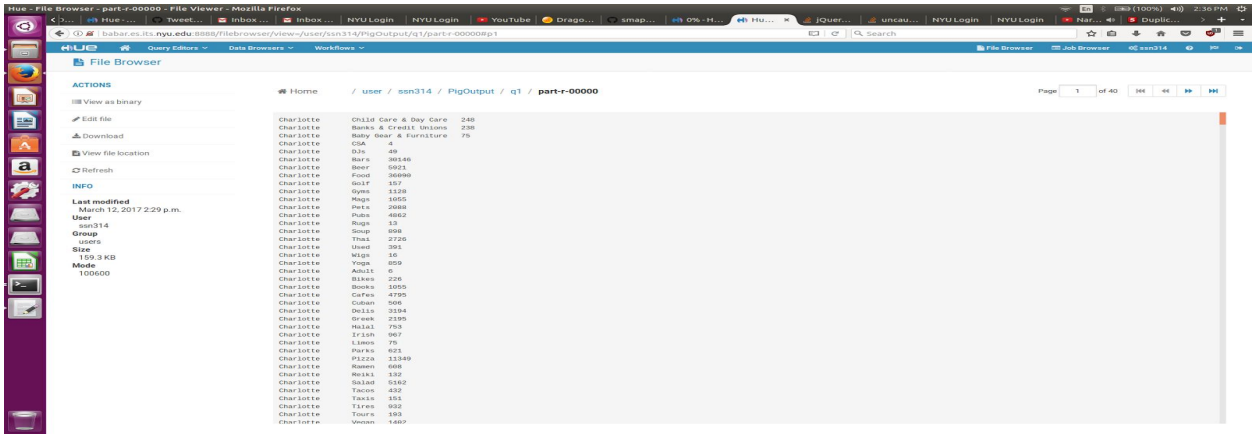
Spark Script:

```
import spark.implicits._
import org.apache.spark.sql.functions._
import scala.collection.mutable.WrappedArray

val spark = new org.apache.spark.sql.SQLContext(sc)

val business = spark.read.json("/usr/shivraj/yelp_academic_dataset_business.json")

val b = business.withColumn("category", explode(
        when(col("categories").isNotNull, col("categories"))
        .otherwise(array(lit(null).cast("string")))
        ))

b.registerTempTable("business")

val business_df = spark.sql("select city,category,sum(review_count) from business where
city='Madison' OR city='Pittsburgh' OR city='Charlotte' OR city='Phoenix' OR city='Cleveland' OR
city='Urbana' OR city='Las Vegas' group by city,category Order by city")
business_df.write
  .option("header", "true")
  .csv("/usr/shivraj/q1Spark.csv")
```

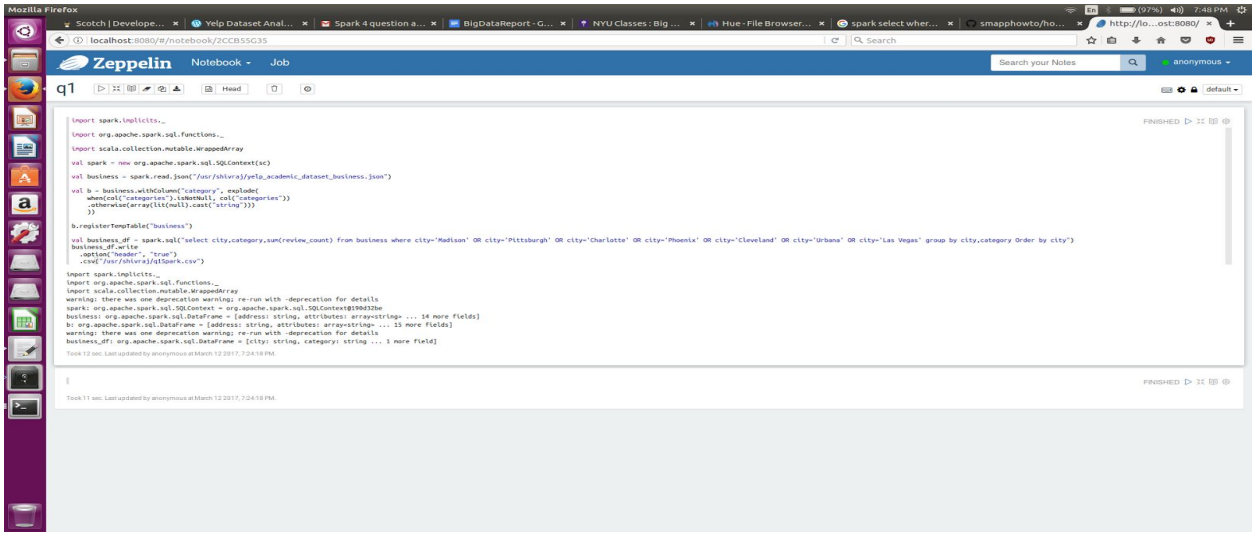
OUTPUT:

Pig Script Output:





Spark Script Output:



Q2) Rank all *cities* by # of stars descending, for each category.

Steps:

- Open up the pig script terminal and click on New Script. Click on the properties tab and add the 3 elephant jars.
- The first line of Pig Script should say **SET elephantbird.jsonloader.nestedLoad 'true'**.
- After this is done, load the yelp_academic_dataset_business.json into the workspace.
- Select the stars, city and flattened categories from the business Json and store it in a relation.
- Then group the previous relation by categories and city in '**yelp_business_group**' relation.
- Then generate the fields categories, city and average of stars and store it in **yelp_business_group_data** relation.
- Then sort the relation by category ascending and stars descending and store the relation in **q2FinalOutput.out.**
- The final output contains the cities ranked by stars descending for each category.

Script:

```
SET elephantbird.jsonloader.nestedLoad 'true';

yelp_business_data = LOAD '/user/ssn314/yelp/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as (json:map[]);

yelp_business_data_category = FOREACH yelp_business_data GENERATE (float)json#'stars' AS stars,json#'city' as city, FLATTEN(json#'categories') AS categories;

yelp_business_group= GROUP yelp_business_data_category BY (categories,city);

yelp_business_group_data= FOREACH yelp_business_group GENERATE group.categories as category,group.city AS city, AVG(yelp_business_data_category.stars) AS stars;

yelp_data_order= ORDER yelp_business_group_data BY category ASC,stars DESC;

STORE yelp_data_order INTO '/user/ssn314/PigOutput/q2FinalOutput.out';
```

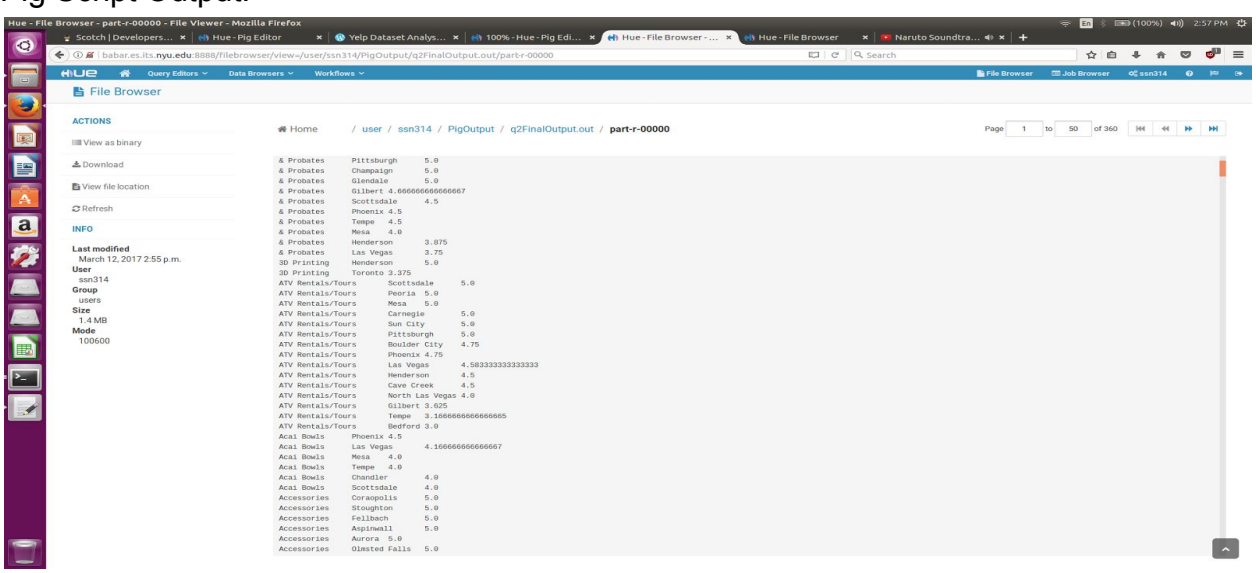
Spark Script:

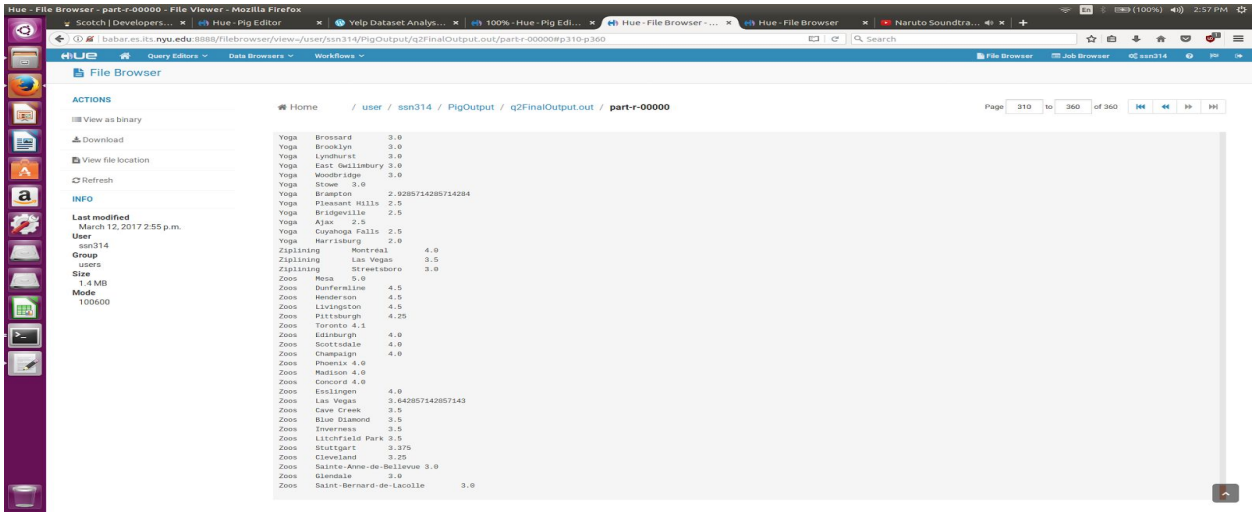
```
import spark.implicits._

import org.apache.spark.sql.functions._

import scala.collection.mutable.WrappedArray
```

```
val spark = new org.apache.spark.sql.SQLContext(sc)

val business = spark.read.json("/usr/shivraj/yelp_academic_dataset_business.json")

val b = business.withColumn("category", explode(
        when(col("categories").isNotNull, col("categories"))
        .otherwise(array(lit(null).cast("string")))
        ))

b.registerTempTable("business")

val business_df = spark.sql("select category,city,AVG(stars) as stars from business group by category,city Order by category,stars DESC")

business_df.write
 .option("header", "true")
 .csv("/usr/shivraj/q2Spark.csv")
```

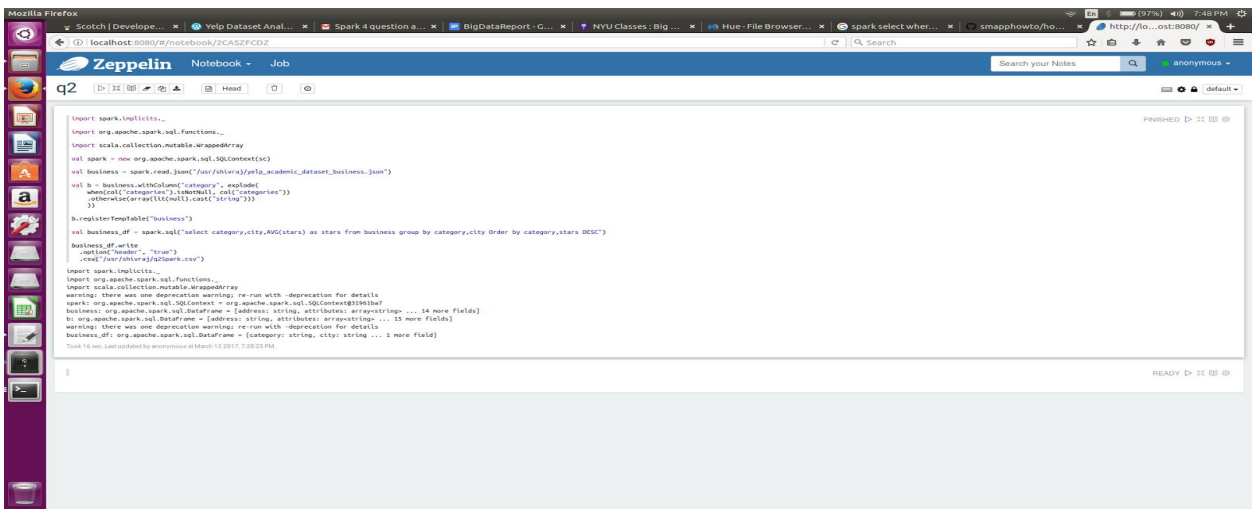
OUTPUT:

Pig Script Output:





Spark Script Output:



Q3) What is the average rank (# stars) for businesses within 10 miles of the University of Wisconsin - Madison, by type of business?

Steps:

- Open up the pig script terminal and click on New Script. Click on the properties tab and add the 3 elephant jars.
- The first line of Pig Script should say **SET elephantbird.jsonloader.nestedLoad 'true'**.
- After this is done, load the yelp_academic_dataset_business.json into the workspace.
- Then we select the stars, latitude,longitude, flattened categories and store in the relation **yelp_business_data_category.**
- Then we need to filter the previous relation based on the bounding box near Wisconsin. This is done using FILTER command and then filtered based on latitude and longitude values.

- Then GROUP the relation by categories and store in the relation **yelp_Wisconsin_Category**
- Then from this relation select categories from group and average of the stars.
- Then we ORDER by stars descending to get the categories rated according to stars.

Script:

```
SET elephantbird.jsonloader.nestedLoad 'true';

yelp_business_data = LOAD '/user/ssn314/yelp/yelp_academic_dataset_business.json' USING
                        com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as (json:map[]);

yelp_business_data_category = FOREACH yelp_business_data GENERATE
                                (float)json#'stars' AS stars,
                                  (float)json#'latitude' AS latitude,
                                  (float)json#'longitude' AS longitude,
                                FLATTEN(json#'categories') AS categories;

yelp_Wisconsin=  FILTER yelp_business_data_category BY (latitude>42.9083)
                 AND (latitude<43.2417)
                  AND (longitude>-89.5839)
                  AND (longitude<-89.2506);

yelp_Wisconsin_Category= GROUP yelp_Wisconsin BY categories;

yelp_Wisconsin_BoundBox= FOREACH yelp_Wisconsin_Category generate group AS categories,
                            AVG(yelp_Wisconsin.stars) AS stars;


yelp_Wisconsin_BoundBox_Ordered= ORDER yelp_Wisconsin_BoundBox BY stars DESC;


STORE yelp_Wisconsin_BoundBox_Ordered INTO '/user/ssn314/PigOutput/q3Final.out';
```

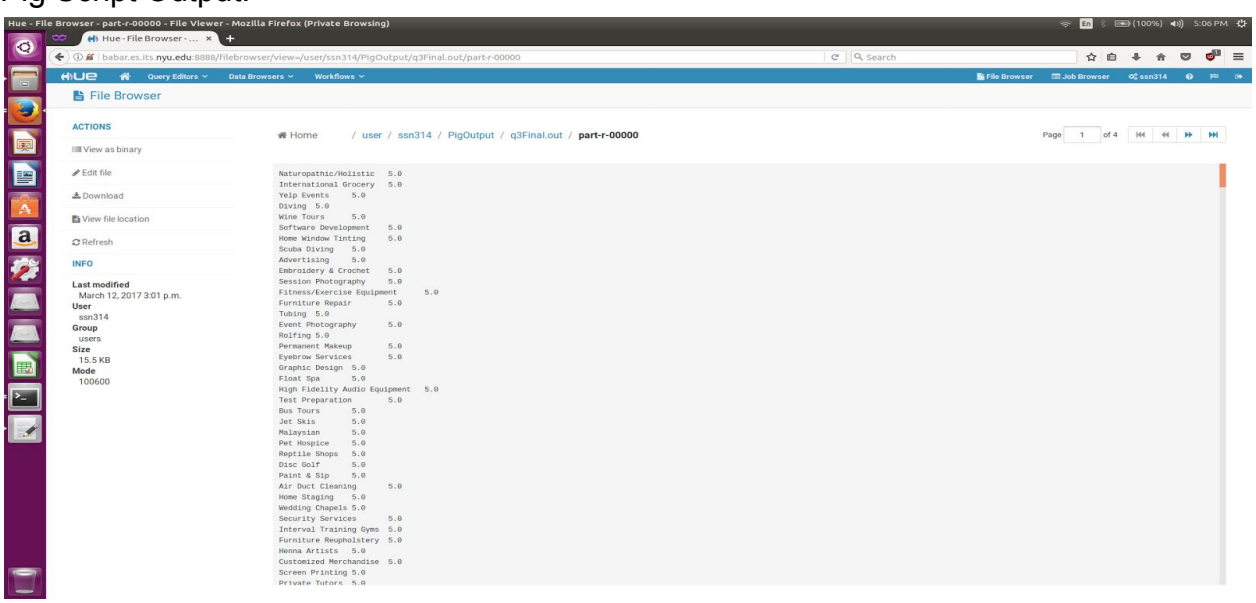
SPARK SCRIPT:

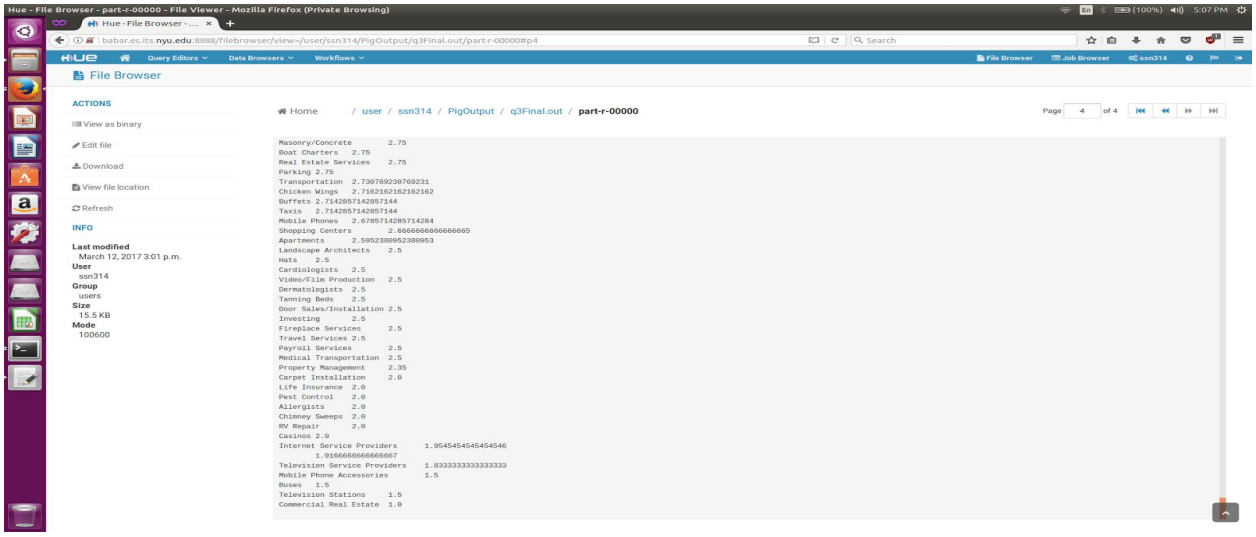
```
import spark.implicits._

import org.apache.spark.sql.functions._

import scala.collection.mutable.WrappedArray
```

```
val spark = new org.apache.spark.sql.SQLContext(sc)

val business = spark.read.json("/usr/shivraj/yelp_academic_dataset_business.json")

val b = business.withColumn("category", explode(
        when(col("categories").isNotNull, col("categories"))
        .otherwise(array(lit(null).cast("string")))
        ))

b.registerTempTable("business")

val q3Output=spark.sql("SELECT category,AVG(stars) AS stars from business WHERE latitude<=43.2467 and latitude>=42.90833 and longitude>=-89.58389 and longitude<=-89.25056 GROUP BY category ORDER BY stars DESC")

q3Output.write
 .option("header", "true")
 .csv("/usr/shivraj/q3Spark.csv")
```

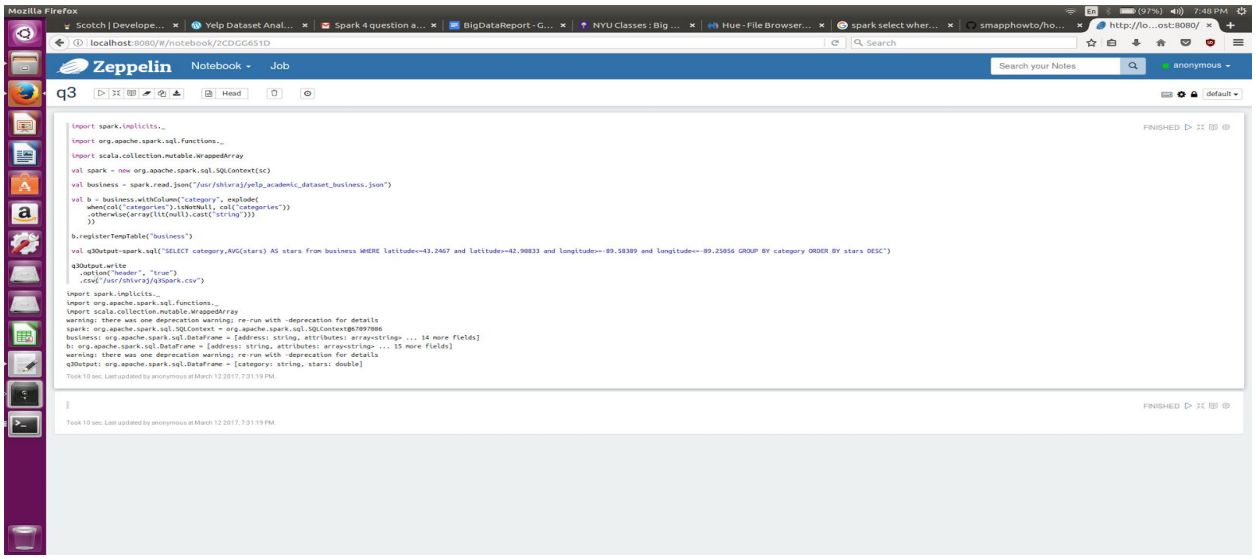
OUTPUT:

Pig Script Output:





Spark Script Output:



Q4) Rank reviewers by number of reviews. For the top 10 reviewers, show their average number of stars, by category.

Steps:

- Open up the pig script terminal and click on New Script. Click on the properties tab and add the 3 elephant jars.
- The first line of Pig Script should say **SET elephantbird.jsonloader.nestedLoad 'true'**.
- After this is done, load the yelp_academic_dataset_user.json into the workspace.
- Select the userId and review count from the userJson and store in the **user_data** relation.
- Select the top 10 reviewers from then.
- Then load the review Json into the workspace.
- Select the userId ,stars and businessId from the review Json.

- We then join user relation for top 10 reviewers and review Json by userId.
- We then select the userId, businessId and stars from this joined relation and store it in **review_user_join** relation..
- The next step is to load the business Json.
- Then we select business Id and flattened categories and store in the relation **categoryID**
- We then join the **categoryID** relation and the **review_user_join** relation by businessId.
- Then we select userId,category and stars from this relation and group it by userid and category and store it in **reviewGrouped** relation.
- The last step is to select the userId,category and average of stars for each category and store the relation in the output directory **/user/ssn314/PigOutput/Q4FinalOutput.out.**

Script:

```
SET elephantbird.jsonloader.nestedLoad 'true';

userJson = LOAD '/user/ssn314/yelp/yelp_academic_dataset_user.json' USING
        com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as (users:map[]);

user_data = FOREACH userJson GENERATE (chararray)users#'user_id' as user_id,
                        (int)users#'review_count' as review_count;

user_ordered_data = ORDER user_data by review_count DESC;

top_10_reviewers = LIMIT user_ordered_data 10;

reviewJson = LOAD '/user/ssn314/yelp/yelp_academic_dataset_review.json' USING
                        com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad')        as
(review:map[]);

review_data = FOREACH reviewJson GENERATE (chararray)review#'user_id' as user_id,
                        (float)review#'stars' as stars,(chararray)review#'business_id' as business_id;

review_user_join = JOIN review_data BY user_id, top_10_reviewers by user_id;

review_counter = FOREACH review_user_join GENERATE review_data::user_id as user_id,
                review_data::business_id as business_id,review_data::stars as stars;

businessJson = LOAD '/user/ssn314/yelp/yelp_academic_dataset_business.json' USING
        com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as (business:map[]);

categoryID = FOREACH businessJson GENERATE (chararray)business#'business_id' AS business_id,
        FLATTEN(business#'categories') AS category;
```

```
reviewJoined = JOIN categoryID BY business_id, review_counter BY business_id;

reviewFinal = FOREACH reviewJoined GENERATE review_counter::user_id as user_id,
                                categoryID::category as category, review_counter::stars as stars;

reviewGrouped = GROUP reviewFinal by (user_id,category);

finalOutput = FOREACH reviewGrouped GENERATE group.user_id as user_id,
                                group.category as category, AVG(reviewFinal.stars) as stars;

STORE top_10_reviewers into '/user/ssn314/PigOutput/Q4_top_ten_reviewers.out';

STORE finalOutput into '/user/ssn314/PigOutput/Q4FinalOutput.out';
```

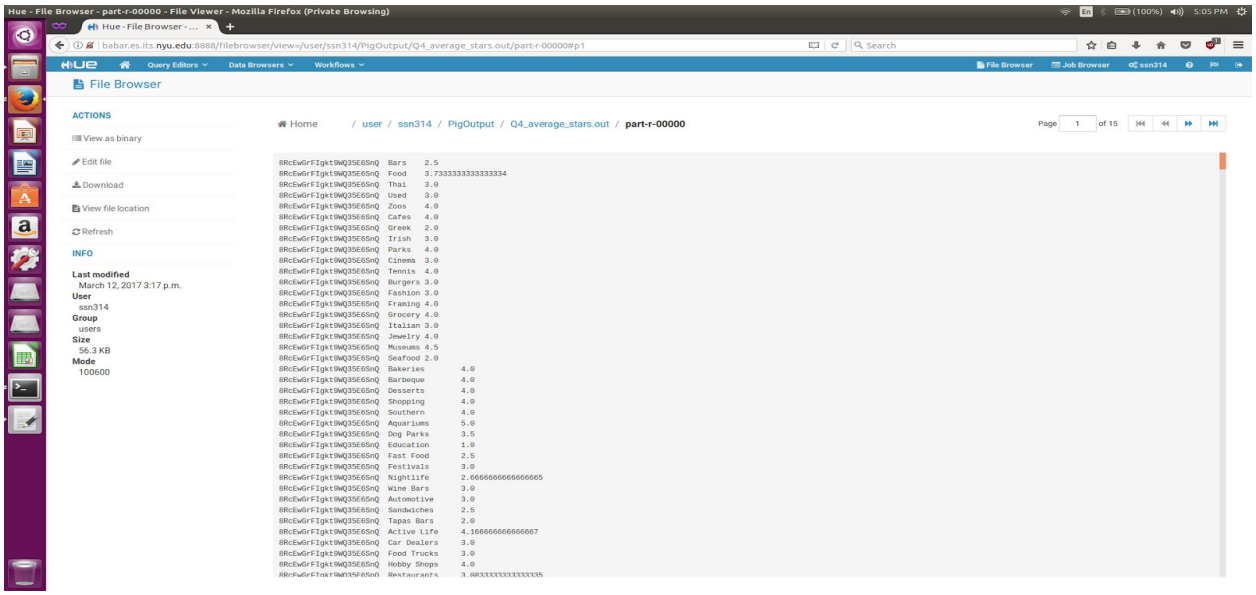
Spark Script:

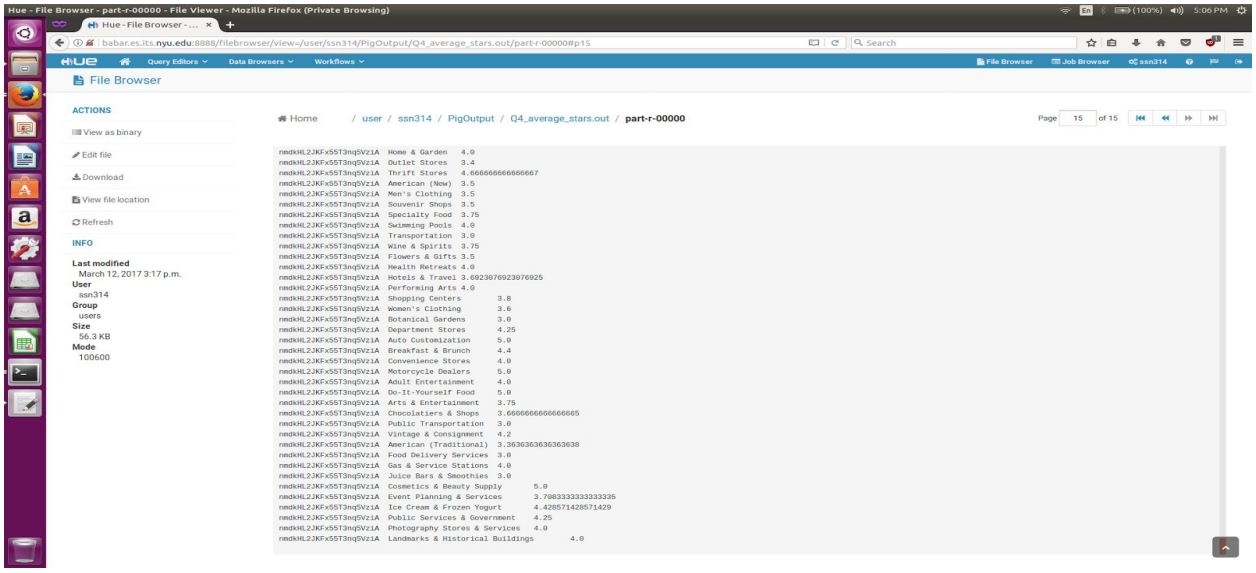
```
import spark.implicits._

import org.apache.spark.sql.functions._

import scala.collection.mutable.WrappedArray

val spark = new org.apache.spark.sql.SQLContext(sc)

val business = spark.read.json("/usr/shivraj/yelp_academic_dataset_business.json")

val b = business.withColumn("category", explode(
        when(col("categories").isNotNull, col("categories"))
        .otherwise(array(lit(null).cast("string")))
        ))

val user = spark.read.json("/usr/shivraj/yelp_academic_dataset_user.json")

val review = spark.read.json("/usr/shivraj/yelp_academic_dataset_review.json")

b.registerTempTable("business")
user.registerTempTable("user")
review.registerTempTable("review")

val top_10_reviewers=spark.sql("SELECT user_id,review_count FROM user ORDER BY review_count
DESC LIMIT 10")

top_10_reviewers.registerTempTable("top_10_reviewers")
```

```
val userReviewJson=spark.sql("SELECT top.user_id,r.business_id,r.stars FROM top_10_reviewers AS top,review AS r WHERE r.user_id=top.user_id")

userReviewJson.registerTempTable("userReviewJson")

val q4Final=spark.sql("SELECT u.user_id,avg(u.stars),b.category FROM userReviewJson AS u,business AS b where b.business_id=u.business_id GROUP BY u.user_id,b.category ORDER BY user_id,category")

q4Final.write
 .option("header", "true")
 .csv("/usr/shivraj/q4Spark.csv")
```

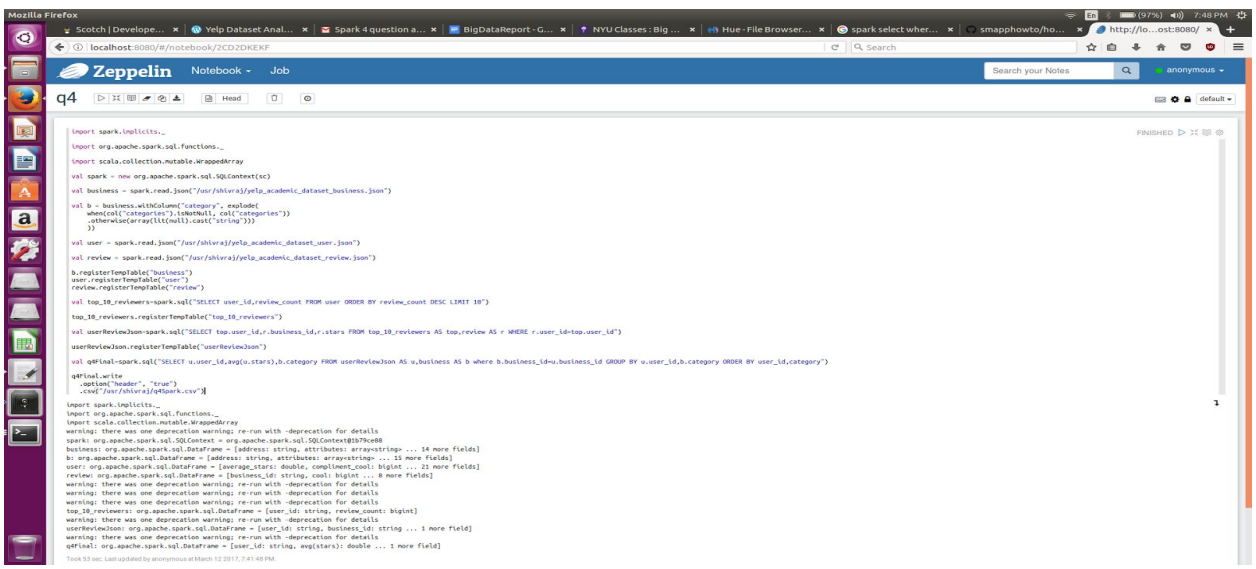
OUTPUT:

Pig Script Output:





Spark Script Output:



Q5) For the top 10 and bottom 10 food business near UWM (in terms of stars), summarize star rating for reviews in January through May.

Steps:

- Open up the pig script terminal and click on New Script. Click on the properties tab and add the 3 elephant jars.
- The first line of Pig Script should say **SET elephantbird.jsonloader.nestedLoad 'true'**.
- After this is done, load the yelp_academic_dataset_business.json and yelp_academic_dataset_review.json into the workspace.
- We select the businessId,latitude,stars,longitude and flattened categories and store it in the **yelp_business_data_category.**

- We select the businessId, reviewId,userId,stars and date and store it in **yelp_review_category** relation.
- The next step is to select the reviews which are in the month range of January to May.
- We then select the Food category restaurants that are near Wisconsin.
- Then from that we order the restaurants by stars and take the top 10 and bottom 10 restaurants from this relation.
- We then make the UNION of the top 10 and bottom 10 relations and store it in a relation **yelp_top_bottom_10.**
- Then we join the **yelp_top_bottom_10** and **yelp_review_date** by businessId.
- We then select the businessId,reviewId,userId,stars,date.
- We then group by businessID and store it in the relation **yelp_biz_group.**
- From the **yelp_biz_group** we select businessId, average of stars and date and store the output in **q5Final.out**

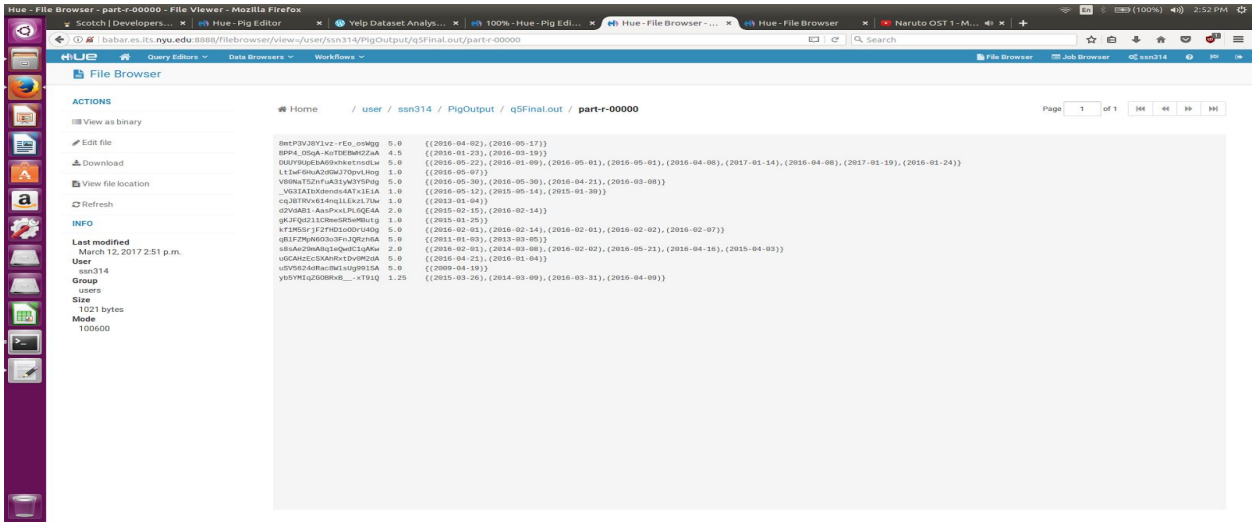
Script:

```
SET elephantbird.jsonloader.nestedLoad 'true';

yelp_business_data = LOAD '/user/ssn314/yelp/yelp_academic_dataset_business.json' USING com.twitter.elephantbird.pig.load.JsonLoader
                                ('-nestedLoad') as (business:map[]);

yelp_review_data = LOAD '/user/ssn314/yelp/yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader
                                ('-nestedLoad') as (review:map[]);

yelp_business_data_category = FOREACH yelp_business_data GENERATE
                                  (chararray)business#'business_id' AS businessID,
                                  (float)business#'latitude' AS latitude,
                                  (float)business#'stars' AS stars,
                                  (float)business#'longitude' AS longitude,
                                FLATTEN(business#'categories') AS categories;

yelp_review_category = FOREACH yelp_review_data GENERATE
                              (chararray)review#'business_id' AS businessID,
                          (chararray)review#'review_id' AS reviewID,
                           (chararray)review#'user_id' AS userID,
                           (float)review#'stars' AS stars,
                           (chararray)review#'date' AS date;

yelp_review_date= FILTER yelp_review_category BY (SUBSTRING(date,5,7) MATCHES
'.*(01|02|03|04|05).*');
```

```
yelp_Wisconsin= FILTER yelp_business_data_category BY (latitude>42.9083)
                    AND (latitude<43.2417)
                     AND (longitude>-89.5839)
                     AND (longitude<-89.2506);


yelp_Wisconsin_Food_Filter= FILTER yelp_Wisconsin BY categories MATCHES '.*Food*.';

yelp_Wisconsin_BoundBox_desc= ORDER yelp_Wisconsin_Food_Filter BY stars DESC;

yelp_top_10_Wisconsin= LIMIT yelp_Wisconsin_BoundBox_desc 10;

yelp_Wisconsin_BoundBox_asc= ORDER yelp_Wisconsin_Food_Filter BY stars ASC;

yelp_bottom_10_Wisconsin= LIMIT yelp_Wisconsin_BoundBox_asc 10;

yelp_top_bottom_10= UNION yelp_top_10_Wisconsin,yelp_bottom_10_Wisconsin;

STORE yelp_top_bottom_10 INTO '/user/ssn314/PigOutput/q5TopBottomUnion';

yelp_joined_data= JOIN yelp_top_bottom_10 BY businessID,yelp_review_date BY businessID;

STORE yelp_joined_data INTO '/user/ssn314/PigOutput/q5BizIdJoined';

yelp_joined_review_data= FOREACH yelp_joined_data GENERATE yelp_top_bottom_10::businessID
AS bizID,reviewID AS reviewID, userID AS userID,yelp_review_date::stars AS stars,date AS date;

yelp_biz_group= GROUP yelp_joined_review_data BY bizID;

STORE yelp_biz_group INTO '/user/ssn314/PigOutput/q5BizGrouped';

yelp_final_data= FOREACH yelp_biz_group GENERATE group AS bizID,
AVG(yelp_joined_review_data.stars) AS stars,yelp_joined_review_data.date AS date;


STORE yelp_final_data INTO '/user/ssn314/PigOutput/q5Final.out';
```

OUTPUT:

Pig Script Output:



Spark Script Output: