NYU CS9223 Big Data Programming

Spring 2017

Assignment 2 – *Yelp Data Challenge*

Report by
Monica Reddy Tirupari
N11349565

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

Pig Script:

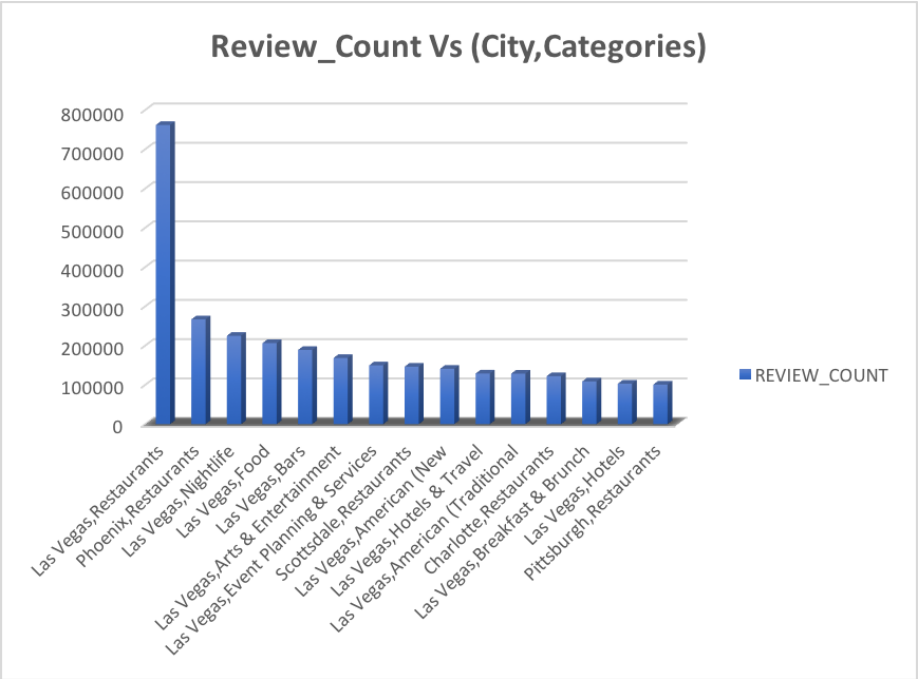
```
SET elephantbird.jsonloader.nestedLoad 'true';

A = LOAD './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, (int)yelp#'review_count' as review_count, yelp#'city' as city,
yelp#'state' as state, (float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude;

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

us_business = FILTER coordinates_business BY NOT ( (state matches '.*ON.*') OR (state matches '.*QC.*') );

businesses = FOREACH us_business GENERATE FLATTEN(categories), city, state, review_count;

grouped = GROUP businesses BY (city,categories);

result = FOREACH grouped GENERATE group, SUM(businesses.review_count);

store result into './FinalQ14';
```

Spark Script:

```
Spark1
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_business.json")
businesses=df.select(pyspark.sql.functions.explode(
df.categories).alias("category"),df.business_id,df.review_count,df.city,df.state,df.longitude,df.latitude)
businesses.registerTempTable("business")
result=sqlContext.sql("select SUM(review_count) as sum_reviews,city,category from business where latitude >
24.520833 and latitude < 49.384472 and longitude > -124.766667 and longitude < -66.950 and state!='ON' and
state!='QC' group by city,category")
result.write.mode('append').json("/user/cloudera/output_spark/Q1spark.json")
```

Analysis & Visualizations:

	CITY	CATEGORIES	REVIEW_COUNT	CITY,CATEGORIES	REVIEW_COUNT
2	Las Vegas	Restaurants	761661	Las Vegas,Restaurants	761661
3	Phoenix	Restaurants	266766	Phoenix,Restaurants	266766
4	Las Vegas	Nightlife	224955	Las Vegas,Nightlife	224955
5	Las Vegas	Food	206226	Las Vegas,Food	206226
6	Las Vegas	Bars	188813	Las Vegas,Bars	188813
7	Las Vegas	Arts & Entertainment	168321	Las Vegas,Arts & Entertainment	168321
8	Las Vegas	Event Planning & Services	149498	Las Vegas,Event Planning & Services	149498
9	Scottsdale	Restaurants	146271	Scottsdale,Restaurants	146271
10	Las Vegas	American (New	140884	Las Vegas,American (New	140884
11	Las Vegas	Hotels & Travel	128908	Las Vegas,Hotels & Travel	128908
12	Las Vegas	American (Traditional	128502	Las Vegas,American (Traditional	128502
13	Charlotte	Restaurants	122200	Charlotte,Restaurants	122200
14	Las Vegas	Breakfast & Brunch	108735	Las Vegas,Breakfast & Brunch	108735
15	Las Vegas	Hotels	102928	Las Vegas,Hotels	102928
16	Pittsburgh	Restaurants	100764	Pittsburgh,Restaurants	100764



Assumptions:

- I have considered US border latitudes (24.520833 to 49.384472) and longitudes (-124.766667 to -66.950)
- For a clear visualization, I have only taken Top 15 highest review count tuples of (city, categories) and I have visualized these using excel pivot tables

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

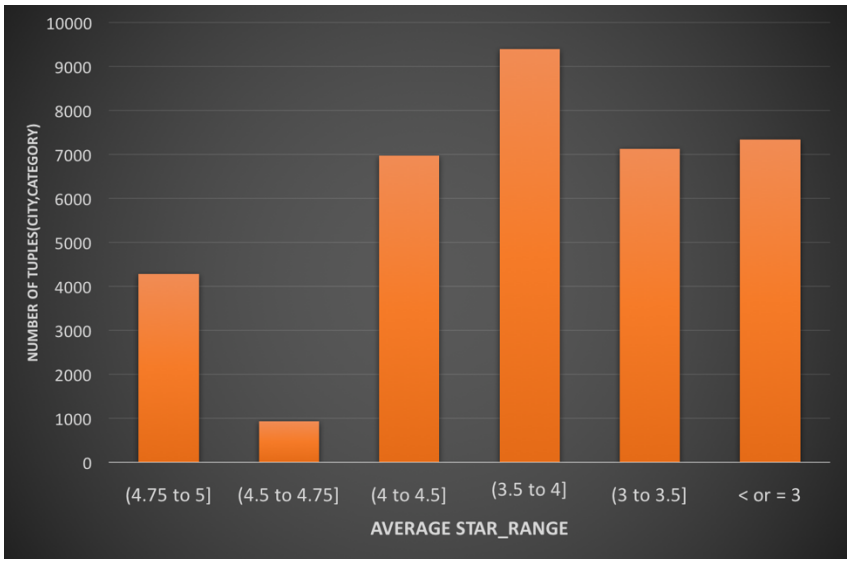
Pig Script:

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

A = LOAD './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, (int)yelp#'review_count' as review_count, yelp#'city' as city,
yelp#'state' as state, (float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude, (float)yelp#'stars' as stars;

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

us_business = FILTER coordinates_business BY NOT ( (state matches '.*ON.*') OR (state matches '.*QC.*') );

businesses = FOREACH us_business GENERATE FLATTEN(categories), city, state, stars;

grouped = GROUP businesses BY (categories,city);

result = FOREACH grouped GENERATE group, AVG(businesses.stars) as average_stars;

Final_result = ORDER result BY group DESC, average_stars DESC;

store Final_result into './Q2PIGORDERED';

|
```

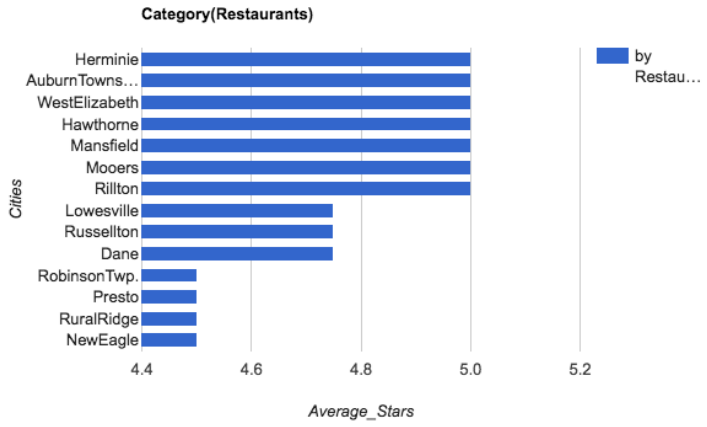
Spark Script:

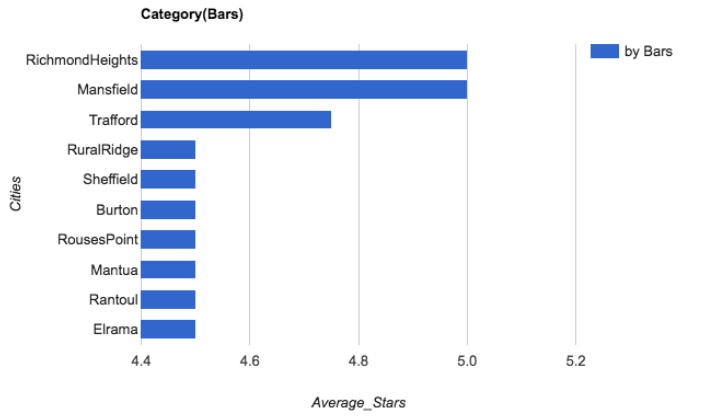
```
Spark2
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_business.json")
businesses=df.select(pyspark.sql.functions.explode(
df.categories).alias("category"),df.business_id,df.stars,df.city,df.state,df.longitude,df.latitude)
businesses.registerTempTable("business")
result=sqlContext.sql("select AVG(stars) as average_stars,city,category from business where latitude >
24.520833 and latitude < 49.384472 and longitude > -124.766667 and longitude < -66.950 and state!='ON' and
state!='QC' group by city,category order by category ASC, average_stars DESC")
result.write.mode('append').json("/user/cloudera/output_spark/Q2spark.json")
```

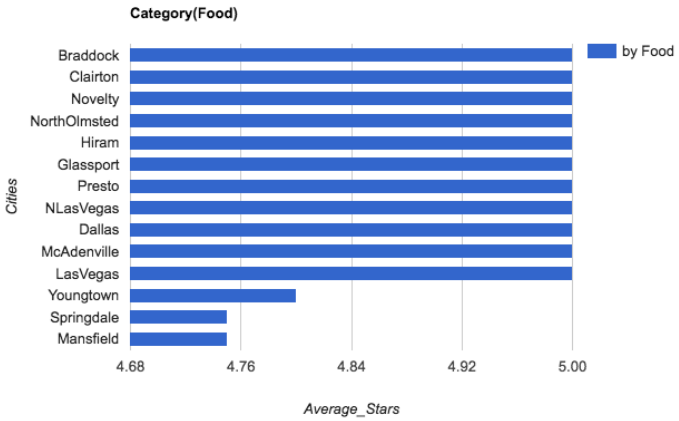
Analysis & Visualizations:

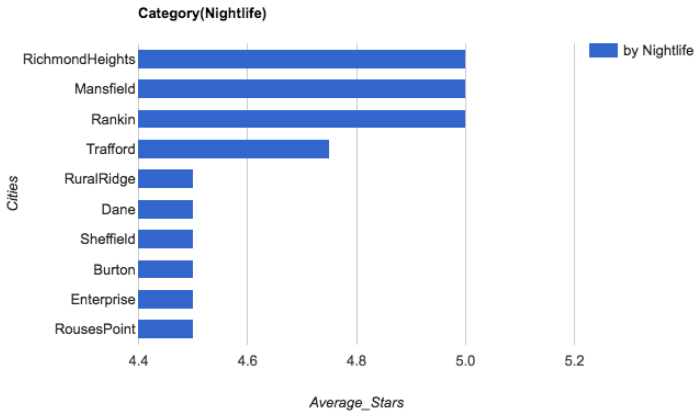


Visualization indicating number of tuples of city, category within an average star range









Assumptions:

- I have considered US border latitudes (24.520833 to 49.384472) and longitudes (-124.766667 to -66.950)

- From the question 1, after analysis i have found that the categories (like restaurants, nightlife, food, bars) have the top most review count. Considering those categories I have visualized average stars for various city having top average star rating.

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

Given Details:

Center: University of Wisconsin - Madison
Latitude: 43 04’ 30” N, Longitude: 89 25’ 2” W
Decimal Degrees: Latitude: 43.0766, Longitude: -89.4125

The bounding box for this problem is ~10 miles, which we will loosely define as 10 minutes. So the bounding box is a square box, 20 minutes long each side (of longitude and latitude), with UWM at the center.

Pig Script:

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

A = LOAD './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, (int)yelp#'review_count' as review_count, yelp#'city' as city,
yelp#'state' as state, (float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude, (float)yelp#'stars' as stars;

coordinates_business = FILTER business BY (latitude<43.221261) AND (latitude>42.931739) AND (longitude<-89.214450) AND (longitude>-89.610310);

businesses = FOREACH coordinates_business GENERATE FLATTEN(categories), city, state, stars;

grouped = GROUP businesses BY (categories, city);

result = FOREACH grouped GENERATE group, AVG(businesses.stars) as average_stars;

Final_result = ORDER result BY group DESC, average_stars DESC;

store Final_result into './FinalQ3PIGORDERED';

```

Spark Script:

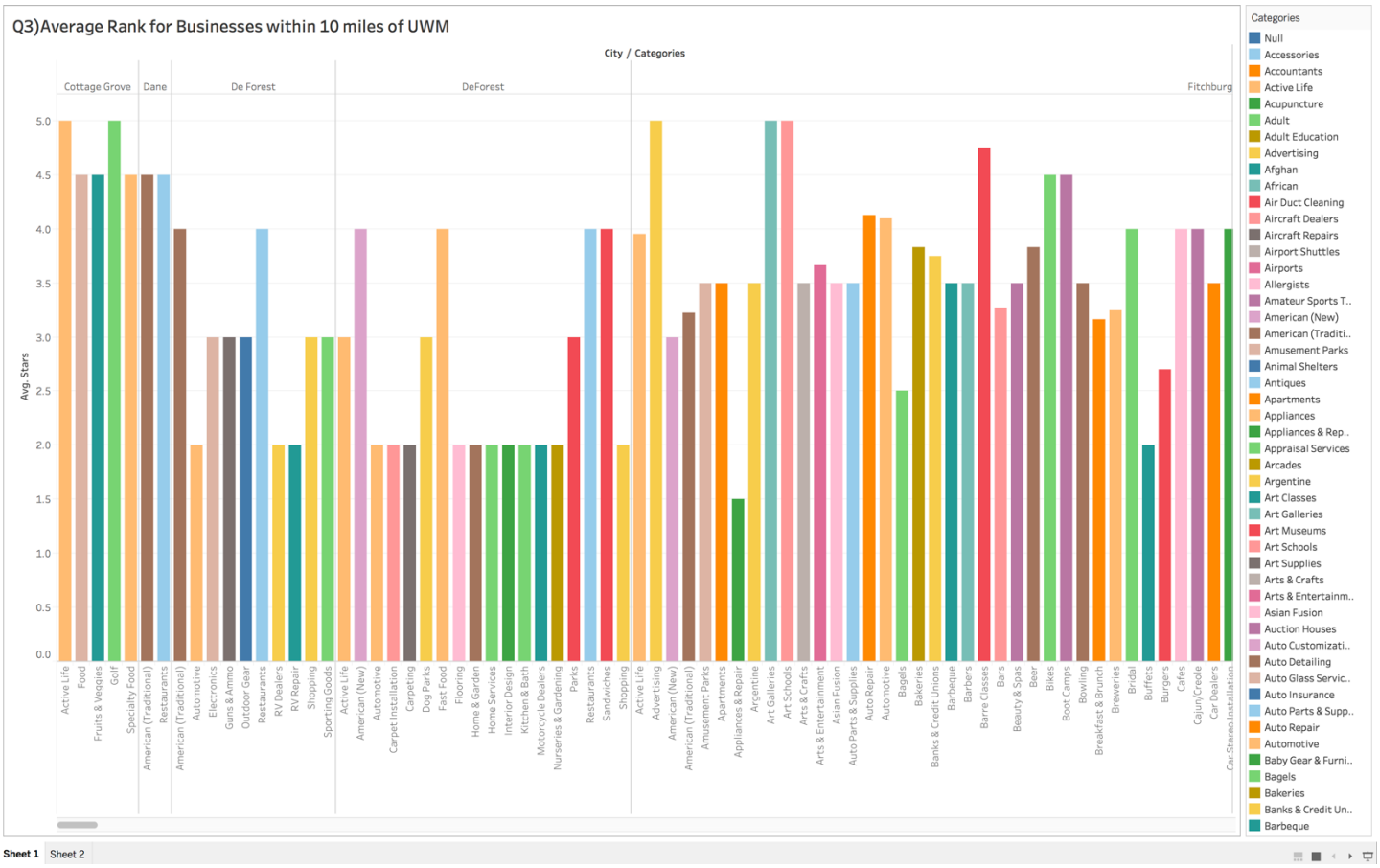
```
Spark3
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_business.json")
businesses=df.select(pyspark.sql.functions.explode(
df.categories).alias("category"),df.business_id,df.stars,df.city,df.longitude,df.latitude)
businesses.registerTempTable("business")
result=sqlContext.sql("select AVG(stars) as average_stars,city,category from business where latitude >
42.931739 and latitude < 43.221261 and longitude > -89.610310 and longitude < -89.214450 group by
city,category order by category ASC, average_stars DESC")
result.write.mode('append').json("/user/cloudera/output_spark/Q3spark.json")
```

Assumptions:

- I have considered 10 miles from center of UWM with latitudes (42.931739 to 43.221261) and longitudes (-89.610310 to -89.214450)

Analysis & Visualizations:

In the Visualization, I have shown Average Star rating for various business categories shown in colors of bar chart within different cities.



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

Pig Script:

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

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

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

B = LOAD './yelp_academic_dataset_review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true') AS (yelp2: map[]);

review = FOREACH B GENERATE yelp2#'user_id' as user_id1, yelp2#'business_id' as business_id, (int)yelp2#'stars' as stars;

C = LOAD './yelp_academic_dataset_user.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad=true') AS (yelp3: map[]);

user = FOREACH C GENERATE yelp3#'user_id' as user_id, (int)yelp3#'review_count' as review_count;

ordered_C = ORDER user BY review_count DESC;

top_10_C = LIMIT ordered_C 10;

top_10_stars_joined_BC = JOIN top_10_C BY user_id, review BY user_id1;

top_10_categories_joined_ABC = JOIN business BY business_id, top_10_stars_joined_BC BY business_id;

top_10_categories_ABC = FOREACH top_10_categories_joined_ABC GENERATE FLATTEN(categories), stars, user_id;

grouped = GROUP top_10_categories_ABC BY (user_id, categories);

grouped_result = FOREACH grouped GENERATE group, AVG(top_10_categories_ABC.stars) as avg_stars;

ordered_result = ORDER grouped_result BY user_id;

STORE grouped_result INTO './FinalQ4PIG';

```

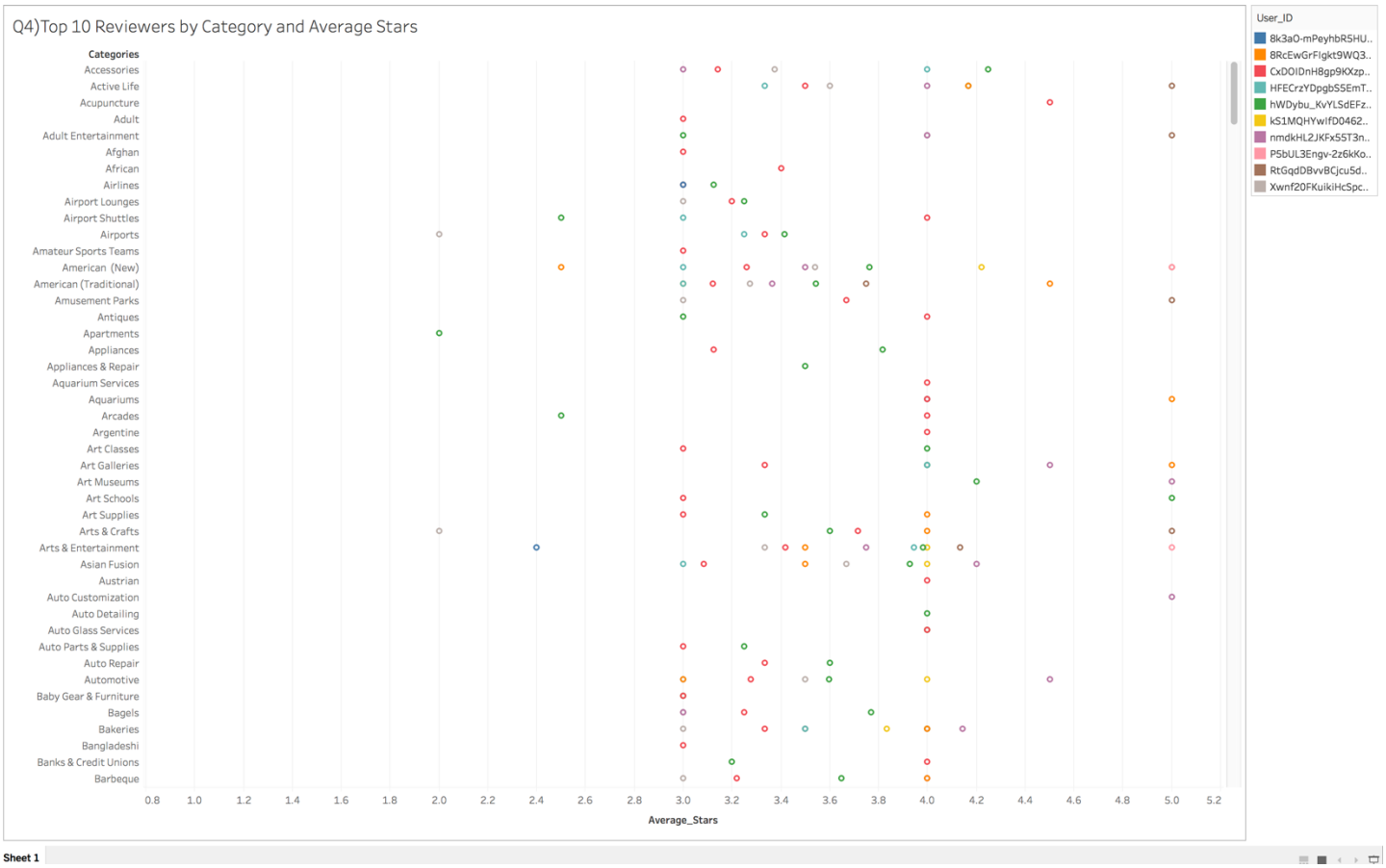
Spark Script:

Spark4

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df_business = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_business.json")
df_review = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_review.json")
df_user = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_user.json")
df_review.registerTempTable("reviews")
top10=df_user.sort(df_user.review_count.desc())
top10=top10.limit(10)
top10.registerTempTable("top10")
businesses=df_business.select(pyspark.sql.functions.explode(
df_business.categories).alias("category"),df_business.business_id,df_business.stars,df_business.city)
businesses.registerTempTable("business")
result=sqlContext.sql("select AVG(reviews.stars) as average_stars,reviews.user_id,category from
business,reviews,top10 where top10.user_id=reviews.user_id and reviews.business_id=business.business_id
group by reviews.user_id,category order by reviews.user_id ASC")
result.write.mode('append').json("/user/cloudera/output_spark2/Q4spark.json")

```

Analysis & Visualizations:

In the Visualization, I have shown Average Star rating for Top 10 Users with high review count, I have shown them in colors of scatter plot within different cities.



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

Pig Script:

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

A = LOAD './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, (int)yelp#'review_count' as review_count, yelp#'city' as city,
yelp#'state' as state, (float)yelp#'latitude' as latitude, (float)yelp#'longitude' as longitude, (float)yelp#'stars' as stars;

coordinates_business = FILTER business BY (latitude<43.221261) AND (latitude>42.931739) AND (longitude<-89.214450) AND
(longitude>-89.610310);

joined = JOIN coordinates_business by business_id, review by business_id;

businesses = FOREACH joined GENERATE FLATTEN(categories), coordinates_business::stars, coordinates_business::date;

filter_food = FILTER businesses BY categories MATCHES '.*(Food|food).*';

ordered_1 = ORDER filter_food BY stars DESC;

top_ten = LIMIT ordered_1 10;

ordered_2 = ORDER filter_food BY stars ASC;

bottom_ten = LIMIT ordered_2 10;

top_bottom_20 = UNION top_ten, bottom_ten;

top_bottom = FOREACH top_bottom_20 GENERATE business.date, stars, categories;

joined2 = FOREACH top_bottom GENERATE SUBSTRING(date,5,7) as month, stars;

filter_food2 = FILTER joined2 BY (month matches '01|02|03|04|05|06');

result_ordered = ORDER filter_food2 BY stars DESC;

store result_ordered into './FinalQ57';
```

Spark Script:

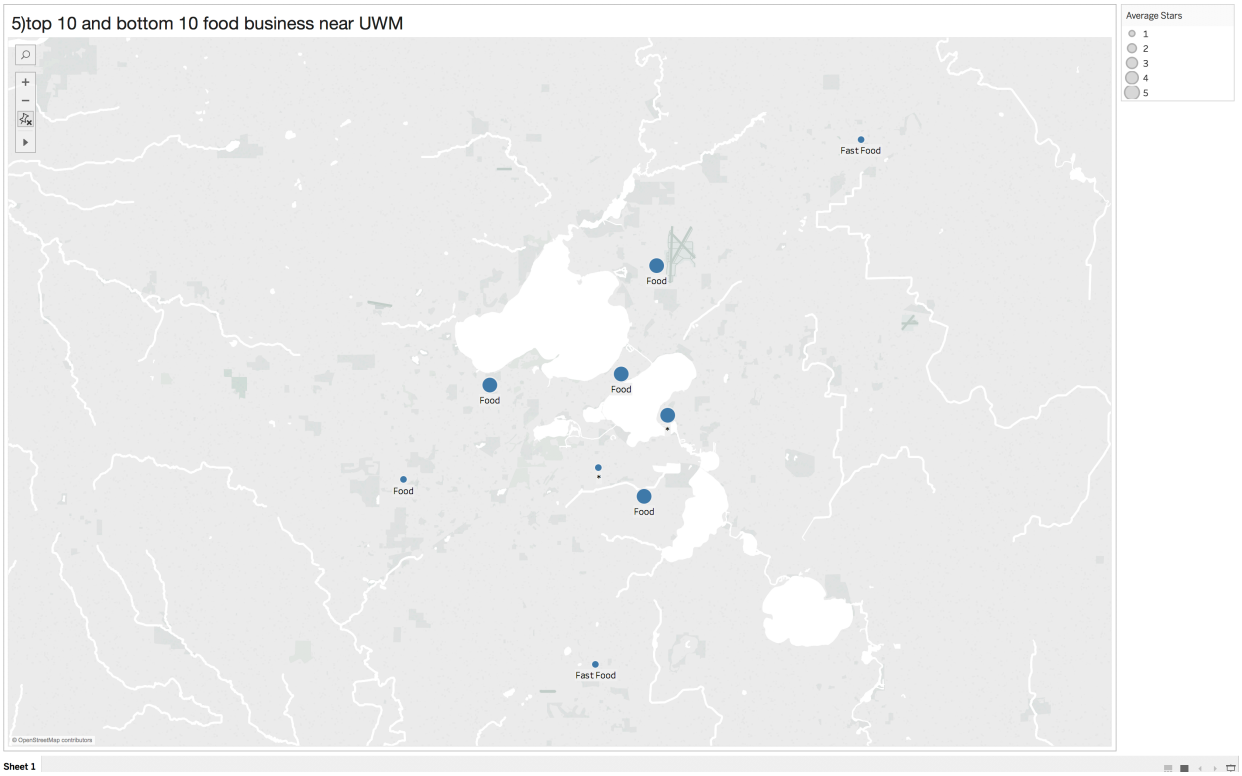
Q5PigCode | Q5SparkCode

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df_business = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_business.json")
df_review = sqlContext.read.json("/user/cloudera/yelp_academic_dataset_review.json")
df_business.registerTempTable("business")
df_review.registerTempTable("reviews")
area=sqlContext.sql("select business_id,stars from business where latitude > 42.931739 and latitude < 43.221261 and longitude > -89.610310 and
longitude < -89.214450")
top10=area.sort(area.stars.desc())
top10=top10.limit(10)
top10.registerTempTable("top10")
bottom10=area.sort(area.stars.asc())
bottom10=bottom10.limit(10)
bottom10.registerTempTable("bottom10")
reviewmonth=df_review.select(pyspark.sql.functions.year(df_review.date).alias('year'),pyspark.sql.functions.month(
df_review.date).alias('month'),df_review.business_id,df_review.stars)
reviewmonth.registerTempTable("reviewmonth")
resulttop10=sqlContext.sql("select AVG(reviewmonth.stars) as average_stars,reviewmonth.business_id,reviewmonth.year,reviewmonth.month from
reviewmonth,top10 where reviewmonth.business_id=top10.business_id group by reviewmonth.business_id,year,month")
resultbottom10=sqlContext.sql("select AVG(reviewmonth.stars) as average_stars,reviewmonth.business_id,reviewmonth.year,reviewmonth.month from
reviewmonth,bottom10 where reviewmonth.business_id=bottom10.business_id group by reviewmonth.business_id,year,month")
reviewmonth.business_id=sqlContext.sql(SELECT  FROM Customers
WHERE City LIKE '%es%';
resulttop10.write.mode('append').json("/user/cloudera/output_spark2/Q5_top10_spark.json")
resultbottom10.write.mode('append').json("/user/cloudera/output_spark/Q5_bottom10_spark.json")
```

Assumptions:

- I have considered 10 miles from center of UWM with latitudes (42.931739 to 43.221261) and longitudes (-89.610310 to -89.214450)

Analysis & Visualizations:



	A	B	C	D	E	F	G
1	Food Catego	City	State	Average_Sta	Longitude	Latitude	ZipCode
2	Fast Food	Sun Prairie	WI	1	-89.2315	43.1866	53590
3	Fast Food	Oregon	WI	1	-89.38384	42.9422	53575
4	Food	Madison	WI	1	-89.50551	43.05725	53719
5	Fast Food	Sun Prairie	WI	1	-89.229965	43.187935	53590
6	Food	Madison	WI	1	-89.32394	43.118195	53704
7	Fast Food	Madison	WI	1	-89.4074	43.0315	53713
8	Food	Madison	WI	1	-89.4074	43.0315	53713
9	Fast Food	Sun Prairie	WI	1	-89.27047	43.16638	53590
10	Fast Food	Madison	WI	1	-89.39466	43.039513	53713
11	Fast Food	Fitchburg	WI	1	-89.40951	43.029854	53713
12	Food	Madison	WI	5	-89.3812	43.074276	53703
13	Food	Monona	WI	5	-89.32558	43.07011	53716
14	Food Trucks	Monona	WI	5	-89.31221	43.047745	53716
15	Food	Monona	WI	5	-89.31221	43.047745	53716
16	Food	Madison	WI	5	-89.42924	43.074207	53705
17	Food	Madison	WI	5	-89.442566	43.075603	53705
18	Food	Fitchburg	WI	5	-89.47182	43.021145	53711
19	Food	Madison	WI	5	-89.34364	43.102917	53704
20	Food	Madison	WI	5	-89.47064	43.078316	53705
21							