

Yelp Restaurants Big Data Analysis Project Tutorial

Introduction

- The goal of this project is to take advantage of the Big Data analysis to capture some valuable insights about the restaurant industry from the 2017 Yelp challenge data set.
- Yelp is a social networking site that lets users post reviews and rate businesses.
- Since its inception in 2004, Yelp has collected a staggering 142 million reviews from users for local businesses. They have an average of 145 million unique visitors to their site every month.
- Restaurants is the 2nd largest business category on Yelp and can provide many interesting insights for analysis.

Objectives

The objective of this tutorial is to provide step-by-step instructions for Big Data Analysis of Restaurants in the 2017 Yelp challenge data set. In this tutorial you will learn how to:

- Download and extract the data set from Yelp
- Upload JSON files to HDFS using Ambari
- Install Rcongui SerDe
- Create tables and queries using HiveQL
- Export results and visualize them in Tableau

Step 1: Data Preparation

- 1) Download the data and extract it from http://bit.ly/SKT_COMB_LAB

Data set size:

1 TAR file - 2.28 GB compressed

6 JSON files - 5.79 GB uncompressed

The included files are: business, reviews, user, check-in, tip and photos.

- 2) Use shell terminal or gitbash to untar yelp_dataset.tar.gz file
 - gitbash: <http://www.techoism.com/how-to-install-git-bash-on-windows/>
 - tar -zxvf yelp_dataset.tar
 - will generate 6 jsons and some pdf fiels: ./dataset/business.json, checkin.json, photos.json, review.json, tip.json, user.json
- 3) Since Yelp data set is of high quality it is not necessary to perform any data cleaning.

Step 2: Uploading Data to HDFS For Storage and Analysis

1. Login as user training. You will work from this account for the remaining lab.
2. Create Yelp folder in your HDFS home directory. You can use Hue or HDFS.

```
hdfs dfs -mkdir /user/training/yelp
```

3. Create separate folders inside the Yelp directory for each JSON file:

- business, review, users, tip, checkin

```
hdfs dfs -mkdir /user/training/yelp/business
```

```
hdfs dfs -mkdir /user/training/yelp/review
```

```
hdfs dfs -mkdir /user/training/yelp/users
```

```
hdfs dfs -mkdir /user/training/yelp/tip
```

```
hdfs dfs -mkdir /user/training/yelp/checkin
```

4. Use Hue or the HDFS command line to “put” the json files in their respective directories

Step 3: Adding RCONGUI JSON SerDe

We will need support to read] JSON arrays, maps and nested structures, since our data set contains a lot of nested attributes. Rcongui is one of the available tools to accomplish this. Please refer to this for more information and other options.

<https://community.cloudera.com/t5/Batch-SQL-Apache-Hive/Create-Hive-Table-from-JSON-Files/td-p/64006>

<https://community.cloudera.com/t5/Cloudera-Manager-Installation/Hive-Aux-Jars-path-With-Cloudera-Manager-Clusters/td-p/18232>

<https://stackoverflow.com/questions/19135869/best-place-for-json-serde-jar-in-cdh-hadoop-for-use-with-hive-hue-mapreduce>

1. Download two JAR files from <http://www.congiu.net/hive-json-serde/> into your HDFS Cluster

```
wget -O json-serde-1.3.8-jar-with-dependencies.jar \
www.congiu.net/hive-json-serde/1.3.8/cdh5/json-serde-1.3.8-jar-with-dependencies.jar
```

```
wget -O json-udf-1.3.8-jar-with-dependencies.jar \
www.congiu.net/hive-json-serde/1.3.8/cdh5/json-udf-1.3.8-jar-with-dependencies.jar
```

If the files have successfully been uploaded you will see the following confirmation:

```
-bash-4.1$ wget -O json-serde-1.3.8-jar-with-dependencies.jar www.congiu.net/hive-j
e-json-serde/1.3.8/hdp23/json-serde-1.3.8-jar-with-dependencies.jar;
--2017-10-26 01:08:08--  http://www.congiu.net/hive-json-serde/1.3.8/hdp23/json-s
erde-1.3.8-jar-with-dependencies.jar
Resolving www.congiu.net... 64.90.44.101
Connecting to www.congiu.net|64.90.44.101|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85492 (83K) [application/java-archive]
Saving to: "json-serde-1.3.8-jar-with-dependencies.jar"

100%[=====>] 85,492          330K/s   in 0.3s

2017-10-26 01:08:08 (330 KB/s) - "json-serde-1.3.8-jar-with-dependencies.jar" sav
ed [85492/85492]
```

```
-bash-4.1$ wget -O json-udf-1.3.8-jar-with-dependencies.jar www.congiu.net/hive-j
son-serde/1.3.8/hdp23/json-udf-1.3.8-jar-with-dependencies.jar
--2017-10-26 01:10:03--  http://www.congiu.net/hive-json-serde/1.3.8/hdp23/json-u
df-1.3.8-jar-with-dependencies.jar
Resolving www.congiu.net... 64.90.44.101
Connecting to www.congiu.net|64.90.44.101|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2275 (2.2K) [application/java-archive]
Saving to: "json-udf-1.3.8-jar-with-dependencies.jar"

100%[=====>] 2,275          --.-K/s   in 0s

2017-10-26 01:10:03 (495 MB/s) - "json-udf-1.3.8-jar-with-dependencies.jar" saved
[2275/2275]
```

2. Add the following at the beginning of each HIVE session:

ADD JAR json-serde-1.3.8-jar-with-dependencies.jar;
ADD JAR json-udf-1.3.8-jar-with-dependencies.jar;

```
hive> ADD JAR json-serde-1.3.8-jar-with-dependencies.jar;
Added [json-serde-1.3.8-jar-with-dependencies.jar] to class path
Added resources: [json-serde-1.3.8-jar-with-dependencies.jar]
hive> █
```

You need to do this every HIVE session!!!! Otherwise the following error will be displayed:

```
FAILED: RuntimeException MetaException(message:java.lang.ClassNotFoundException
Class org.openx.data.jsonserde.JsonSerDe not found)
hive> █
```

If you get any SerDe error, it is necessary to reenter these again, if you get out of HIVE, then you need to enter again when you start hive

3. Optionally you can enter the following in HIVE to see the query column names displayed

set hive.cli.print.header=true;

Step 4: Creating Tables in HIVE

The following Hive statements create external tables that allows Hive to query data stored in HDFS. Since we are using JSON Serde it is important to include the following format when creating each table:

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
```

Additionally, if you followed the instructions above and have your JSON data file in the respective folder the following statement will automatically load the data into your tables:

```
LOCATION '/user/training/yelp/folder_name';
```

Optional: You may choose to try Tableau with a free trial of 30 days. Tableau has a nice tutorial that you can try to quickly learn the basics. The maximum JSON file size that can be uploaded directly to Tableau is 128MB and the files in our data set such as for example review.json and users.json files are 3.5GB and 1.5GB in size respectively, we will use HDFS to create smaller tables and queries and that can be exported to Tableau for visualizations.

I. Create Restaurants Table

Follow the following procedure to create Restaurants table from business.json file:

TABLE 1: Create initial BUSINESS table (with all categories and attributes included)

```
CREATE EXTERNAL TABLE business4 (  
  address string,  
  business_id string,  
  categories array<string>,  
  city string,  
  hours struct<friday:string, monday:string, saturday:string, sunday:string, thursday:string,  
  tuesday:string, wednesday:string>,  
  is_open int,  
  latitude double,
```

```
longitude double,  
name string,  
neighborhood string,  
postal_code string,  
review_count int,  
stars double,  
state string,  
Attributes struct<  
  Accepts_Insurance:boolean,  
  Ages_Allowed:string,  
  Alcohol:string,  
  Bike_Parking:boolean,  
  Business_Accepts_Bitcoin:boolean,  
  Business_Accepts_Credit_Cards:boolean,  
  By_Appointment_Only:boolean,  
  Byob:boolean,  
  BYOB_Corkage:string,  
  Caters:boolean,  
  Coat_Check:boolean,  
  Corkage:boolean,  
  Dogs_Allowed:boolean,  
  Drive_Thru:boolean,  
  Good_For_Dancing:boolean,  
  Good_For_Kids:boolean,  
  Happy_Hour:boolean,  
  Has_TV:boolean,  
  Noise_Level:string,  
  Open24Hours:boolean,  
  Outdoor_Seating:boolean,  
  Restaurants_Attire:string,  
  Restaurants_Counter_Service:boolean,
```

Restaurants_Delivery:boolean,
Restaurants_Good_For_Groups:boolean,
Restaurants_Reservations:boolean,
Restaurants_Table_Service:boolean,
Restaurants_Take_Out:boolean,
Smoking:string,
WheelchairAccessible:boolean,
WiFi:string,
Ambience:struct<
Casual:boolean,
Classy:boolean,
Divey:boolean,
Hipster:boolean,
Intimate:boolean,
Romantic:boolean,
Touristy:boolean,
Trendy:boolean,
Upscale:boolean>,
BestNights:struct<
Friday1:boolean,
Monday1:boolean,
Saturday1:boolean,
Sunday1:boolean,
Thursday1:boolean,
Tuesday1:boolean,
Wednesday1:boolean>,
BusinessParking:struct<
Garage:boolean,
Lot:boolean,
Street:boolean,
Valet:boolean,

Validated:boolean>
DietaryRestrictions:struct<
Dairy_Free:boolean,
Gluten_Free:boolean,
Halal:boolean,
Kosher:boolean,
Soy_Free:boolean,
Vegan:boolean,
Vegetarian:boolean>
GoodForMeal:struct<
Breakfast:boolean,
Brunch:boolean,
Dessert:boolean,
Dinner:boolean,
Latenight:boolean,
Lunch:boolean>
HairSpecializesIn:struct<
Africanamerican:boolean,
Asian:boolean,
Coloring:boolean,
Curly:boolean,
Extensions:boolean,
Kids:boolean,
Perms:boolean,
Straightperms:boolean>
Music:struct<
BackgroundMusic:boolean,
Dj:boolean,
Jukebox:boolean,
Karaoke:boolean,
Live:boolean,

```

NoMusic:boolean,
Video:boolean>,
restaurantspricerange2:int>)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/training/yelp/business';

```

To verify that your table was created properly and data was loaded you can test your table as following:

1. Verify total number of businesses in the data set:

```
SELECT count(business_id) FROM business4;
```

```

OK
 _c0
156639
Time taken: 43.331 seconds, Fetched: 1 row(s)
hive>

```

2. Take a look at your data

```
SELECT * FROM business4;
```

```

hive> select * from business4 limit 1;
OK
business4.address      business4.business_id  business4.categories  business4.city
business4.hours       business4.is_open      business4.latitude     business4.longitude
business4.name         business4.neighborhood business4.postal_code   business4.review_count
business4.stars        business4.state        business4.attributes    691 Richmond Rd
YDf95gJZaq05wvo7hTQbbQ ["Shopping","Shopping Centers"] Richmond Heights
{"friday":"10:00-21:00","monday":"10:00-21:00","saturday":"10:00-21:00","sunday":"11:00-18:00","thursday":"10:00-21:00","tuesday":"10:00-21:00","wednesday":"10:00-21:00"} 1 41.5417162 -81.4931165 Richmond Town Square
44143 17 2.0 ON {"accepts_insurance":null,"ages_allowed":null,"alcohol":null,"bike_parking":null,"business_accepts_bitcoin":null,"business_accepts_credit_cards":null,"by_appointment_only":null,"byob":null,"byob_corkage":null,"caters":null,"coat_check":null,"corkage":null,"dogs_allowed":null,"drive_thru":null,"good_for_dancing":null,"good_for_kids":null,"happy_hour":null,"has_tv":null,"noise_level":null,"open24hours":null,"outdoor_seating":null,"restaurants_attire":null,"restaurants_counter_service":null,"restaurant_delivery":null,"restaurants_good_for_groups":null,"restaurants_reservations":null,"restaurants_table_service":null,"restaurants_take_out":null,"smoking":null,"wheelchairaccessible":true,"wifi":null,"ambience":null,"bestnights":null,"businessparking":{"garage":false,"lot":true,"street":false,"valet":false,"validated":false},"dietaryrestrictions":null,"goodformeal":null,"hairspecializesin":null,"music":null,"restaurantspricerange2":2}
Time taken: 0.227 seconds, Fetched: 1 row(s)
hive>

```

```
DESCRIBE business4;
```



```
hive> describe business4;
OK
col_name      data_type      comment
address       string         from deserializer
business_id   string         from deserializer
categories     array<string>   from deserializer
city          string         from deserializer
hours         struct<friday:string,monday:string,saturday:string,sunday:string,thursday:string,tuesday:string,wednesday:string> from deserializer
is_open       int            from deserializer
latitude      double         from deserializer
longitude     double         from deserializer
name          string         from deserializer
neighborhood  string         from deserializer
postal_code   string         from deserializer
review_count  int            from deserializer
stars         double         from deserializer
state         string         from deserializer
attributes    struct<accepts_insurance:boolean,ages_allowed:string,alcohol:string,bike_parking:boolean,business_accepts_bitcoin:boolean,business_accepts_credit_cards:boolean,by_appointment_only:boolean,byob:boolean,byob_corkage:string,caters:boolean,coat_check:boolean,corkage:boolean,dogs_allowed:boolean,drive_thru:boolean,good_for_dancing:boolean,good_for_kids:boolean,happy_hour:boolean,has_tv:boolean,noise_level:string,open24hours:boolean,outdoor_seating:boolean,restaurants_attire:string,restaurants_counter_service:boolean,restaurants_delivery:boolean,restaurants_good_for_groups:boolean,restaurants_reservations:boolean,restaurants_table_service:boolean,restaurants_take_out:boolean,smoking:string,wheelchairaccessible:boolean,wifi:string,ambiance:struct<casual:boolean,classy:boolean,divey:boolean,hipster:boolean,intimate:boolean,romantic:boolean,touristy:boolean,trendy:boolean,upscale:boolean>,bestnights:struct<friday1:boolean,monday1:boolean,saturday1:boolean,sunday1:boolean,thursday1:boolean,tuesday1:boolean,wednesday1:boolean>,businessparking:struct<garage:boolean,lot:boolean,street:boolean,valet:boolean,validated:boolean>,dietaryrestrictions:struct<dairy_free:boolean,gluten_free:boolean,halal:boolean,kosher:boolean,soy_free:boolean,vegan:boolean,vegetarian:boolean>,goodformeal:struct<breakfast:boolean,brunch:boolean,dessert:boolean,dinner:boolean,latenight:boolean,lunch:boolean>,hairspecializesin:struct<africanamerican:boolean,asian:boolean,coloring:boolean,curly:boolean,extensions:boolean,kids:boolean,perms:boolean,straightperms:boolean>,music:struct<backgroundmusic:boolean,dj:boolean,jukebox:boolean,karaoke:boolean,live:boolean,nomusic:boolean,video:boolean>,restaurantspricerange2:int> from deserializer
Time taken: 0.086 seconds, Fetched: 15 row(s)
hive>
```

After create the table in Hive, you may want to execute your queries in Impala. Give it a try. What did you get? (Error message that the JSON serde is not supported) In order to query using Impala, it is best to convert the format to Parquet. The easiest way to do this is to create another table in Hive using CTAS and saving the format as Parquet. For example:

```
CREATE TABLE biz4
STORED AS PARQUET
AS SELECT * FROM business4;
```

TABLE 2: Create EXPLODED table with flattened categories

Since our file contains an array of categories we need to flatten those categories in order to be able to query them easily. We use LATERAL VIEW explode function for this purpose as following:

```
CREATE TABLE exploded  
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  
STORED AS TEXTFILE  
LOCATION '/user/training/yelp/business/exploded'  
AS
```

```
SELECT * FROM business4 LATERAL VIEW explode(categories) c AS cat_exploded;
```

1. To verify that your table was created properly and data was loaded you can test your table as following:

```
SELECT * FROM exploded LIMIT 1;
```

```
hive> select * from exploded limit 1;
OK
exploded.address      exploded.business_id  exploded.categories    exploded
.city exploded.hours  exploded.is_open      exploded.latitude      exploded
.longitude exploded.name  exploded.neighborhood  exploded.postal_code e
exploded.review_count exploded.stars exploded.state exploded.attributes e
exploded.cat_exploded
691 Richmond Rd YDf95gJZaq05wvo7hTQbbQ ["Shopping","Shopping Centers"] Richmond
Heights {"friday":"10:00-21:00","monday":"10:00-21:00","saturday":"10:00
-21:00","sunday":"11:00-18:00","thursday":"10:00-21:00","tuesday":"10:00-21:00",
"wednesday":"10:00-21:00"} 1 41.5417162 -81.4931165 Richmond
Town Square 44143 17 2.0 OH {"accepts_insurance":nul
l,"ages_allowed":null,"alcohol":null,"bike_parking":null,"business_accepts_bitco
in":null,"business_accepts_credit_cards":null,"by_appointment_only":null,"byob":
null,"byob_corkage":null,"caters":null,"coat_check":null,"corkage":null,"dogs_al
lowed":null,"drive_thru":null,"good_for_dancing":null,"good_for_kids":null,"happ
y hour":null,"has_tv":null,"noise_level":null,"open24hours":null,"outdoor_seatin
g":null,"restaurants_attire":null,"restaurants_counter_service":null,"restaurant
s_delivery":null,"restaurants_good_for_groups":null,"restaurants_reservations":n
ull,"restaurants_table_service":null,"restaurants_take_out":null,"smoking":null,
"wheelchairaccessible":true,"wifi":null,"ambience":null,"bestnights":null,"busin
essparking":{"garage":false,"lot":true,"street":false,"valet":false,"validated":
false},"dietaryrestrictions":null,"goodformeal":null,"hairspecializesin":null,"m
usic":null,"restaurantspricerange2":2} Shopping
Time taken: 0.362 seconds, Fetched: 1 row(s)
hive>
```

2. Verify that total number of businesses in the data set is the same as before:

```
SELECT count (DISTINCT business_id) number_businesses FROM exploded;
```

```
OK
number_businesses
156261
Time taken: 38.898 seconds, Fetched: 1 row(s)
hive>
```

3. To find the number of restaurants in the dataset:

```
SELECT count (business_id) number_restaurants FROM exploded
WHERE cat_exploded="Restaurants";
```

```
OK
number_restaurants
51613
Time taken: 35.564 seconds, Fetched: 1 row(s)
hive>
```

TABLE 3: Create RESTAURANTS table

This is our final table which contains only restaurants.

```
CREATE TABLE restaurants
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/mboldin/yelp/business/restaurants'
AS
SELECT * FROM exploded WHERE cat_exploded="Restaurants";
```

To verify that your restaurants table was created properly you can test your table as following:

```
SELECT name, review_count, stars, cat_exploded category FROM restaurants LIMIT 5;
```

```
hive> select name, review_count, stars, cat_exploded category from restaurants limit 5;
OK
name      review_count  stars  category
South Florida Style Chicken & Ribs      4      4.5      Restaurants
Blimpie 10      4.5      Restaurants
Applebee's      21      2.0      Restaurants
China Garden      3      3.0      Restaurants
Rocky's 15      3.0      Restaurants
Time taken: 0.103 seconds, Fetched: 5 row(s)
hive>
```

NOTE ON USING SerDe FOR QUERING NESTED DATA OBJECTS:

Since as mentioned before, our data set contains multiple nested attributes, you can use the following format to query them.

a) To select nested columns it works as following:

parent.child.grandchild

```
SELECT name, attributes.ambience.romantic FROM restaurants LIMIT 5;
```

```
hive> select name, attributes.ambience.romantic from restaurants limit 5;
OK
name      romantic
South Florida Style Chicken & Ribs      false
Blimpie false
Applebee's      false
China Garden      NULL
Rocky's false
Time taken: 0.102 seconds, Fetched: 5 row(s)
```

b) To query BOOLEAN values from nested objects:

```
SELECT name, state, city, attributes.ambience.romantic FROM restaurants
WHERE attributes.ambience.romantic = true LIMIT 10;
```

```
hive> select name, state, city, attributes.ambience.romantic from re
aurants where attributes.ambience.romantic == true limit 10;
OK
name      state  city      romantic
Ristorante Beatrice    QC      Montreal    true
Verona Chophouse      AZ      Chandler    true
Bass Lake Taverne Inn  OH      Chardon true
Edulis   ON      Toronto true
Edge Steakhouse NV     Las Vegas    true
Caffe Boa Ahwatukee    AZ      Phoenix true
White Oaks            OH      Westlake    true
Latinada Tapas Restaurant    ON      Toronto true
Chez Chose            QC      Montréal    true
The Keg Steakhouse + Bar    AZ      Chandler    true
Time taken: 8.106 seconds, Fetched: 10 row(s)
```

II. Create Review Table

Table 4: Create Review Table

```
CREATE EXTERNAL TABLE review (
  business_id string,
  cool int,
  review_date string,
  funny int,
  review_id string,
  stars int,
  text string,
  useful int,
  user_id string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/training/yelp/review';
```



```

> CREATE EXTERNAL TABLE review (
>   business_id string,
>   cool int,
>   review_date string,
>   funny int,
>   review_id string,
>   stars int,
>   text string,
>   useful int,
>   user_id string)
> ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
> LOCATION '/user/mboldin/yelp/review';
OK
Time taken: 0.836 seconds
hive>

```

```
SELECT * FROM review LIMIT 1;
```

```

OK
review.business_id    review.cool    review.review_date    review.funny    r
review.review_id    review.stars    review.text    review.useful    review.user_id
uYHaNptLzDL0V_JZ_MuzUA    0    NULL    0    VfBHSwC5Vz_pbFluy07i9Q    5    M
y girlfriend and I stayed here for 3 nights and loved it. The location of this h
otel and very decent price makes this an amazing deal. When you walk out the fro
nt door Scott Monument and Princes street are right in front of you, Edinburgh C
astle and the Royal Mile is a 2 minute walk via a close right around the corner,
and there are so many hidden gems nearby including Calton Hill and the newly op
ened Arches that made this location incredible.

The hotel itself was also very nice with a reasonably priced bar, very considera
te staff, and small but comfortable rooms with excellent bathrooms and showers.
Only two minor complaints are no telephones in room for room service (not a huge
deal for us) and no AC in the room, but they have huge windows which can be ful
ly opened. The staff were incredible though, letting us borrow umbrellas for the
rain, giving us maps and directions, and also when we had lost our only UK adap
ter for charging our phones gave us a very fancy one for free.

I would highly recommend this hotel to friends, and when I return to Edinburgh (
which I most definitely will) I will be staying here without any hesitation.    0
cjpdDjZyprfyDG3RlkVG3w
Time taken: 0.06 seconds, Fetched: 1 row(s)
hive>

```

To verify number of reviews in the data set:

```
SELECT count(*) FROM review;
```

```

OK
_c0
4736897
Time taken: 139.758 seconds, Fetched: 1 row(s)
hive>

```

Table 5: Create Review Filtered Table

```

CREATE TABLE review_filtered
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE

```

```
LOCATION '/user/training/yelp/review_filtered'
AS
SELECT re.business_id, r.stars, r.user_id
FROM review r JOIN restaurants re
ON r.business_id = re.business_id;
```

To find the number of restaurant reviews:

```
SELECT count(*) FROM review_filtered;
```

```
hive> select count(*) from review_filtered;
OK
_c0
2927731
Time taken: 0.091 seconds, Fetched: 1 row(s)
hive> █
```

III. Create Users Table

Table 6: Create Users Table

```
CREATE EXTERNAL TABLE users (
  average_stars double,
  compliment_cool int,
  compliment_cute int,
  compliment_funny int,
  compliment_hot int,
  compliment_list int,
  compliment_more int,
  compliment_note int,
  compliment_photos int,
  compliment_plain int,
  compliment_profile int,
  compliment_writer int,
  cool int,
  elite array<int>,
  fans int,
```

```
friends array<string>,
funny int,
name string,
review_count int,
useful int,
user_id string,
yelping_since string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION '/user/training/yelp/users';
```

```
hive> CREATE EXTERNAL TABLE users (
>   average_stars double,
>   compliment_cool int,
>   compliment_cute int,
>   compliment_funny int,
>   compliment_hot int,
>   compliment_list int,
>   compliment_more int,
>   compliment_note int,
>   compliment_photos int,
>   compliment_plain int,
>   compliment_profile int,
>   compliment_writer int,
>   cool int,
>   elite array<int>,
>   fans int,
>   friends array<string>,
>   funny int,
>   name string,
>   review_count int,
>   useful int,
>   user_id string,
>   yelping_since string)
> ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
> LOCATION '/user/mboldin/yelp/user';
OK
Time taken: 0.117 seconds
hive> █
```



```
hive> select * from users limit 1;
OK
users.average_stars      users.compliment_cool    users.compliment_cute    users.co
mpliment_funny  users.compliment_hot    users.compliment_list    users.compliment
_more    users.compliment_note    users.compliment_photos    users.compliment_plain u
sers.compliment_profile    users.compliment_writer    users.cool    users.elite    u
sers.fans      users.friends    users.funny    users.name    users.review_cou
nt      users.useful    users.user_id    users.yelping_since
3.8      5174      284      5174      5175      78      299      1435      7829      7397      5
69      1834      16856      [2014,2016,2013,2011,2012,2015,2010,2017]      209      [
"M19NwFwAXKRZzt8koF1lhQ","QRcMZ8pJJBZaKubHOoMDQ","uimsjcHoBnXz1MAKGvB26w","v325
XGF-l9da74ZMWEjyoA","vP5ajc1oGURsNvCXewsnDw","9nSutZOliE9Vg4XVGEx1HA","--2vR0Dis
mQ6WfcSzKWigw","LDJ51sk5SJXovRI2yQZimA","3R_dB9VQ_D3WPJEw7pmorA","8drMKNHWavs2g6
uf0pLtvG","wOGfOjBaP-lCS1NW_En2LQ","AK2-Pvb6E9vgeXWyY4Jxog","DbUSCSMQwD3eiAre0Ue
u8A","B_2gev6exPELs7ZnO4iljg","LQALTuDeCRLwR9NOxUWS5A","kSUU18CH2BRPLK1uUzslWg",
"M-HINGCHOnaQkKq8WDtRMA","9lWyDOySHcc6Jigp2-EPuW","j2Eu9pE22Rp_DRoSp3KgQg","neuz
9oCcHiW4k-jltcC1BA","PRQxRp1IFHPBlbXeDwG3mA","t9vCxlTuXJ941V8ppWVsVQ","pYK8JuByl
omxLIwwyv0Iyw","WTLPH3jIWOUTFMpD4o_7Vg","qAE5pJYa75gRbpC7bgI3Ow","70xFrOWLP04hSC
GY_g3aUw","nRdfX_I0CaOq7lJunJMPpA","W81-CPVrM9c6F8XiNuEUvA","VaVkc537R46xRNpOucR
gvA","j8YxElKHhbg1ghQDSI1v3Q","sYLXiL6q8eiB-D4e3LfWaw","6ueSDFjnsnr-ypVR15WTRg",
"Mi1VEgRHfJ75ESxPuAV9MQ","itz3iIH8qPpm0RowHZ63cA","Wc5L6iuvSNF5WGB1qIO8nw","zb_8
3ib91kA6jBislfTSOq","Pt1yV4SQUR_4LYmkynCGww","Nh_o4LibitBckghR1_8CBQ","vGr0B5weD
DNNmx0zQaQVhw","BujuYvqpupySiAD9ykhXqg","dOfYIGbyfB69VB4pnoKqQA","PSflctuopnjaIW
n-P_o7YQ","zGeG-yb2IDKAGeWSZW2Y1Q","nU5-DpWwid61hHtksA2DMQ","KOS4YIWIxYobnym0sn1
esQ","qNrHLZPurBWJzeAMkFLvvA","5YjvvIgBf-65In3AKXnNjw","Izm94TyF60xP4mPDg9oEBw",
"3ePYVkxiMxBBanReYIZuUw","fGQwLxFbo7RGnNoSDnI4Cw","5rDW0VrYEc9-XyuAU4aMHw","vTug
BdYg34rX4KvGpzBrNA","ebYpHPQWSKoxlzCPJFsFjA","a9x2BusJ-E7aG14LYQw3xw","r3X1OUy7F
UpX22Mi-Eo29Q","IXdmrbRu0veA-OuaP0URwQ","fCiXeYNwrwpM2MGilA1ViQ","pBzBgtCTN9PNU
PfgPDI8A","hZKVx71GlTvg5AaWemQWIQ","4RKq0POQ5jpToRkiiUvJLg","tuIofXv7QuW0GFm-osF
aZA","L-5NjRmaE5KC1XYJk8HVcA","o9XzWtzTuV2X9fyYevXmkw","qtUUQNbKlXy02Dr2TPrtZg",
"29XxHvrJAyvuRaPXu_h-QA","mLXIzLHZ2RAw3MMzpBsFlA","-gQm-IoK2_BMEMx9OgtQnw","qu7C
tlnJjLTtXc_J6YgsTA","s81VRLqYFEpXPIQtGx2mnw","418F_vnEiJmLU-sKpGttHA","RNChXr9iN
cYPP0G6zN4gYA","Cg7HLuddh8s6yKpq4SFpVg","zqXNpAw8zMcuNNPsRXxZ9Q","KB5ooQeFAiVMFb
BavGh_kg","8U8b1EkLQ66djWVcNYyoRQ","n5wlyjHGoe8JTIJmDoGL9Q","u8WIVYVQxiAFJWLdjJN
kIQ","QuPACjm9M7dMqXbXrE9UWQ","YwaKGmRNnSa3R3N4Hf9jLw","P5wzuJlAD8qz-SqpwcUaCQ",
"brqm9p7FMfolAqJgElqGHw","a-Ug_MFryz3utca-NaMkNQ","-UekDWg_Wy4FvxU8l38DIQ","PgKC
Yy3NYMSEIR2Iz1NQrW","OE7LvVziQgLmNachJNg-KA","yKNf3fxNiXnZr67FDTLQgw","7OvtYnfsc
IWBahX6hL2xkQ","bQ3BLXeuDtSdSyGNnLo9mA","pPCzUWTqoiAWUF3MyOXDvQ","xgI4uIQiCmMlyQ
oJnHtzeA","WRTKHSPSum0sg6HDYM5prw","ajCBUlkrK7sdNqWlGvPh3Q","n6hHjOuv8NAWubj0U7I
FLA","eWLJMa7m_pHRdg1VANIK_Q","DONwuwg9iySZ7LFjtcHdCA","CLmvJN5a3l9KutC-nF6LzA",
"tufuEc5f9TWR05_yko46QQ","_ijx1PqANQVFLGNWCibdig","0bwmSWsi5WZfcDu61ZMGhg","IB0
DRx-L80lMtzwPF-lQw","6DC0wklyCCirKpglOOCyow","TwPlaNvnziaN2wB3lk_9Mw","NRRz3KbCS
```

To find the number of all users in dataset:

```
SELECT count(distinct user_id) FROM users;
```

```
OK
c0
1183362
Time taken: 68.755 seconds, Fetched: 1 row(s)
hive> █
```

Table 7: Create Elite Users Table

This table allows us to separate elite users only

```
CREATE TABLE elite_users
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/training/yelp/users/elite'
AS
SELECT * FROM users LATERAL VIEW explode(elite) c AS elite_year;
```

IV. Create Tip Table

Table 8: Create Tip Table

```
CREATE EXTERNAL TABLE tip (
text string,
date_tip string,
likes int,
business_id string,
user_id string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/mboldin/yelp/tip';
```

```
hive> CREATE EXTERNAL TABLE tip (
> text string,
> date_tip string,
> likes int,
> business_id string,
> user_id string)
> ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
> STORED AS TEXTFILE
> LOCATION '/user/mboldin/yelp/tip';
OK
Time taken: 10.297 seconds
hive>
```

Step 4: Create tables and queries using HiveQL and visualize in Hue (or Tableau if you want to try)

We can save the results of HIVE queries in HDFS in form of JSON files. Tableau works really well with nested JSON files, making it easy to visualize the results of analysis.

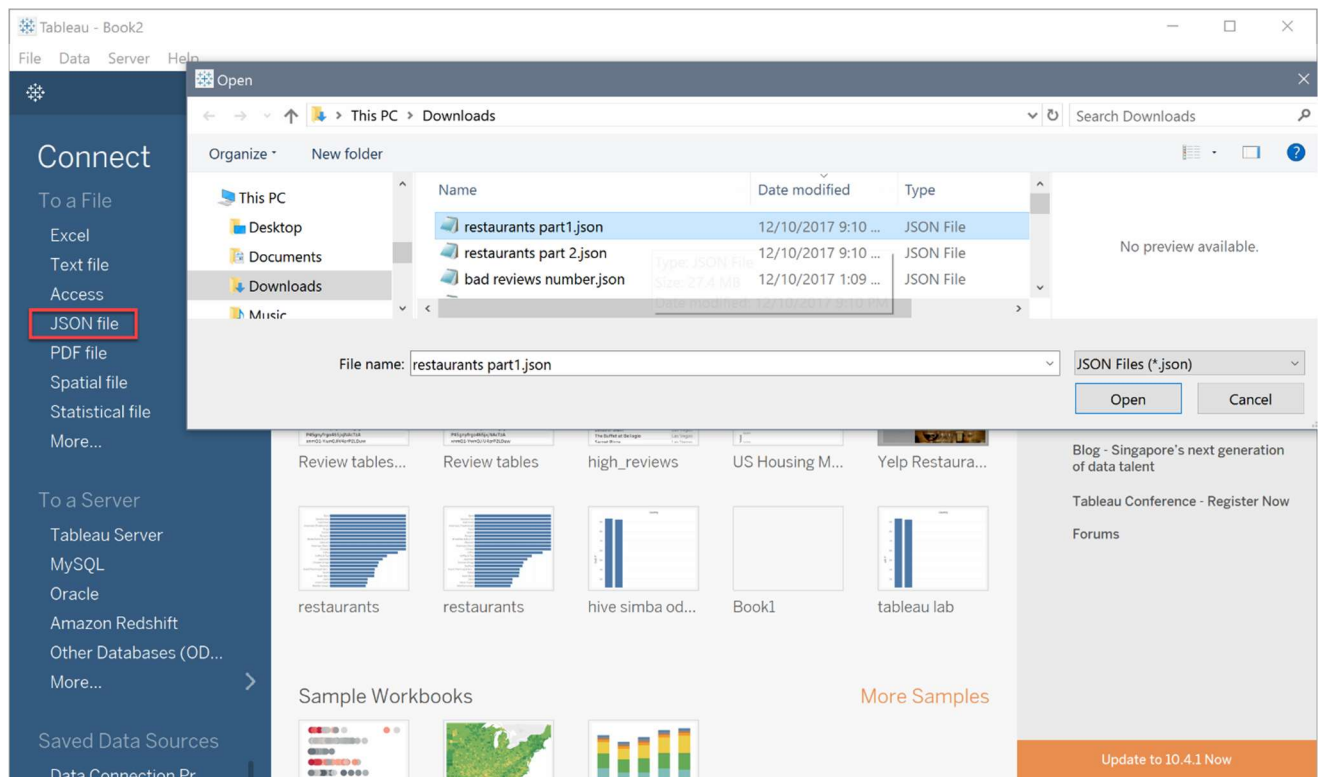
To create visualizations in Tableau we have 2 options:

- If JSON file size is less than 128MB we can upload it directly to Tableau
- Otherwise, we can export just the results of the Hive query in form of JSON files to Tableau for visualization.

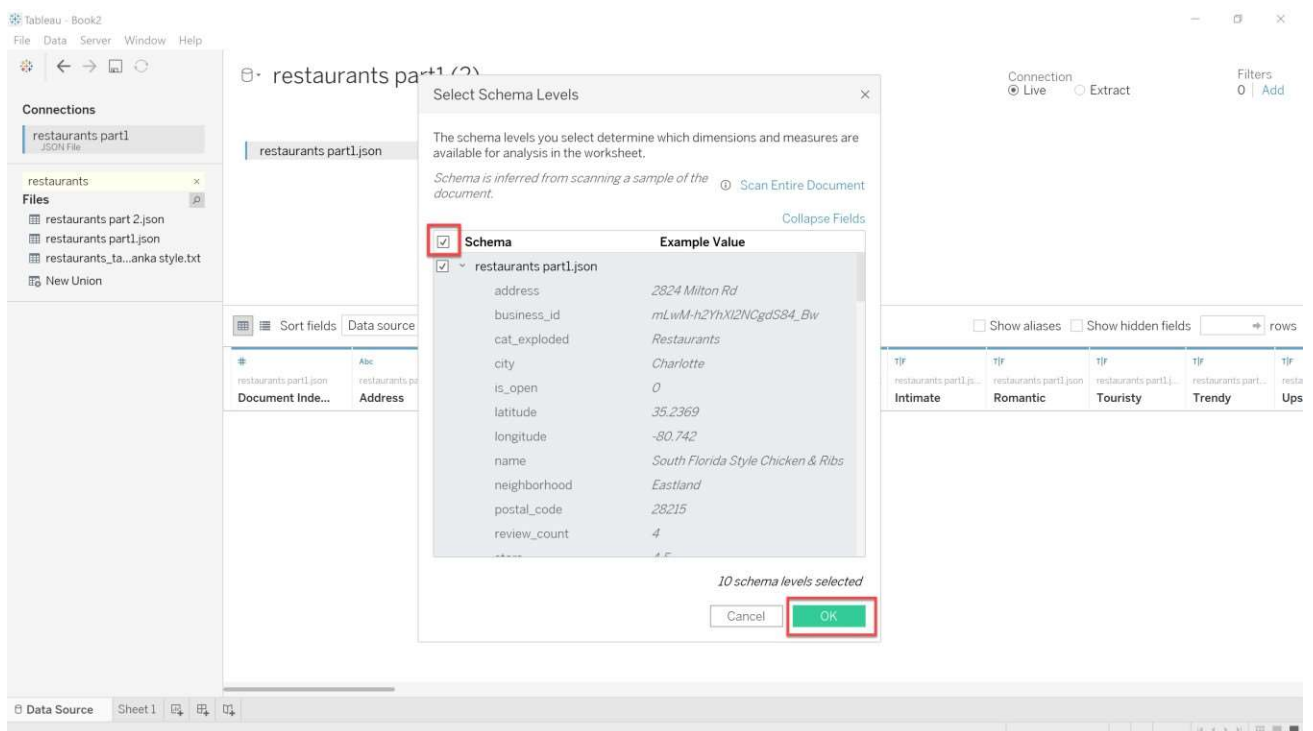
1) Map of restaurants across United States

In order to visualize the results of queries in Tableau you should do the following:

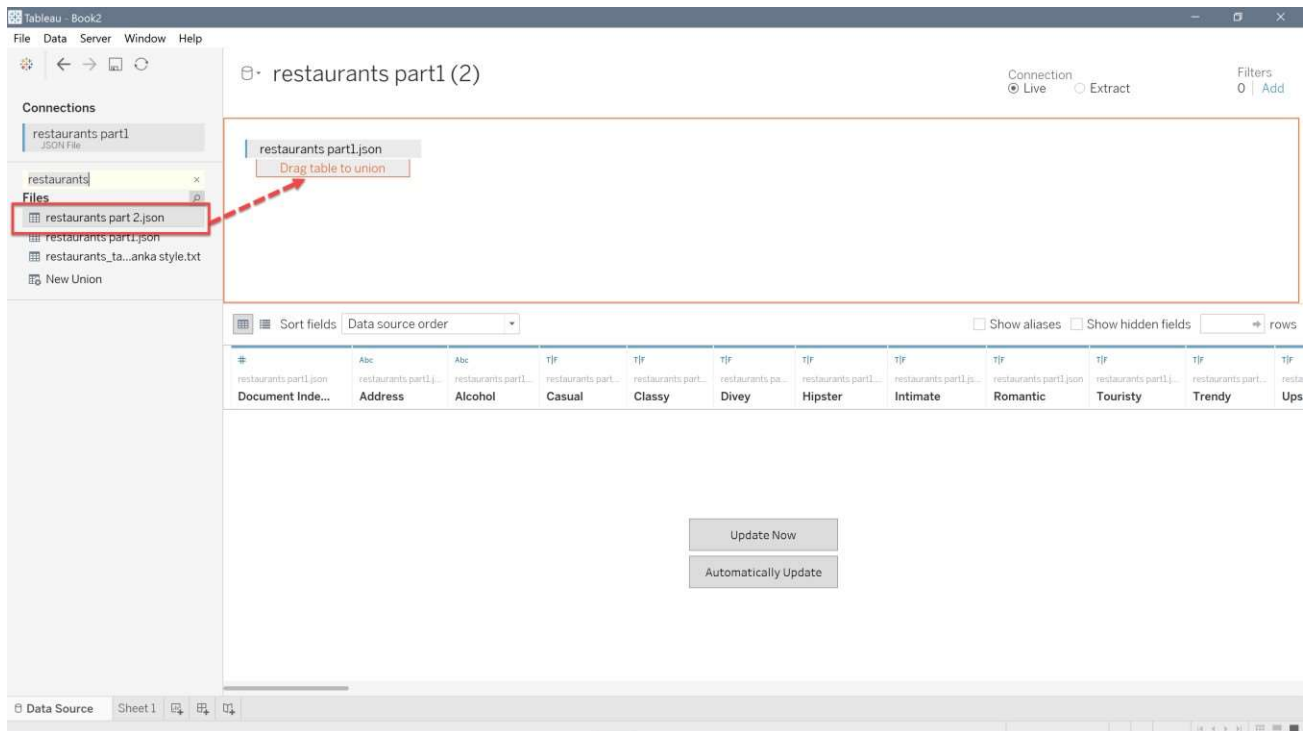
- a Open Hue. Download restaurants table to your computer by downloading two files 000000_0 and 000001_0 which are located in the business/ restaurants folder in HDFS.
- b Click on the file, a pop-up preview window will open, then click on Download file. Repeat for the second file.
- c For ease of use, you can rename the files as restaurants part1.json and restaurants part2.json
- d Open Tableau and connect JSON file as following:



e. Open the first file and select the Schema check mark as shown below.



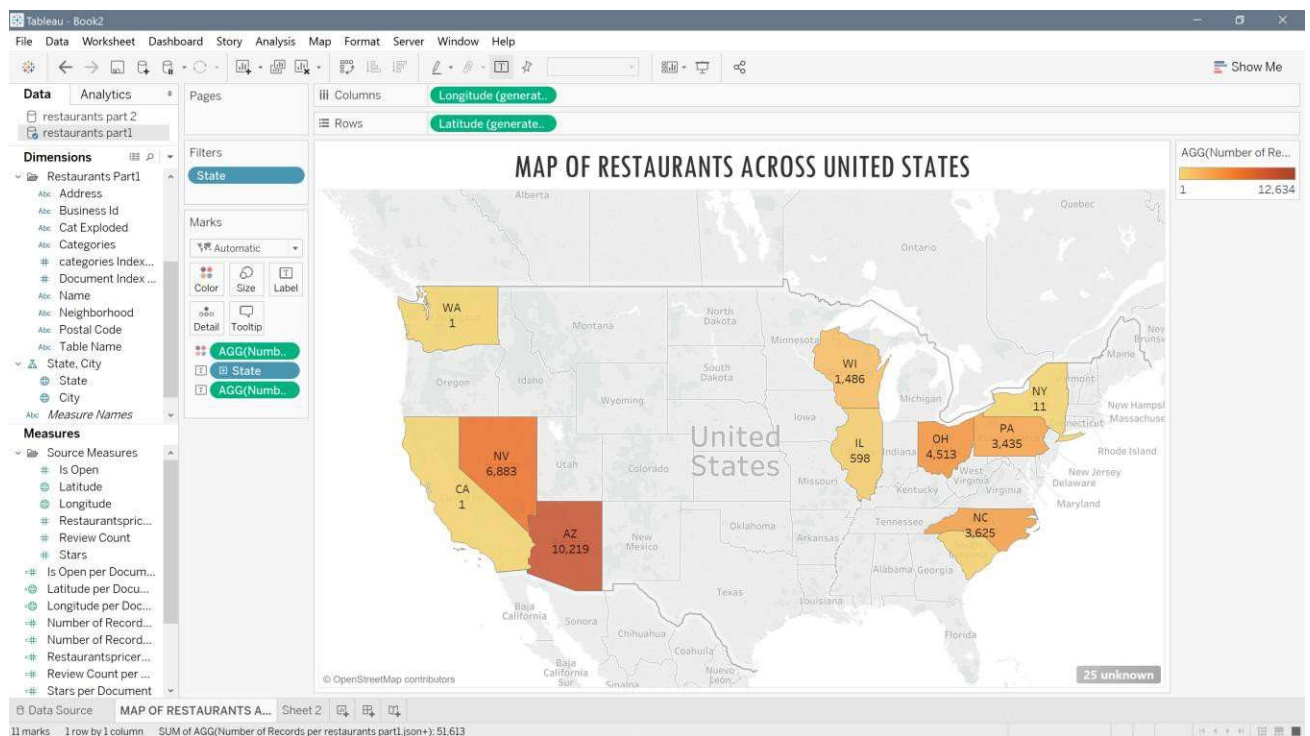
f. Since we have 2 files, we need to join them in Tableau as following:



g To create the map visualization follow these instructions.

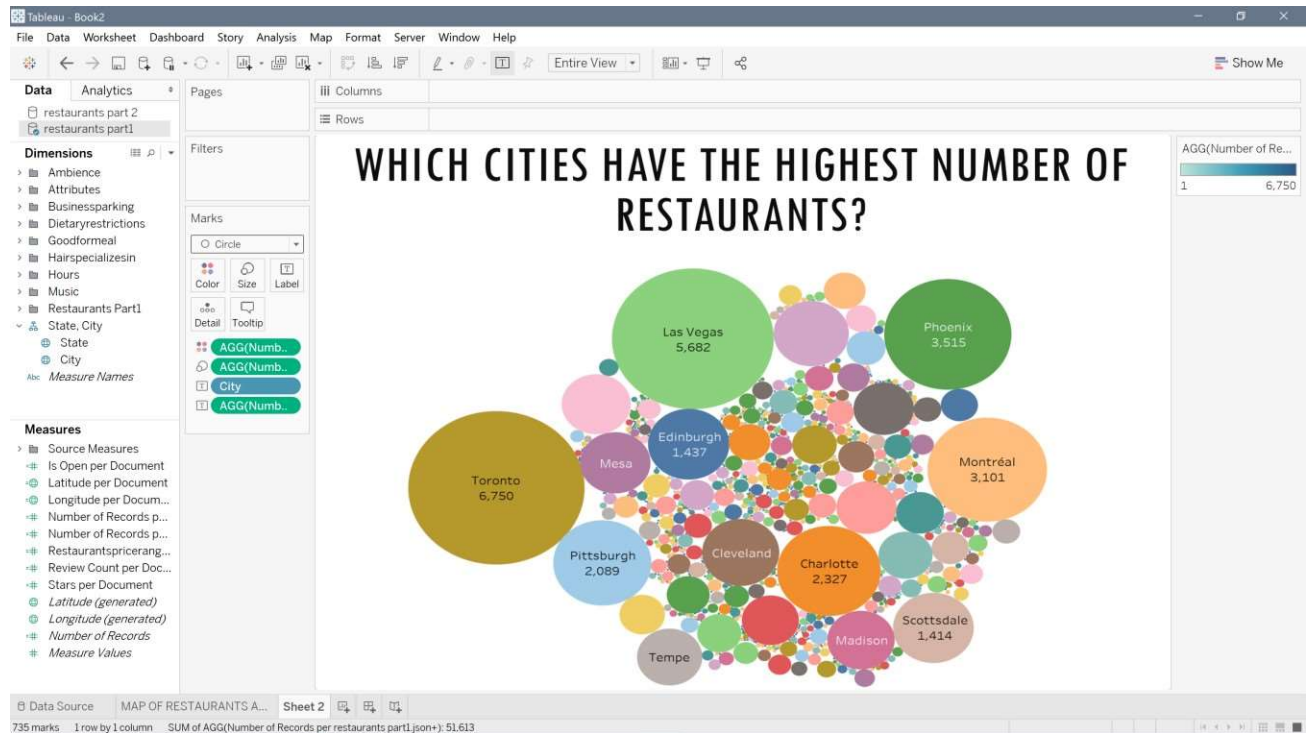
Click on Sheet 1, drag longitude to columns, latitude to rows, number of records to color on Marks, number of records to label on Marks, state to label on Marks.

Click on color on the marks to adjust the colors to your choice. Click on title to change the title of the visualization. Your tableau worksheet should look like the image below:

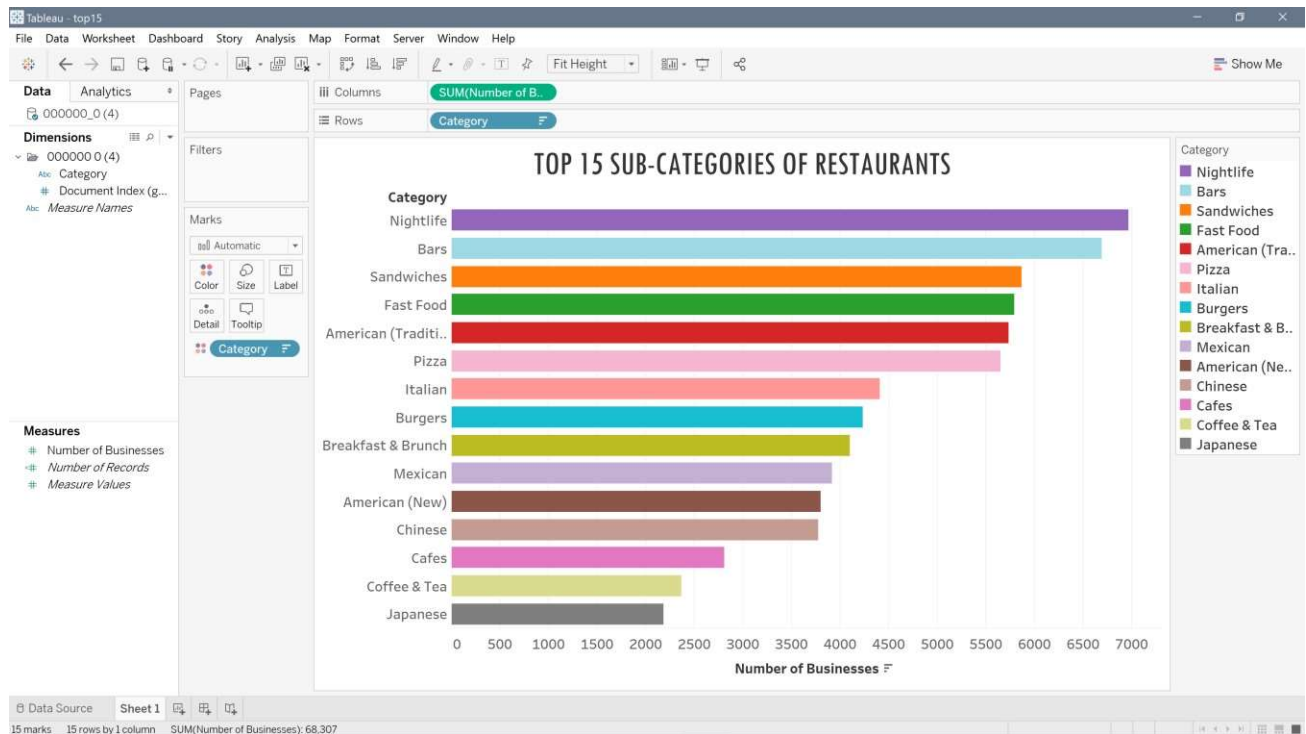


- h You can follow procedures described above to produce the rest of the visualizations in this tutorial.

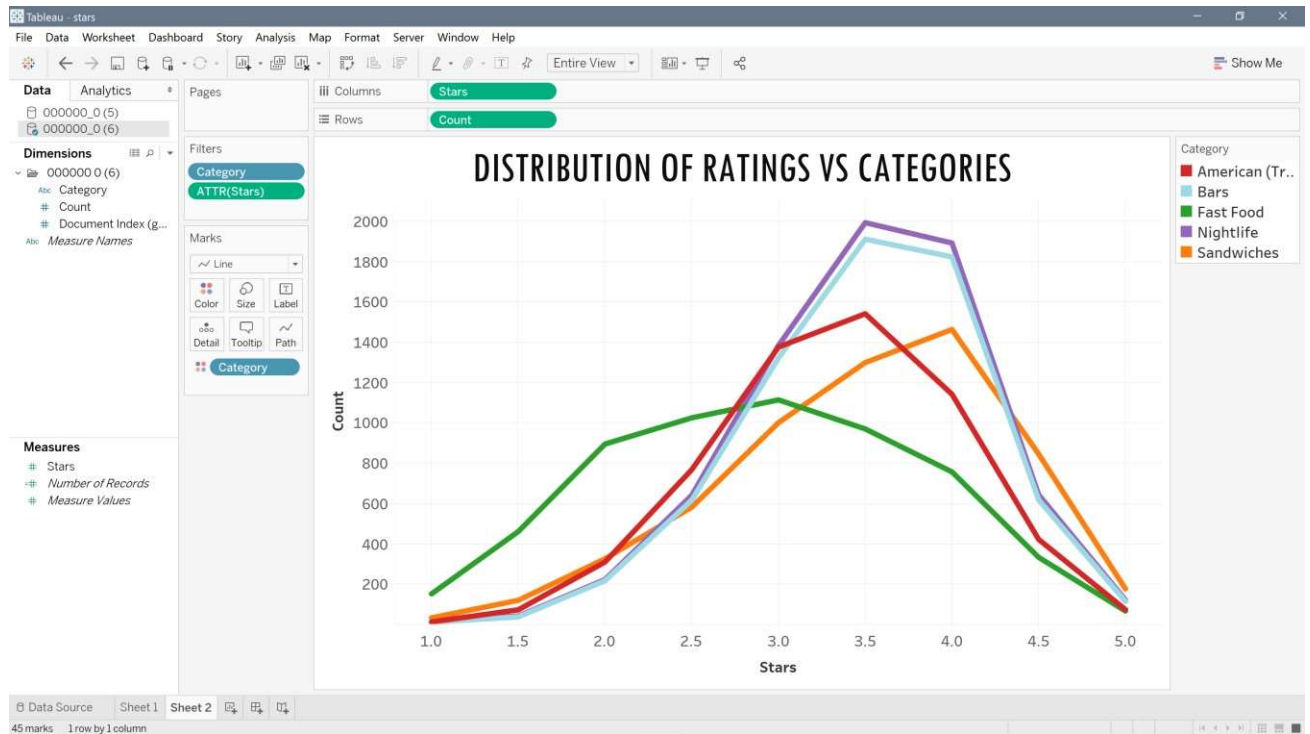
2) Which Cities Have The Highest Number Of Restaurants?



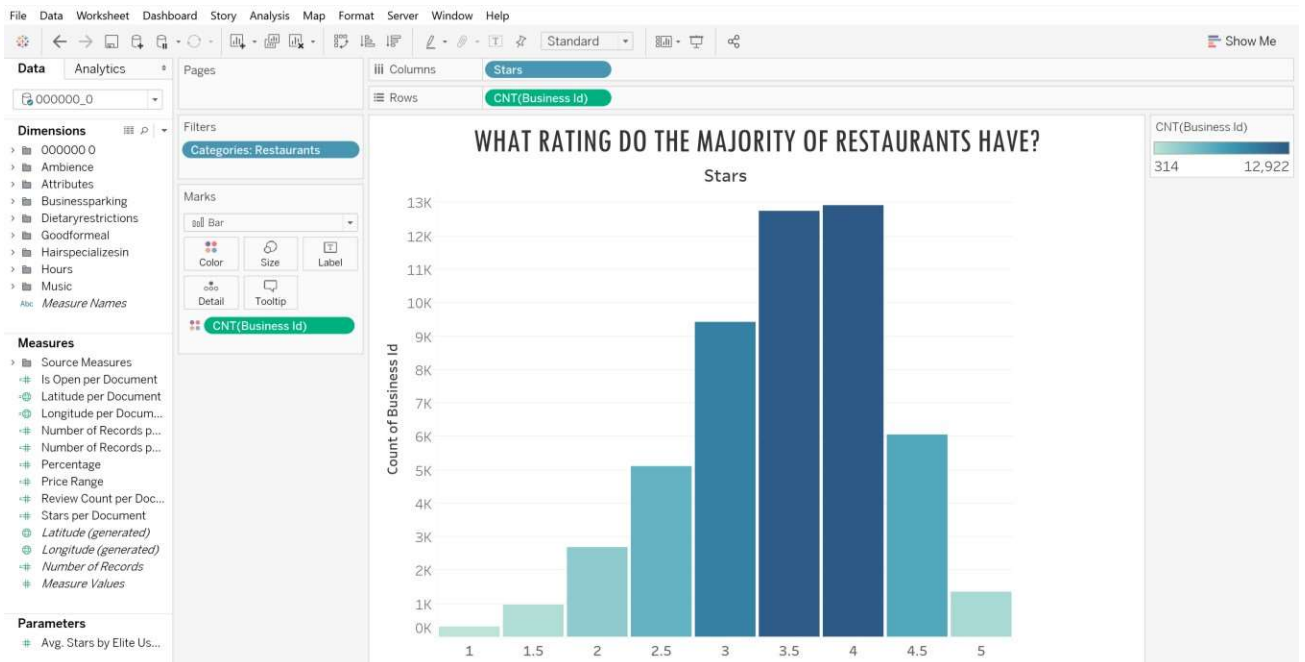
3) Top 15 Sub-Categories Of Restaurants



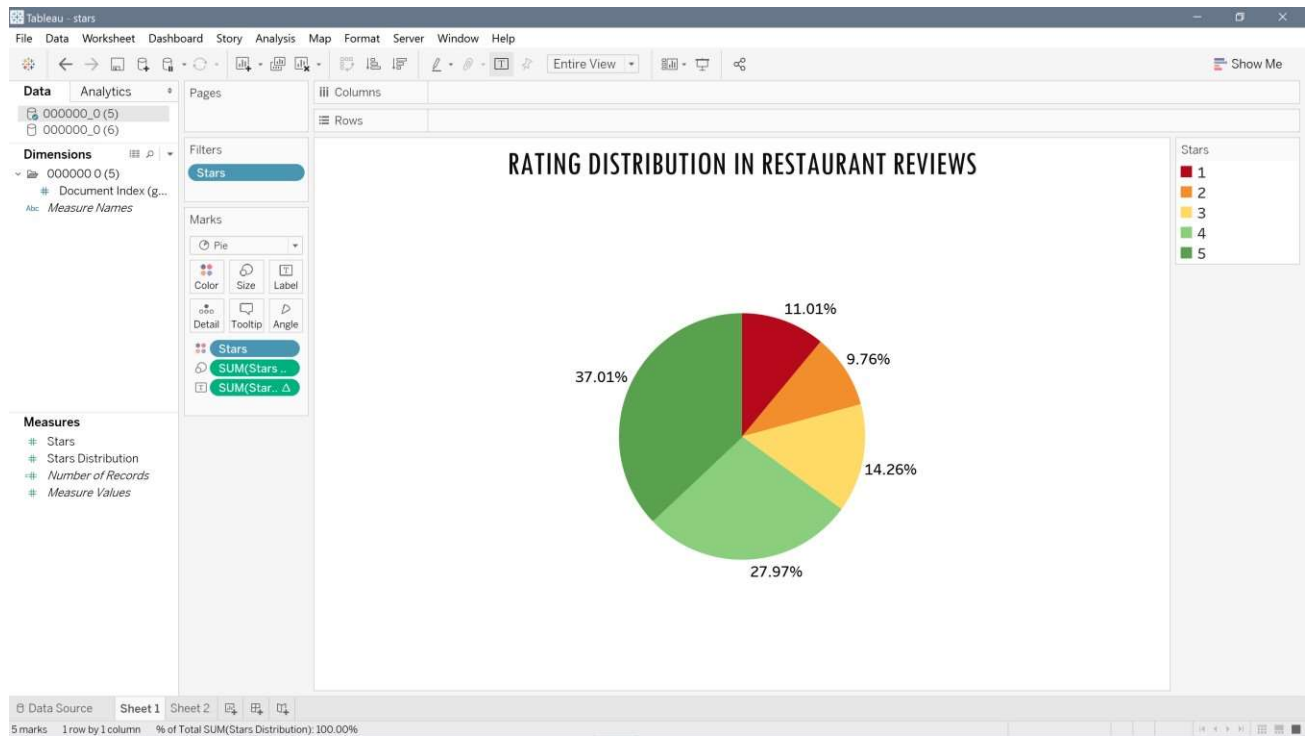
4) Distribution of ratings vs categories



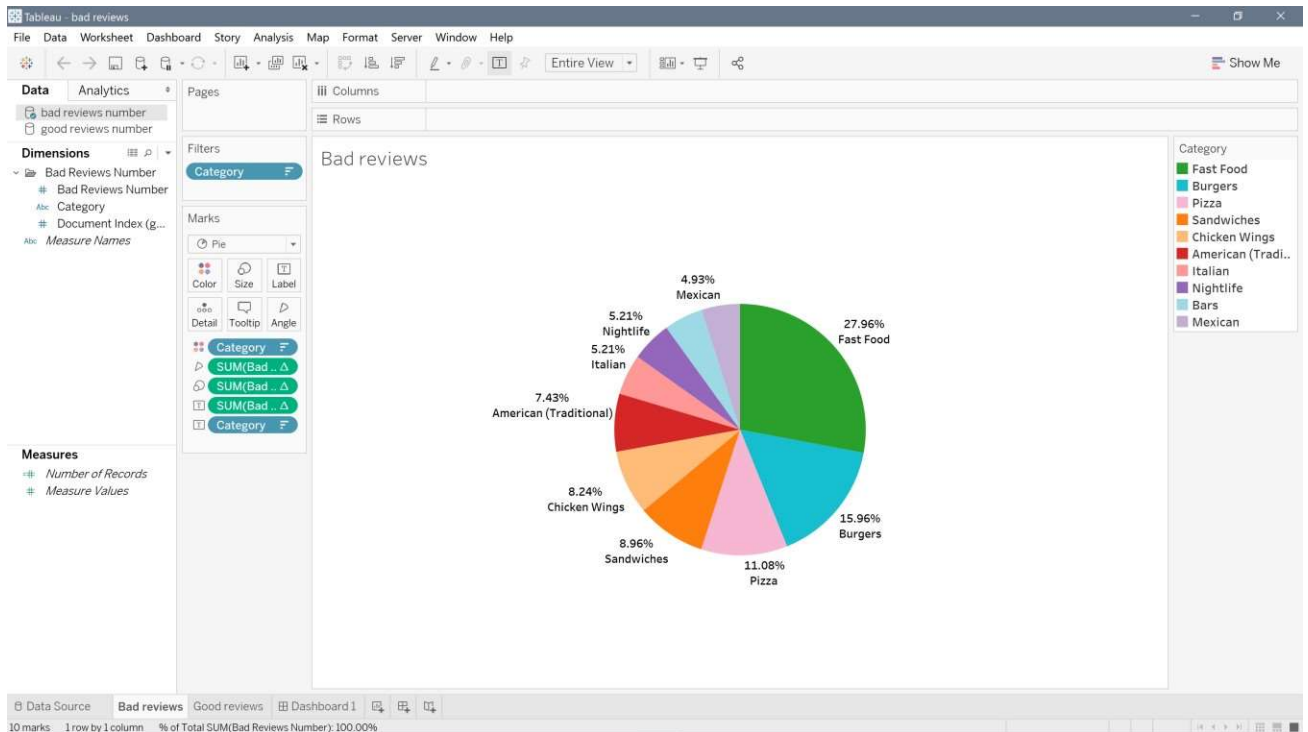
5) What ratings do the majority of restaurants have?

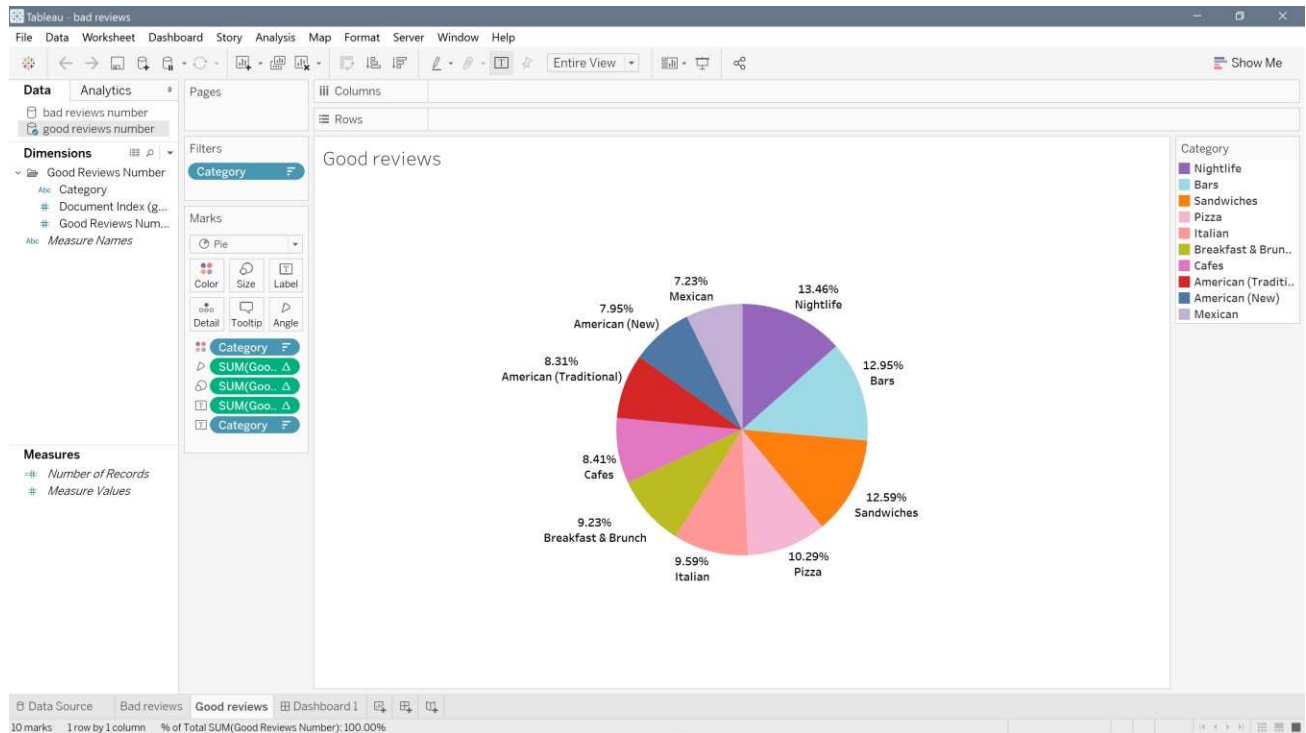


6) Rating distribution in restaurant reviews

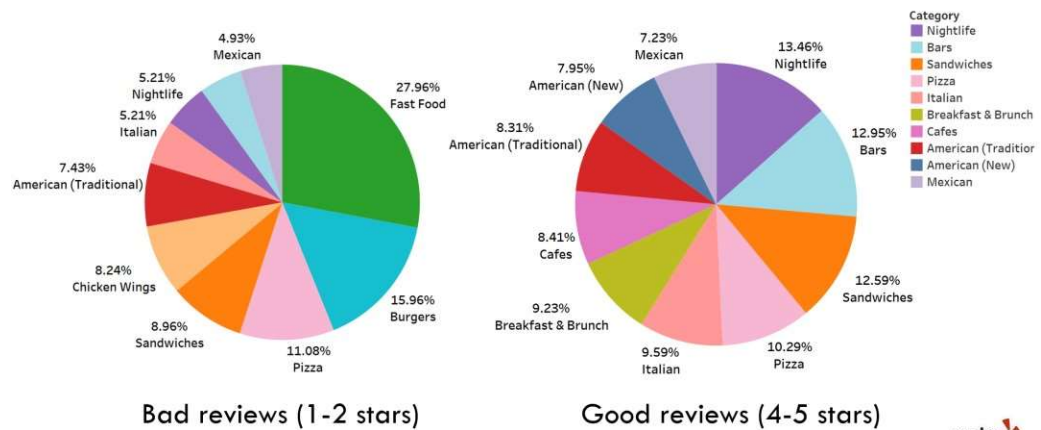


7) Which restaurants get bad vs good reviews?

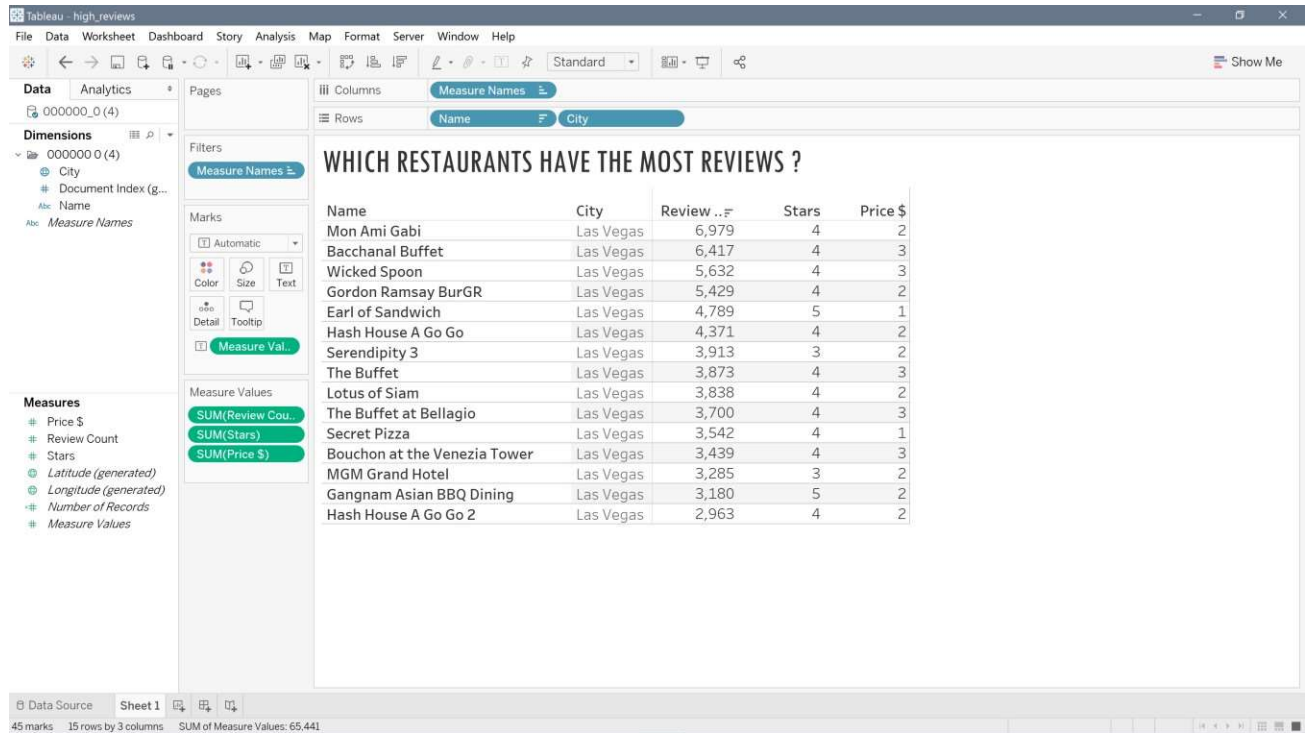




WHICH RESTAURANTS GET BAD VS GOOD REVIEWS?



8) Which restaurants have the most reviews?



9) What number of yelp users are elite? Do they rate differently than non -elite users

