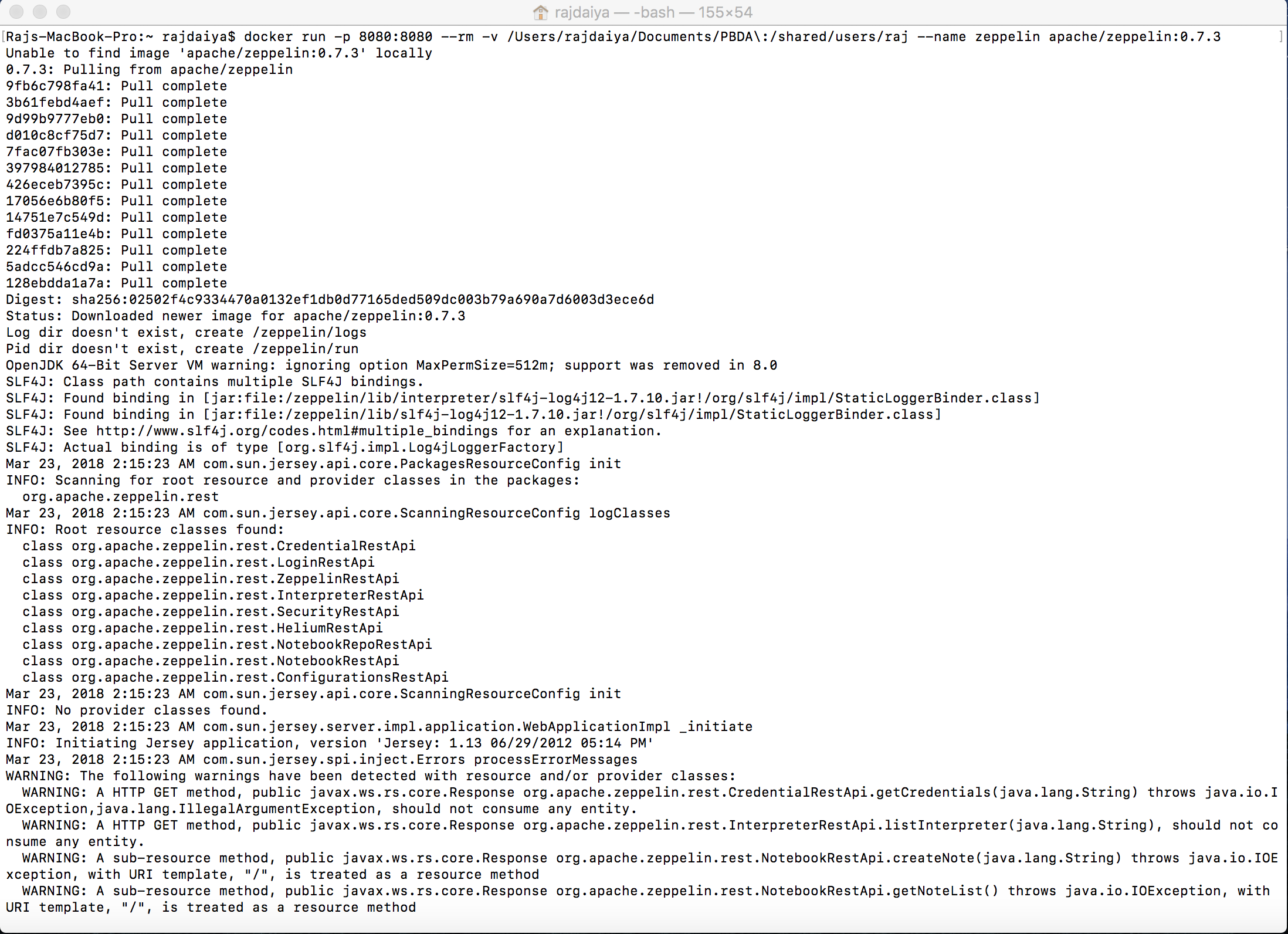
**ZEPPELIN INSTALLATION**



**PIG QUERIES (Pig Output in solution/ans.tsv)**

register '/shared/users/raj/piggybank-0.11.0.jar';

register '/shared/users/raj/elephant-bird-core-4.15.jar';

register '/shared/users/raj/elephant-bird-hadoop-compat-4.15.jar';

register '/shared/users/raj/elephant-bird-pig-4.15.jar';

register '/shared/users/raj/json-simple-1.1.1.jar';

**Query 1: Summarize the number of unique *reviewers* by US city, by business category. That is, count the unique reviewers by city, by business.**

businesses = LOAD '/shared/users/raj/dataset/business.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

tip = LOAD '/shared/users/raj/dataset/tip.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

uscities = LOAD '/shared/users/raj/dataset/uscities.csv' USING PigStorage(',') AS (a:chararray, b:chararray, c:chararray, d:chararray, e:chararray, f:chararray,

g:chararray, h:chararray, i:chararray, j:chararray, k:chararray, l:chararray);

guscities = FOREACH(GROUP uscities BY (c)) GENERATE FLATTEN(group) AS st;

businesses = FOREACH businesses GENERATE json#'business\_id' as business\_id,json#'name' as name,json#'neighborhood' as neighborhood,json#'address' as address,json#'city' as city,json#'state' as state,json#'postal\_code' as postal\_code,json#'latitude' as latitude,json#'longitude' as longitude,json#'stars' as stars,json#'review\_count' as review\_count,json#'is\_open' as is\_open,json#'attributes' as attributes,json#'categories' as categories,json#'hours' as hours,json#'type' as type;

businesses\_uscities = JOIN guscities by st LEFT OUTER, businesses BY state;

attributes = FOREACH businesses\_uscities GENERATE (int)review\_count AS review\_count, city AS city, FLATTEN(categories) as categories;

groupcitycat = GROUP attributes BY (city, categories);

reviewcounts = FOREACH groupcitycat GENERATE FLATTEN(group) as (city, categories), COUNT(attributes.review\_count) as total\_count;

orderreviewcount = ORDER reviewcounts BY city;

STORE orderreviewcount INTO '/shared/users/raj/solution/ans1.tsv';

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

businesses = LOAD '/shared/users/raj/dataset/business.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

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

groupcitycat = GROUP attributes BY (city,categories);

avgstars = FOREACH groupcitycat GENERATE AVG(attributes.stars) as st, FLATTEN(group) as (city,categories);

result = RANK avgstars by categories ASC, st DESC;

STORE result INTO '/shared/users/raj/solution/ans2.tsv';

**Query 3: What is the average rank (# stars) for businesses within 15 km of Edinburgh Castle, Scotland, by type of business (category)?**

businesses = LOAD '/shared/users/raj/dataset/business.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

business\_distance= FOREACH businesses GENERATE FLATTEN(json#'categories') as categories,(double)json#'stars' as stars, json#'business\_id' AS business\_id, json#'name' AS name, json#'city' AS city, json#'latitude' AS latitude, json#'longitude' AS longitude,ACOS(SIN(55.9469753\*3.14159/180)\*SIN((json#'latitude')\*3.14159/180)+COS(55.9469753\*3.14159/180)\*COS((json#'latitude')\*3.14159/180)\*COS(-3.2096308\*3.14159/180 - (json#'longitude')\*3.14159/180))\*6371 as distance;

business\_distance= FILTER business\_distance BY distance<15;

filteredDataForQ5 = business\_distance;

groupedData = GROUP business\_distance BY categories;

finalData = FOREACH groupedData GENERATE group as category, AVG(business\_distance.stars);

STORE finalData INTO '/shared/users/raj/solution/ans3.tsv';

**Query 4: Rank reviewers in Q3 by their number of reviews. For the top 10 reviewers, show their average number of stars, by category.**

businesses = LOAD '/shared/users/raj/dataset/business.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

business\_distance= FOREACH businesses GENERATE FLATTEN(json#'categories') as categories,(double)json#'stars' as stars, json#'business\_id' AS business\_id, json#'name' AS name, json#'city' AS city, json#'latitude' AS latitude, json#'longitude' AS longitude,**ACOS(SIN(55.9469753\*3.14159/180)\*SIN((json#'latitude')\*3.14159/180)+COS(55.9469753\*3.14159/180)\*COS((json#'latitude')\*3.14159/180)\*COS(-3.2096308\*3.14159/180 - (json#'longitude')\*3.14159/180))\*6371** as distance;

business\_distance= FILTER business\_distance BY distance<=15;

filteredDataForQ5 = business\_distance;

data\_user = LOAD '/shared/users/raj/dataset/user.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (json:map[]);

data\_review = LOAD '/shared/users/raj/dataset/review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (json:map[]);

at\_user = FOREACH data\_user GENERATE json#'user\_id' AS user\_id, (int)json#'review\_count' as review\_count;

at\_review = FOREACH data\_review GENERATE json#'user\_id' AS user\_id, json#'review\_id' as review\_id, json#'business\_id' as business\_id,(float)json#'stars' AS stars;

rank\_users = ORDER at\_user BY review\_count DESC;

rank\_limit = LIMIT rank\_users 10;

rev\_bus = JOIN at\_review BY business\_id, filteredDataForQ5 BY business\_id;

total\_combine = JOIN rank\_limit BY user\_id, rev\_bus BY at\_review::user\_id;

get\_reqd = FOREACH total\_combine GENERATE at\_review::user\_id, filteredDataForQ5::categories, at\_review::star;

grouping = GROUP get\_reqd by (at\_review::user\_id, filteredDataForQ5::categories);

avg\_stars = FOREACH grouping GENERATE FLATTEN(group) as (usr, cat), AVG(get\_reqd.star) as star;

STORE avg\_stars INTO '/shared/users/raj/solution/ans4.tsv';

**Query 5: For the top 10 and bottom 10 category *Food* businesses in Q3, (in terms of stars), summarize star rating for reviews in January through May only.**

businesses = LOAD '/shared/users/raj/dataset/business.json' using com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') as json:map[];

business\_distance= FOREACH businesses GENERATE FLATTEN(json#'categories') as categories,(double)json#'stars' as stars, json#'business\_id' AS business\_id, json#'name' AS name, json#'city' AS city, json#'latitude' AS latitude, json#'longitude' AS longitude,ACOS(SIN(55.9469753\*3.14159/180)\*SIN((json#'latitude')\*3.14159/180)+COS(55.9469753\*3.14159/180)\*COS((json#'latitude')\*3.14159/180)\*COS(-3.2096308\*3.14159/180 - (json#'longitude')\*3.14159/180))\*6371 as distance;

business\_distance= FILTER business\_distance BY distance<=15;

filteredDataForQ5 = business\_distance;

reviewdata = LOAD '/shared/users/raj/dataset/review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (json:map[]);

filterlatlong = filter filteredDataForQ5 by categories == 'Food';

topfilter = ORDER filterlatlong BY stars DESC;

bottomfilter = ORDER filterlatlong BY stars ASC;

topele = LIMIT topfilter 10;

bottomele = LIMIT bottomfilter 10;

topbottomunion = UNION topele, bottomele;

rev\_attributes = FOREACH reviewdata GENERATE json#'business\_id' AS business\_id, (datetime)json#'date' AS date, json#'review\_id' AS review\_id, (float)json#'stars' AS stars, json#'user\_id' AS user\_id;

monthfilter = FILTER rev\_attributes BY (GetMonth(date)==1) OR (GetMonth(date)==2) OR

(GetMonth(date)==3) OR (GetMonth(date)==4) OR (GetMonth(date)==5);

combined = JOIN topbottomunion BY business\_id, monthfilter BY business\_id;

comb\_result = FOREACH combined GENERATE monthfilter::stars, monthfilter::business\_id,

monthfilter::review\_id;

res = GROUP comb\_result by monthfilter::business\_id;

final = FOREACH res GENERATE group as bus\_id, AVG(comb\_result.stars) as star;

STORE final INTO '/shared/users/raj/solution/ans5.tsv';

**SCALA QUERIES IN APACHE SPARK**

1. import scala.collection.mutable.WrappedArray

            import spark.implicits.\_

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

      val business = spark.read.json("/Users/Raj/Downloads/dataset/business.json")

      val ans1 = business.withColumn("category", explode(

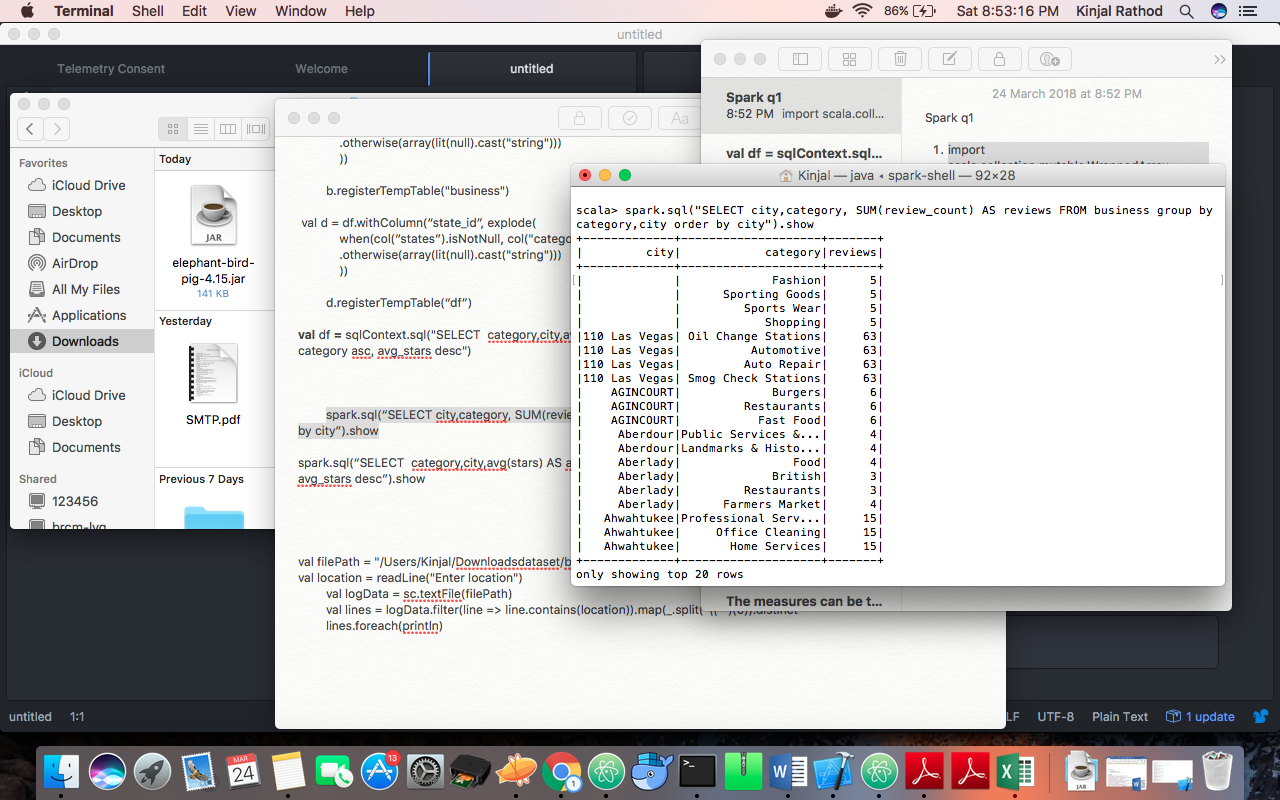
      when(col("categories").isNotNull,col("categories"))

   .otherwise(array(lit(null).cast("string")))

   ))

      ans1.registerTempTable("business")

      spark.sql("SELECT city,category, SUM(review\_count) AS total\_review FROM business group by category,city order by city").show



1. import scala.collection.mutable.WrappedArray

import spark.implicits.\_

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

val business = spark.read.json("/Users/Raj/Downloads/dataset/business.json")

val b = business.withColumn("category", explode(

   when(col("categories").isNotNull, col("categories"))

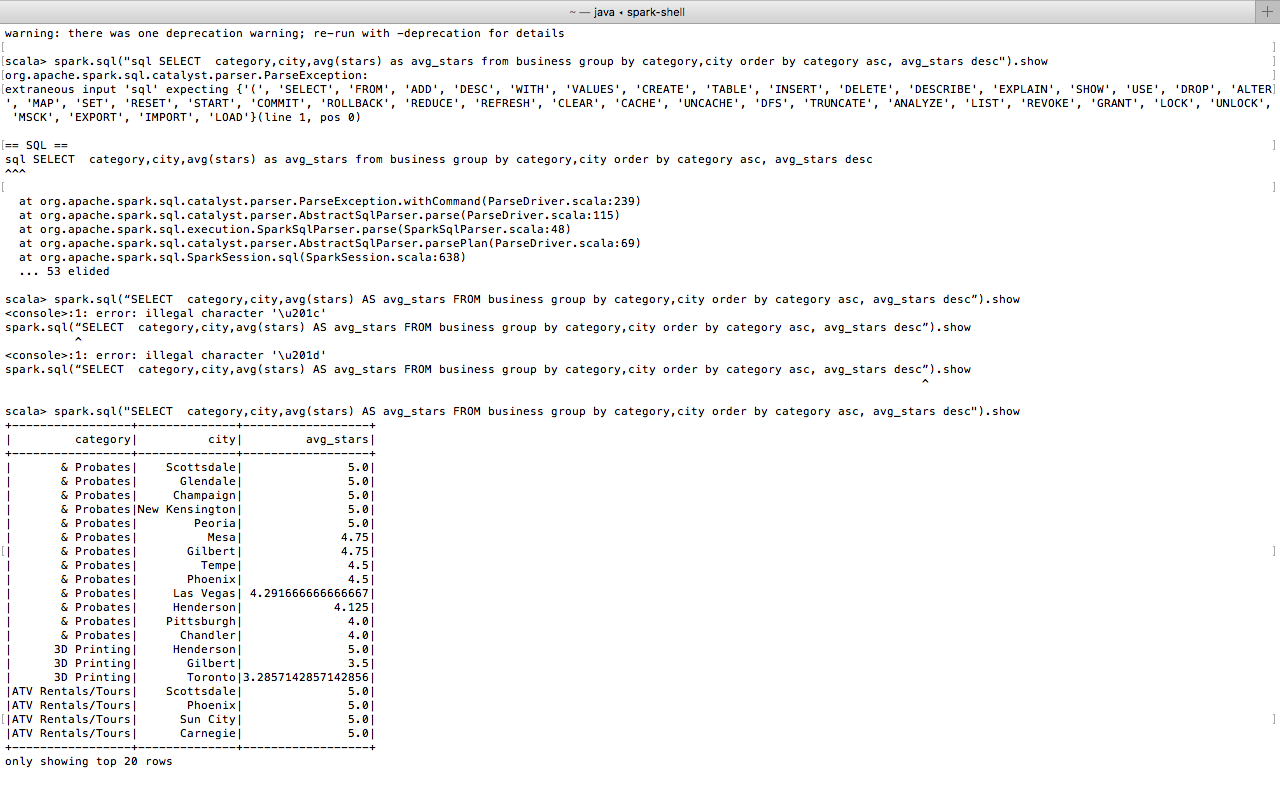
   .otherwise(array(lit(null).cast("string")))

   ))

b.registerTempTable("business")

val df = sqlContext.sql("SELECT  category,city,avg(stars) as avg\_stars from business group by category,city order by category asc, avg\_stars desc")

df.write.csv("/Users/Raj/Downloads/dataset/ans2.csv")



  3. import scala.collection.mutable.WrappedArray

            import spark.implicits.\_

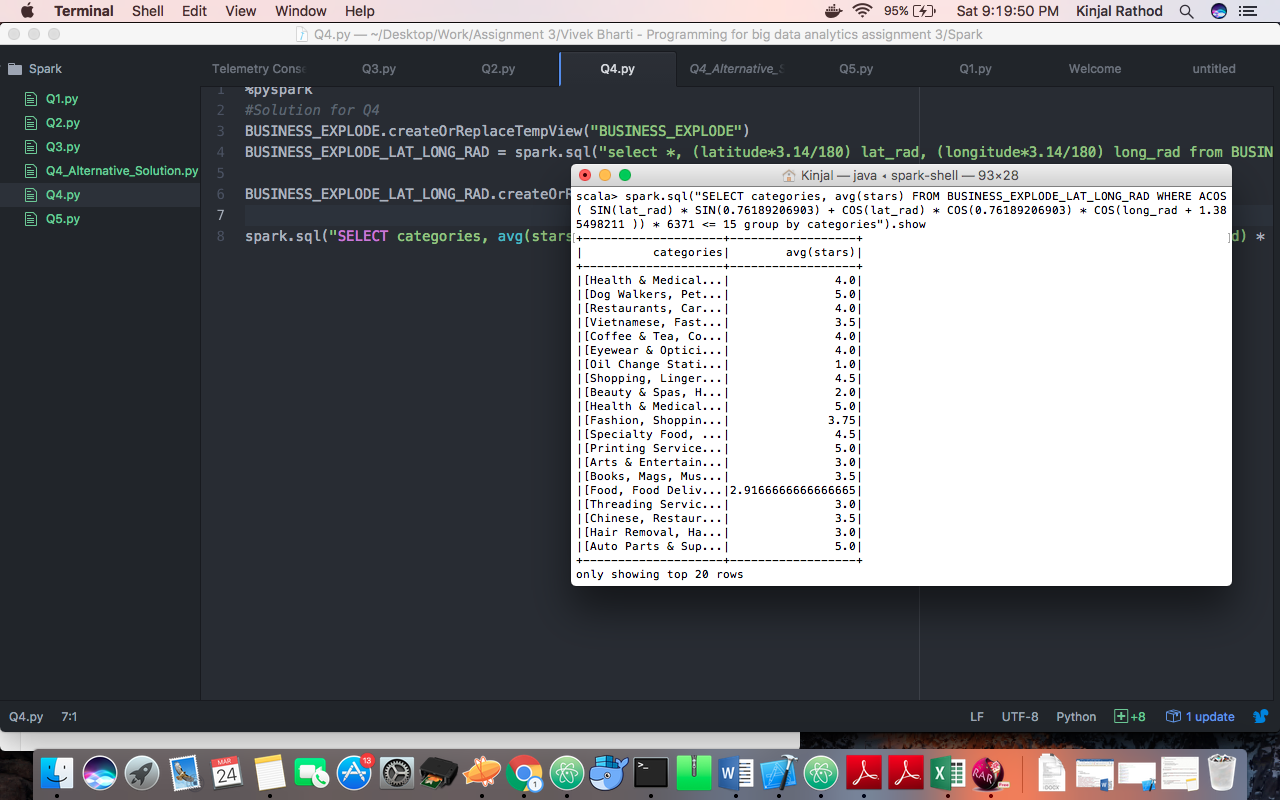
      import org.apache.spark.sql.functions.\_

      val business = spark.read.json("/Users/Raj/Downloads/dataset/business.json")

val lat\_long\_business = spark.sql("select \*, (latitude\*3.14/180) lat\_rad, (longitude\*3.14/180) long\_rad from business")

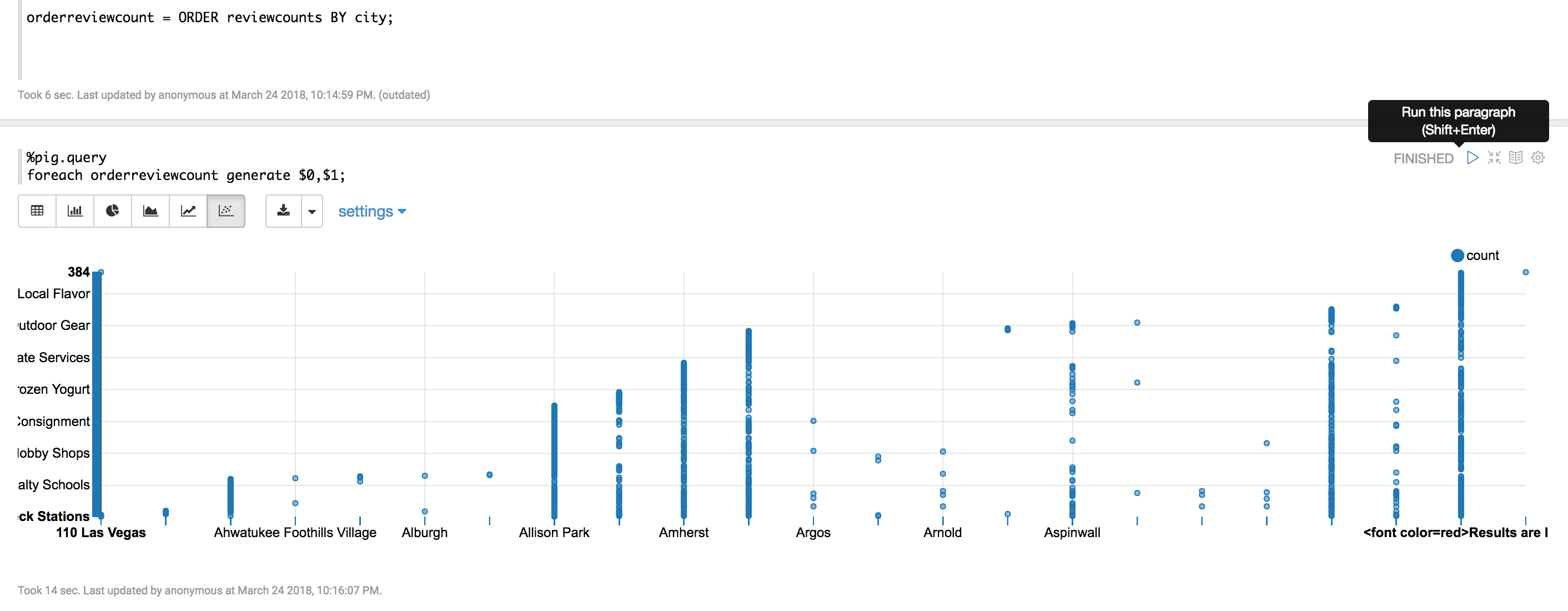
lat\_long\_business.createOrReplaceTempView("lat\_long\_business")

spark.sql("SELECT categories, avg(stars), avg(review\_count) FROM lat\_long\_business WHERE ACOS( SIN(lat\_rad) \* SIN(0.76189206903) + COS(lat\_rad) \* COS(0.76189206903) \* COS(long\_rad + 1.385498211 )) \* 6371 <= 15 group by categories").show

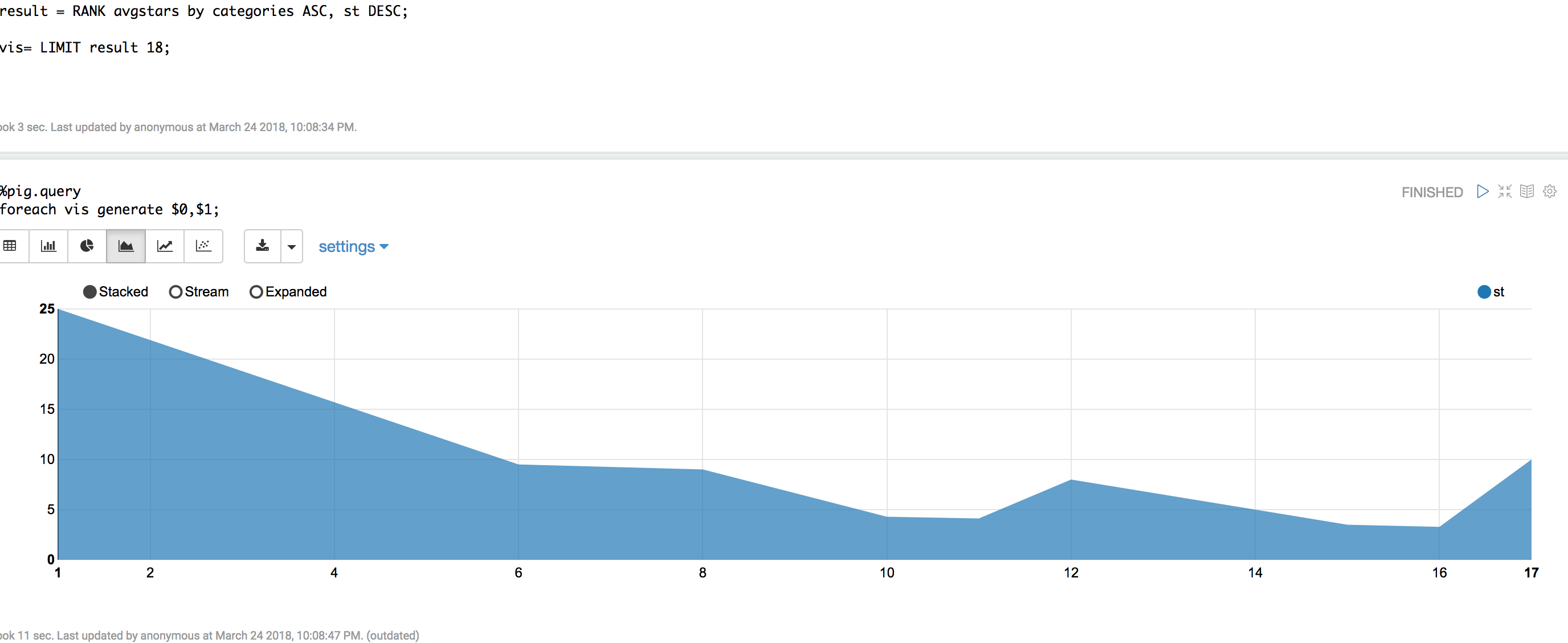


**VISUALIZATION of Pig Scripts using Zeppelin**

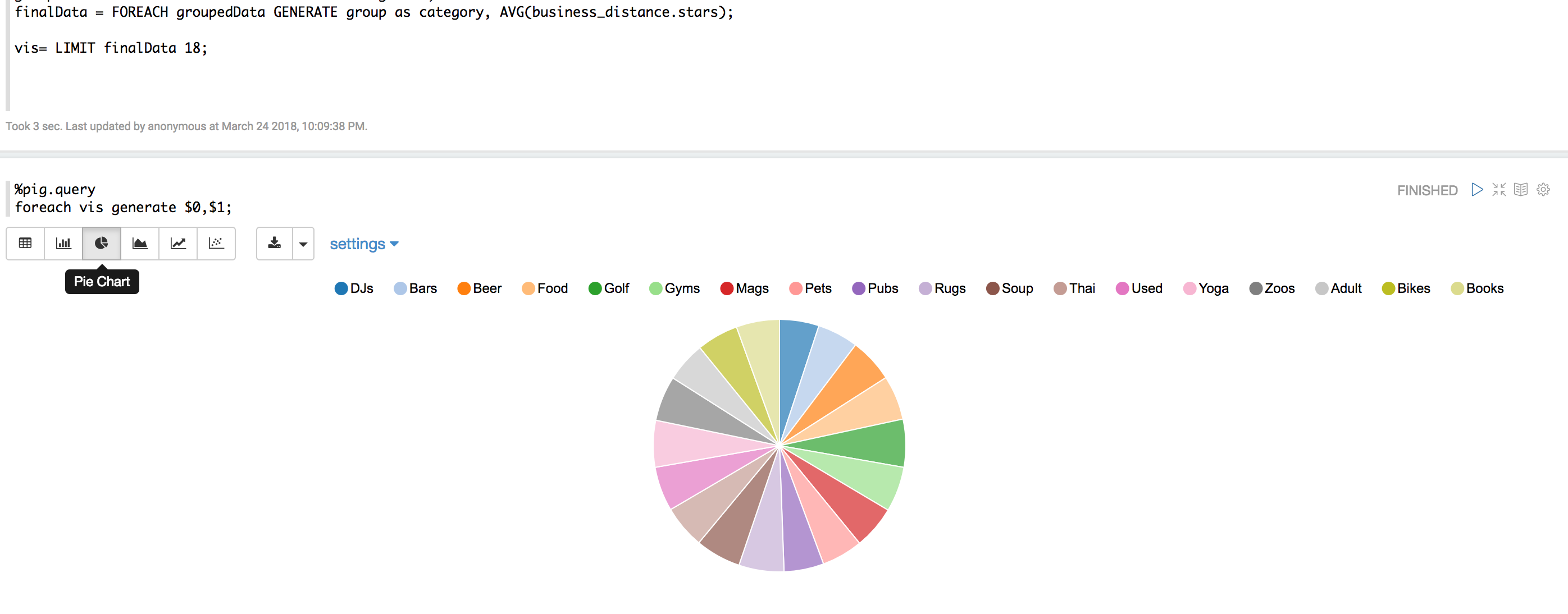
Query 1:



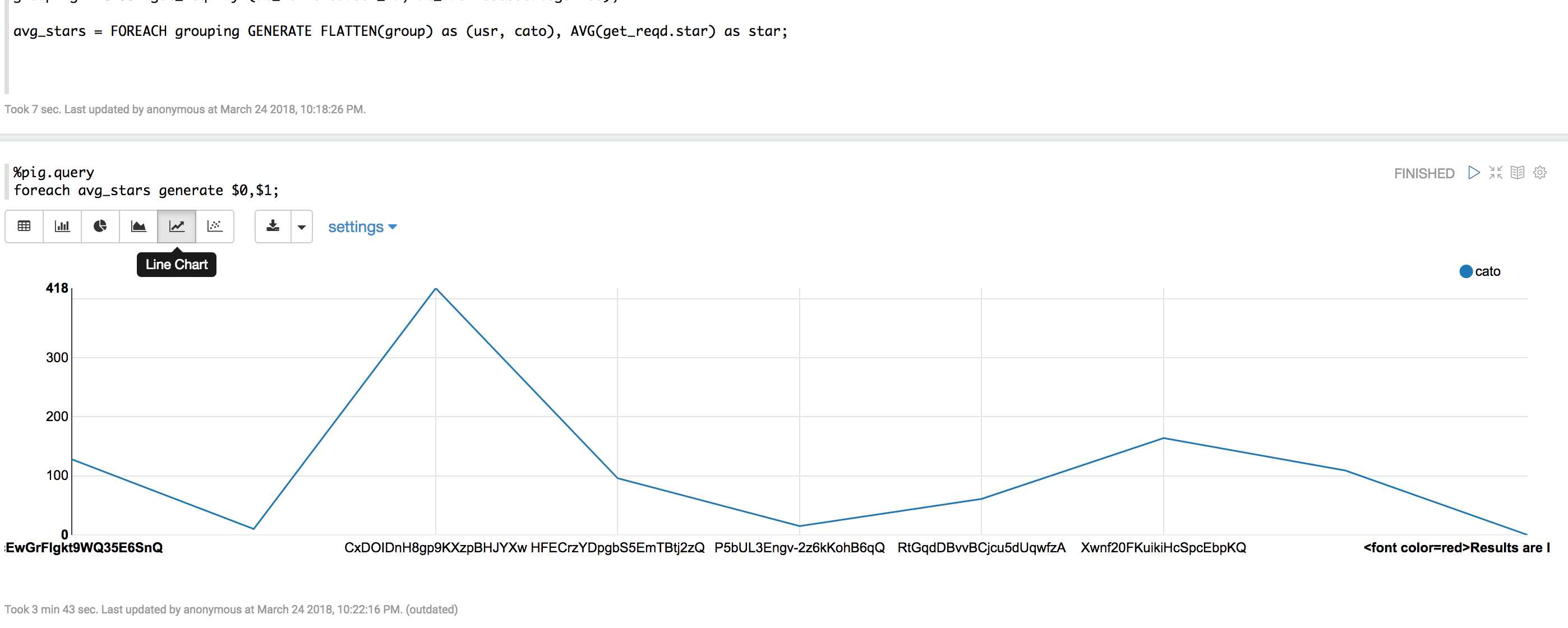
Query 2:



Query 3:



Query 4:



Query 5:

