**Programming for Big Data Analytics**

**Homework 3**

**Submitted By: Harsh Yadav**

**Net Id: hy1217**

**Question1:**

**Spark Code:**

%spark.pyspark

import pyspark

%pyspark

df=spark.read.json("/shared/d/business.json")

**#First method, by not exploding the categories**

df1 = df.groupBy("city","categories").agg(avg("review\_count"), avg("stars"))

df1.show()

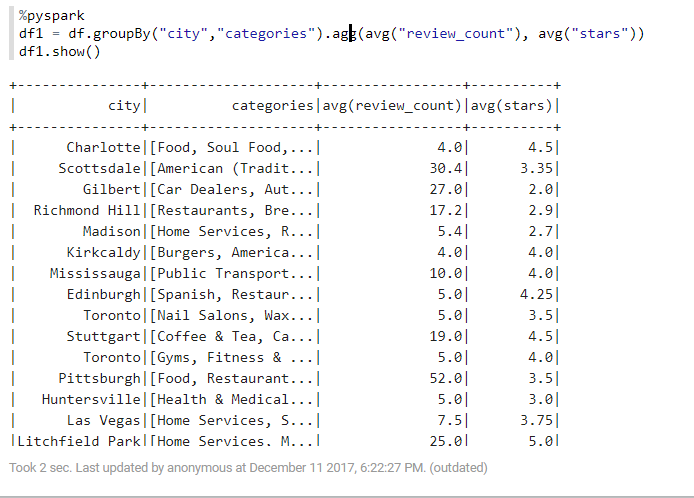
**#Second method, by exploding categories and then grouping**

a = df.withColumn("category", explode(col("categories")))

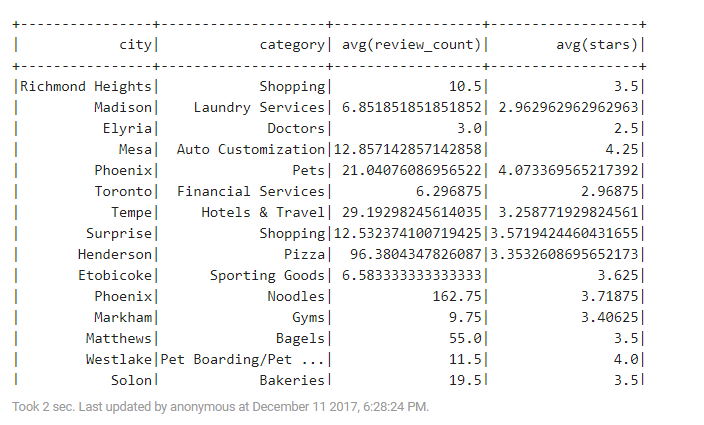
a.groupBy("city","category").agg(avg("review\_count"), avg("stars")).show()

**Spark Output:**

**Without Exploding the categories:**



**Exploding categories into distinct values:**



**Pig Code:**

register elephant-bird-hadoop-compat-4.1.jar

register elephant-bird-pig-4.1.jar

register json-simple-1.1.1.jar

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

B = FOREACH A GENERATE (int)json#'review\_count' AS review\_count, (float)json#'stars' AS stars, json#'city' as city, FLATTEN(json#'categories') AS categories;

C = GROUP B BY (city,categories);

D = FOREACH C GENERATE group.city as city, group.categories as category,AVG(B.review\_count) AS reviewCount, AVG(B.stars) AS stars;

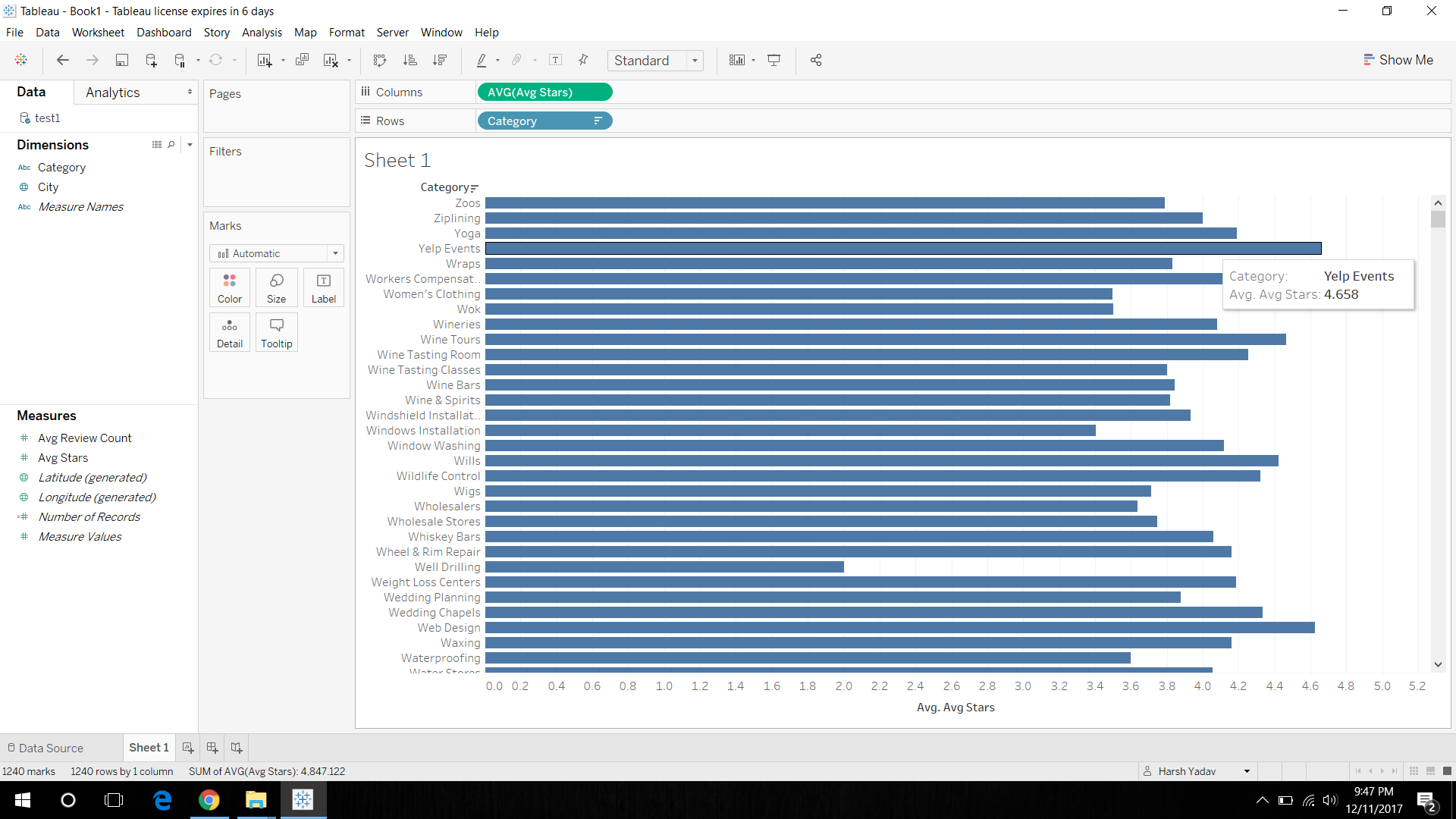
STORE D INTO './answer1.out';

**Pig Output:**

Output file has been attached in the zip folder named Pig Output.

**Visualization:**

Categories by average number of stars:



**Question2:**

**Spark Code:**

%spark.pyspark

import pyspark

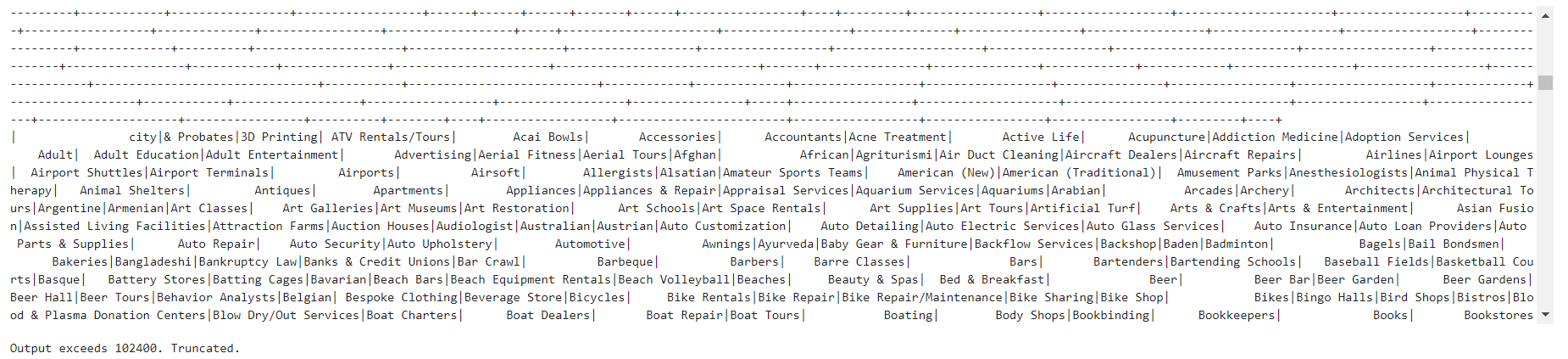
%pyspark

df=spark.read.json("/shared/d/business.json")

a = df.withColumn("category", explode(col("categories")))

a.groupBy("city").pivot("category").avg("stars").show()

**Spark Output:**



**Pig Code:**

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

B = FOREACH A GENERATE (int)json#'review\_count' AS review\_count, (float)json#'stars' AS stars, json#'city' as city, FLATTEN(json#'categories') AS categories;

C = GROUP B BY city;

D = FOREACH C GENERATE group, (B.categories) AS categories, AVG(B.stars) AS stars;

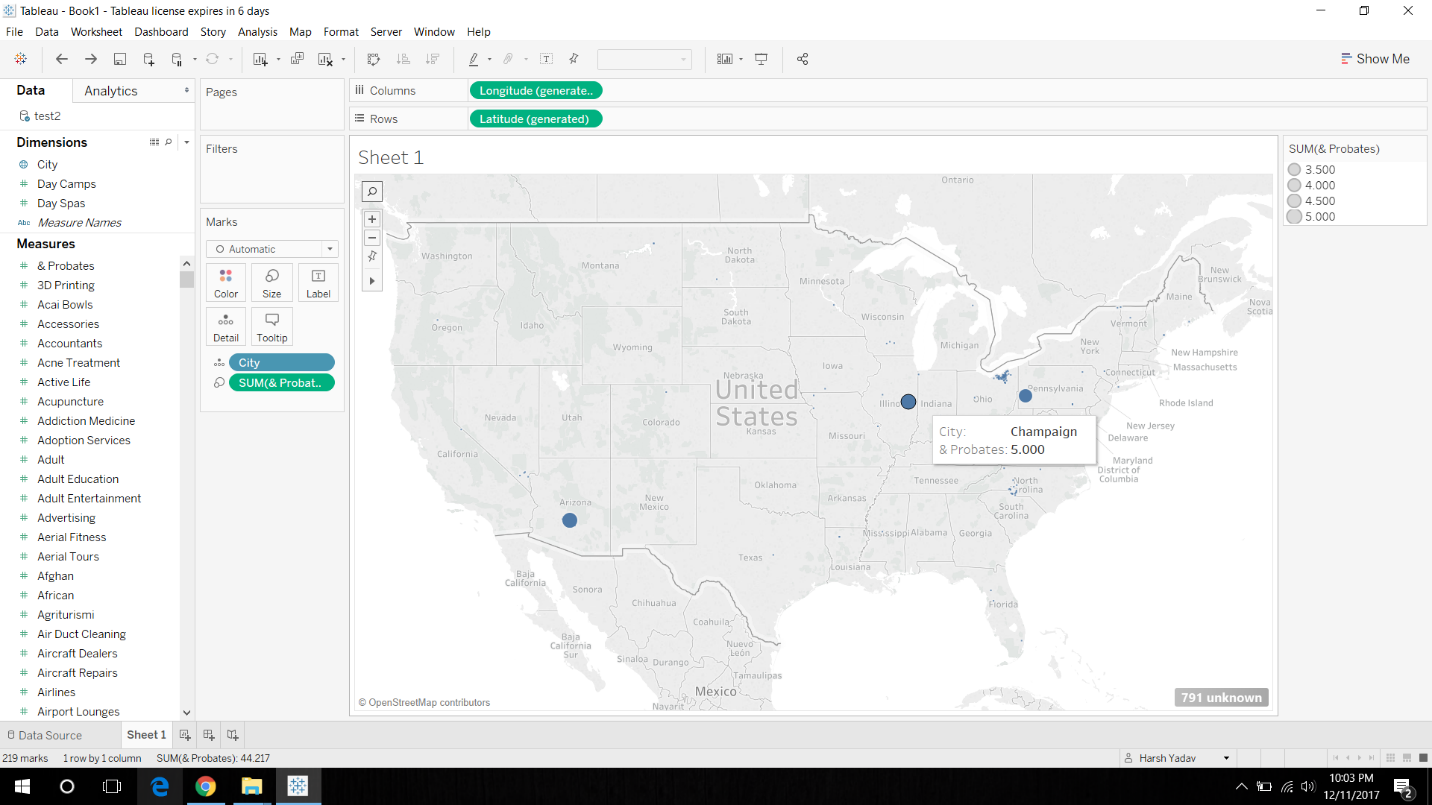
STORE D INTO './answer2.out';

**Pig Output:**

Output file is stored in the Zip Folder.

**Visualization:**

Representation of one category in different parts of US.



From this visualization we can observe that most of the categories are not present all over the US, but only at some place.

**Question 3:**

**Spark Code:**

%spark.pyspark

import pyspark

%pyspark

df=spark.read.json("/shared/d/business.json")

from pyspark.sql.functions import split, explode

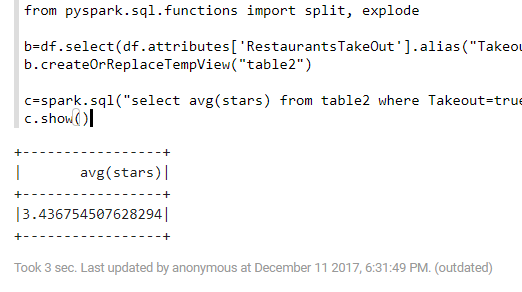
b=df.select(df.attributes['RestaurantsTakeOut'].alias("Takeout"), df.business\_id, explode(df.categories).alias("category"), df.stars)

b.createOrReplaceTempView("table2")

c=spark.sql("select avg(stars) from table2 where Takeout=true and category='Mexican'")

c.show()

**Spark Output:**



**Pig Code:**

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

B = FOREACH A GENERATE FLATTEN(json#'categories') as categories, json#'attributes'#'RestaurantsTakeOut' as attribute, (float)json#'stars' as stars;

C = FILTER B BY attribute matches '.\*true.\*';

D = FILTER C BY categories matches 'Mexican';

E = GROUP D BY (categories);

F = FOREACH E GENERATE group, AVG(D.stars) as avg\_stars;

store F into './answer3.out';

**Pig Output:**

Output file is present in the Zip folder.

From this part, we can infer that the restaurants who offer Mexican food and offer take out have good star ratings.

**Question 4:**

**Spark Code:**

%spark.pyspark

import pyspark

%pyspark

df=spark.read.json("/shared/d/business.json")

from pyspark.sql.functions import split, explode, col

d = df.withColumn("category", explode(col("categories")))

d.createOrReplaceTempView("table4")

aa=spark.sql("select business\_id,latitude, longitude, category, stars, review\_count from table4")

from pyspark.sql.functions import \*

import math

from math import radians, cos, sin, asin, sqrt, atan2, pi

from pyspark.sql.types import \*

aa2=aa.withColumn('latitude\_r', (aa.latitude\*pi)/180)

aa3=aa2.withColumn('longitude\_r', (aa.longitude\*pi)/180)

lat\_tor=((43.6532)\*pi/180)

lon\_tor=((-79.3832)\*pi/180)

aa3.createOrReplaceTempView("table111")

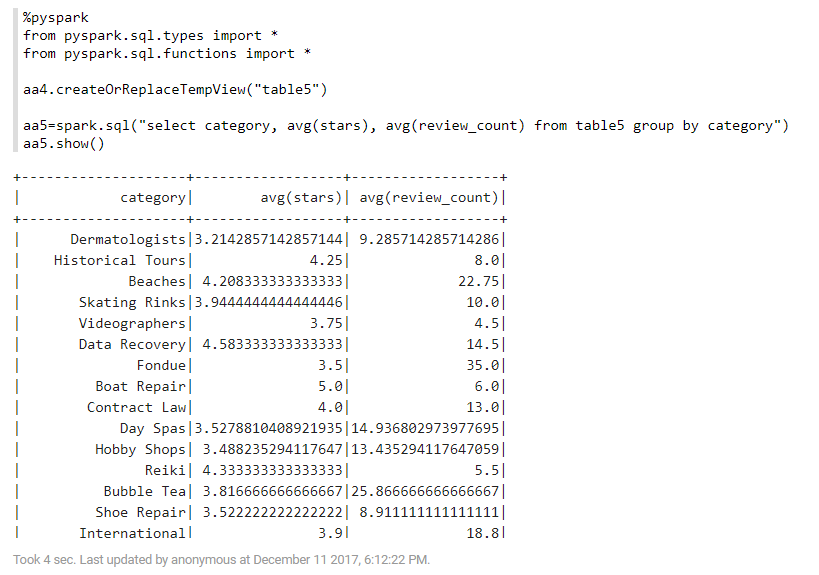
aa4=spark.sql("SELECT \* FROM table111 WHERE acos(sin(0.7618921) \* sin(latitude\_r) + cos(0.7618921) \* cos(latitude\_r) \* cos(longitude\_r - (-1.3855))) \* 6371 <= 15")

aa4.createOrReplaceTempView("table5")

aa5=spark.sql("select category, avg(stars), avg(review\_count) from table5 group by category")

aa5.show()

**Spark Output:**



**Pig Code:**

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

B = FOREACH A GENERATE FLATTEN(json#'categories') as categories, (float)json#'stars' as stars, (int)json#'review\_count' as review\_count, (float)json#'latitude' as latitude, (float)json#'longitude' as longitude;

C = FILTER B BY latitude<43.7889 AND latitude>43.5182 AND longitude< -79.1971 AND longitude> -79.5694;

category1 = GROUP C BY (categories);

D = FOREACH category1 GENERATE group, AVG(C.review\_count), AVG(C.stars);

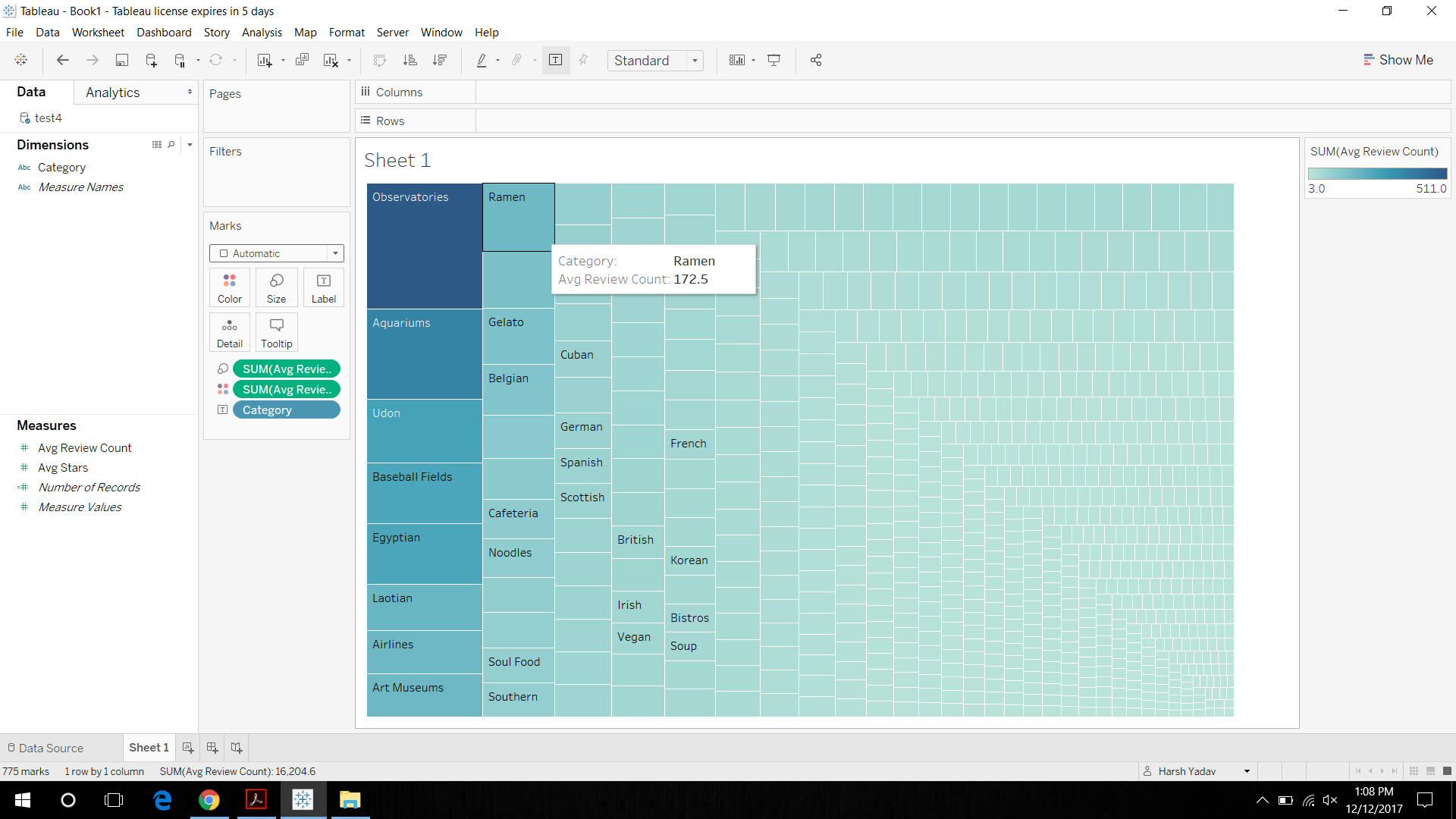
store D into './answer4.out';

**Pig Output:**

Output file is stored in the zip folder.

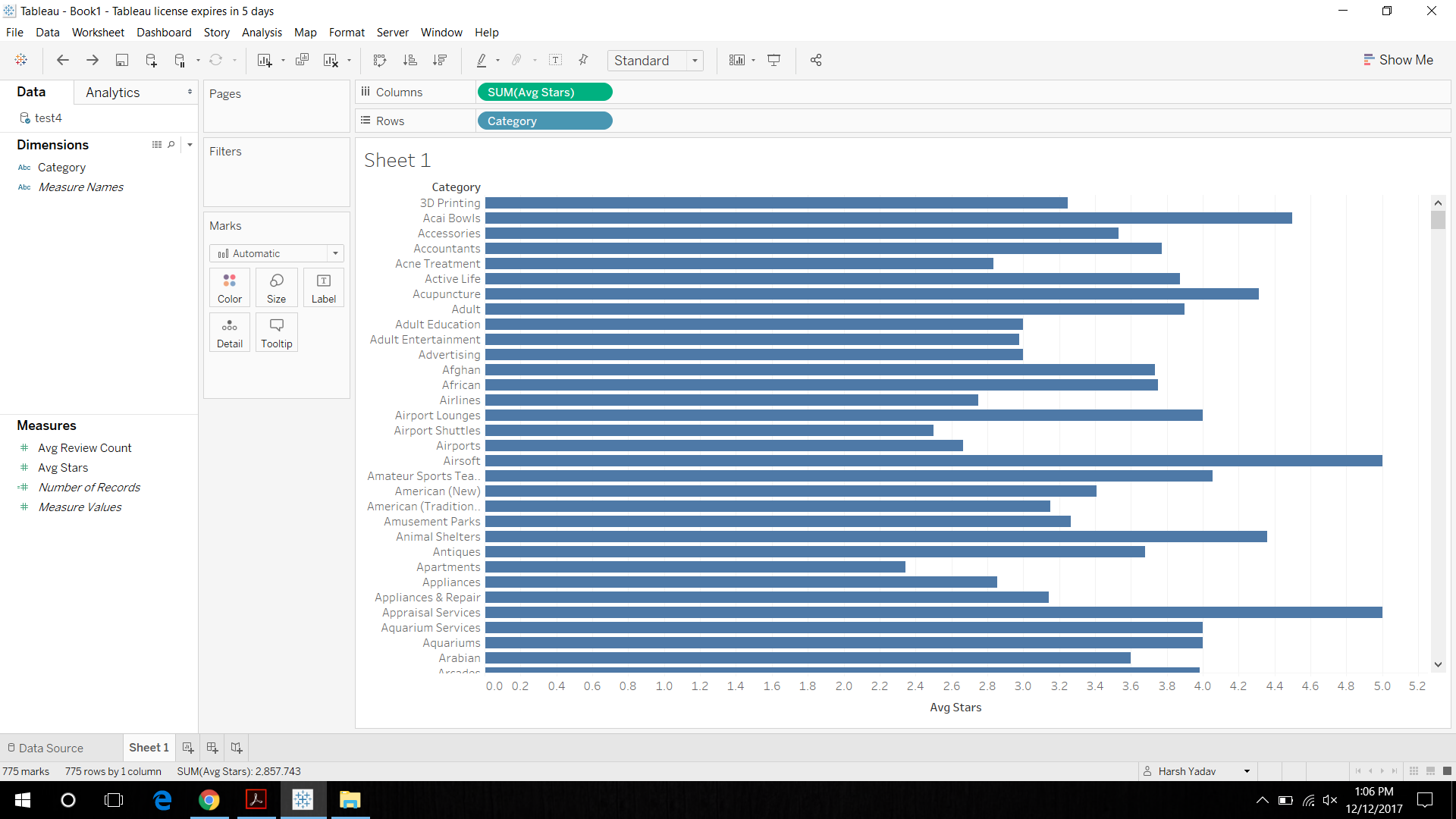
**Visualization:**

Average Review count by Category:



This shows that Ramen Category has the highest average review count.

Average Stars with category:



**Question 5:**

**Spark Code:**

%spark.pyspark

import pyspark

%pyspark

df=spark.read.json("/shared/d/business.json")

from pyspark.sql.functions import split, explode, col

d = df.withColumn("category", explode(col("categories")))

d.createOrReplaceTempView("table4")

aa=spark.sql("select business\_id,latitude, longitude, category, stars, review\_count from table4")

from pyspark.sql.functions import \*

import math

from math import radians, cos, sin, asin, sqrt, atan2, pi

from pyspark.sql.types import \*

aa2=aa.withColumn('latitude\_r', (aa.latitude\*pi)/180)

aa3=aa2.withColumn('longitude\_r', (aa.longitude\*pi)/180)

lat\_tor=((43.6532)\*pi/180)

lon\_tor=((-79.3832)\*pi/180)

aa3.createOrReplaceTempView("table111")

aa4=spark.sql("SELECT \* FROM table111 WHERE acos(sin(0.7618921) \* sin(latitude\_r) + cos(0.7618921) \* cos(latitude\_r) \* cos(longitude\_r - (-1.3855))) \* 6371 <= 15")

aa4.createOrReplaceTempView("table5")

aa5=spark.sql("select \* from table5 where category=='Food'")

aa5.createOrReplaceTempView("table7")

df4=spark.sql("select \* from table7 order by stars asc limit 10")

df5=spark.sql("select \* from table7 order by stars desc limit 10")

df1=spark.read.json("/shared/d/review.json")

df1 = df1.alias('df1')

df2 = df4.alias('df2')

df3 = df5.alias('df3')

dff1=df1.join(df2, "business\_id")

dff2=df1.join(df3, "business\_id")

dfff1=dff1.withColumn('month', concat(dff1.date.substr(6,2)))

dfff2=dff2.withColumn('month', concat(dff2.date.substr(6,2)))

dfff1=dfff1.withColumn("month", dfff1["month"].cast(IntegerType()))

dfff2=dfff2.withColumn("month", dfff2["month"].cast(IntegerType()))

dfff1.createOrReplaceTempView("table8")

#Top 10 businesses

dfff3=spark.sql("select \* from table8 where month<=6")

dfff3.show()

dfff2.createOrReplaceTempView("table9")

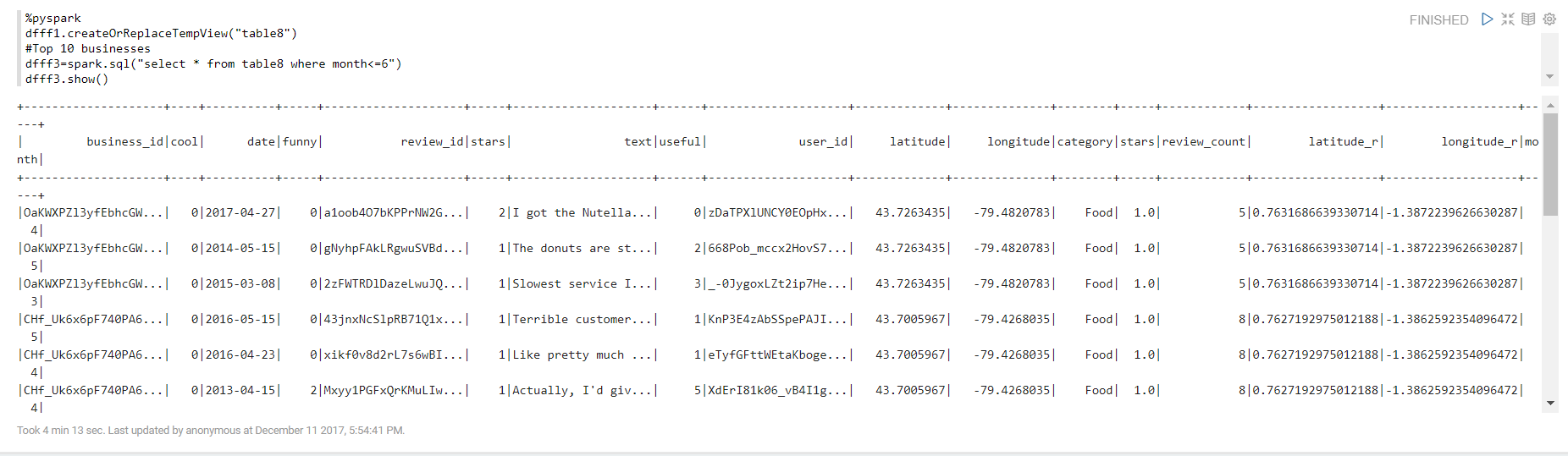
dfff4=spark.sql("select \* from table9 where month<=6")

#For bottom 10 businesses

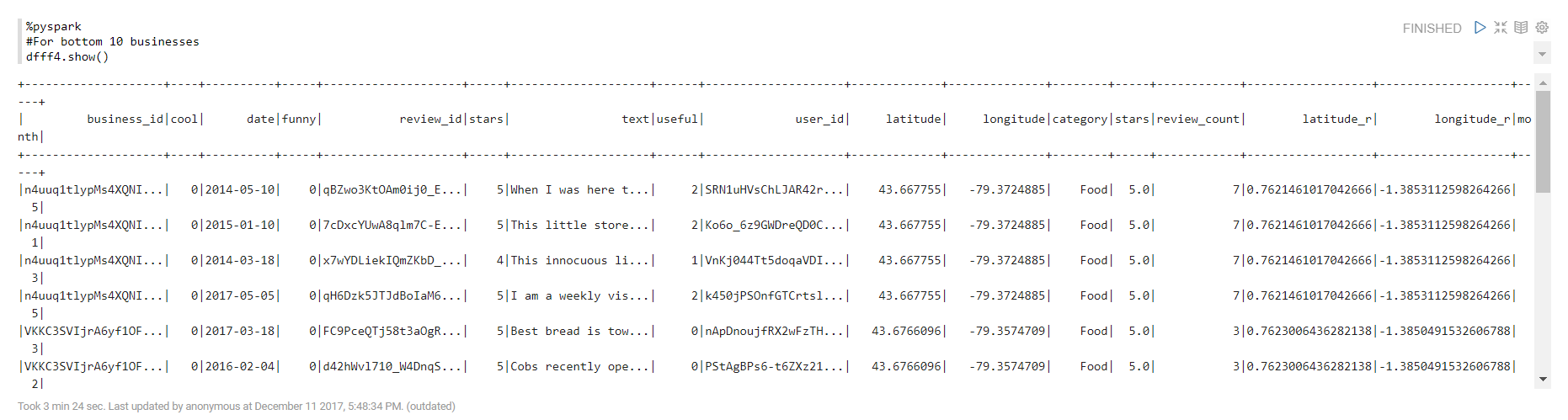
dfff4.show()

**Spark Output:**

Top 10:



Bottom 10:



**Pig Code:**

A = LOAD 'business.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (json1:map []);

B = FOREACH A GENERATE json1#'business\_id' as businessid, FLATTEN(json1#'categories') as categories, (float)json1#'stars' as stars, (int)json1#'review\_count' as review\_count, (float)json1#'latitude' as latitude, (float)json1#'longitude' as longitude;

C = FILTER B BY latitude<43.7889 AND latitude>43.5182 AND longitude< -79.1971 AND longitude> -79.5694;

D = FILTER C BY categories matches 'Food';

E = ORDER D BY stars DESC;

TOP = LIMIT E 10;

F = ORDER D BY stars ASC;

BOTTOM = LIMIT F 10;

G = LOAD 'review.json' USING com.twitter.elephantbird.pig.load.JsonLoader('-nestedLoad') AS (json2:map []);

H = FOREACH G GENERATE json2#'business\_id' as businessid, (float)json2#'stars' as stars, json2#'date' as date;

TOP\_REVIEW = JOIN H BY businessid, TOP BY businessid;

data1 = FOREACH TOP\_REVIEW GENERATE (INT)SUBSTRING (H::date, 5,7) as dateint, H::businessid as businessid, H::stars as stars;

TOP\_MONTH = FILTER data1 BY dateint >= 1 AND dateint < 6;

store TOP\_MONTH into './answer5\_top.out';

BOTTOM\_REVIEW = JOIN H BY businessid, BOTTOM BY businessid;

data2 = FOREACH BOTTOM\_REVIEW GENERATE (INT)SUBSTRING (H::date, 5,7) as dateint, H::businessid as businessid, H::stars as stars;

BOTTOM\_MONTH = FILTER data2 BY dateint < 6;

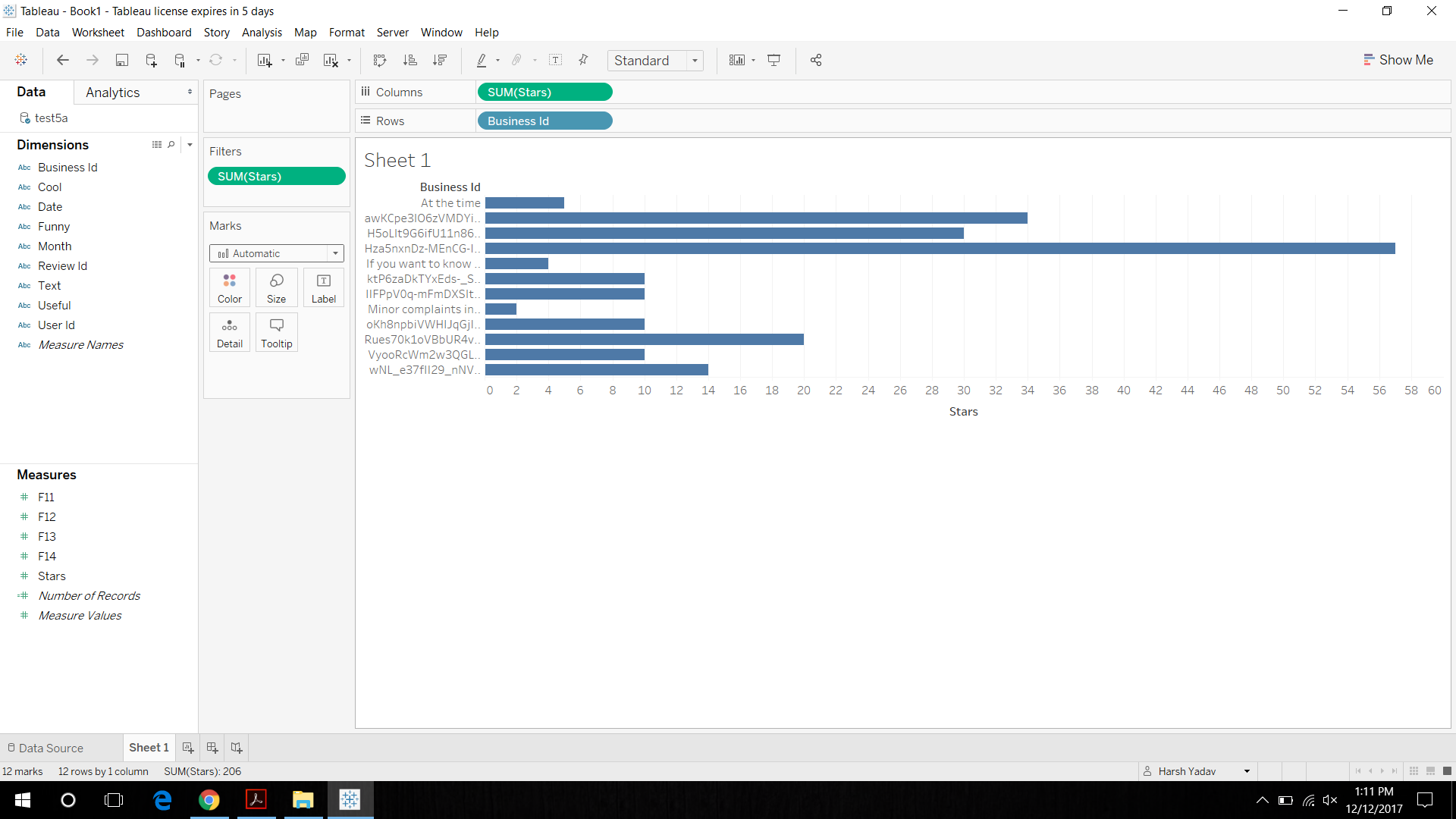
store BOTTOM\_MONTH into './answer5\_bottom.out';

**Pig Output:**

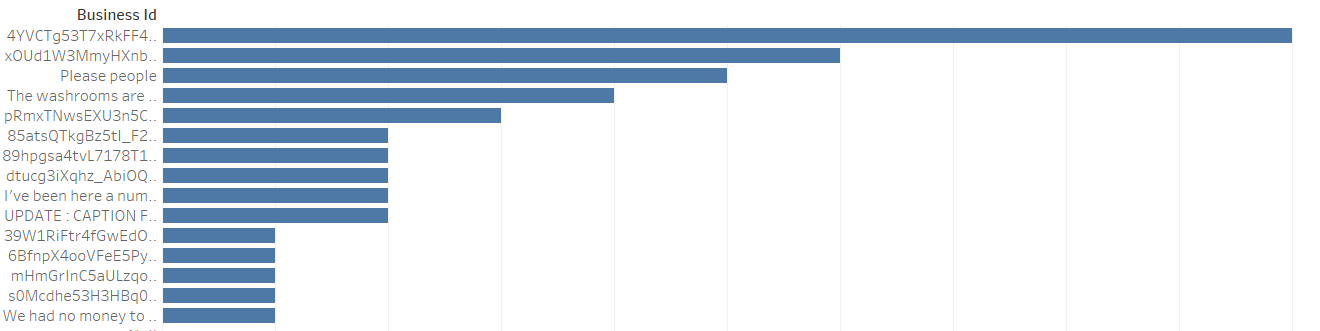
Output file has been stored in the zip folder.

**Visualization:**

Top 10 Business Categories with Stars.



Bottom 10 business categories with stars:



This shows that the business id 4YVCTg53T7xRkFF4 has the highest star ratings.