

Project: YELP Dataset Analysis

Bhagyashree Bhagwat

Manali Joshi

Priyanka Purushu

Objective:

Yelp is an internet based company which contains reviews for millions of restaurants and other businesses which are provided by the yelp users. The Yelp dataset challenge that this project was done on contains 2.7M reviews by 687K users for about 86K businesses and focuses on U.K, Germany, Canada and 5 cities in US from 2014 to June 2016.

In our project, we have considered only the 5 cities in the US and did our analysis for the Food and Restaurant business category from 3 data files – business, user and Review. The analysis performed on the Yelp Dataset are:

- In which year Yelp has got maximum number of users?
- Forecast of users joining Yelp in coming years.
- Who are the most active users on Yelp?
- Which users' review is the most popular based on votes?
- Which city has the maximum no of closed business?
- Sentiment Analysis on Top 8 food chains in the USA based on user reviews.
- Best suited restaurants for tourists.

Introduction:

This project aims at performing data analysis and providing insights on Yelp's Dataset using HIVE, PIG and presenting visualization on Microsoft Excel - Power view, 3D maps and Tableau. In this tutorial, through each analysis we did you'll learn how to use BigInsights to:

- Load data from local desktop(windows) to Linux shell and unzip tar files
- Extract Json file using PIG and Hive
- Data cleaning using PIG and Hive
- Download and upload files to HDFS
- Create Hive tables to query the Yelp dataset for analysis
- Create a dictionary table for sentiment analysis
- Create Hive queries to analyze the sentiment of data using dictionary
- Use Microsoft Excel to connect to BigInsights (directly or using an ODBC connection) to retrieve the analyzed data
- Use Excel 3D Map for 3D visualization – Sentiment Analysis
- Use Tableau for visualization of the analyzed data and also to present the forecast of yelp users in the upcoming years

Prerequisites:

Everything you need to go through the scripts and queries is already provisioned with the cluster. To export the analyzed data to Microsoft Excel, you must meet the following requirements:

- You must have **Microsoft Excel 2010, 2013 or 2016** installed.
- You must have your Excel PowerView and 3D Map enabled.
- Tableau 9.2 or 9.3 installed for visualization of the analyzed data
- You must have Microsoft Hive ODBC Driver to import data from Hive into Excel. Select either the 32-bit or 64-bit version based on your version of Microsoft Excel. But, BigInsights does not support it yet as of Sept 2016.

Yelp Dataset Loaded into BigInsights:

You need to download the Yelp dataset from the following link mentioned below to your local desktop in order to retrieve the .tar file.

- https://www.yelp.com/dataset_challenge

Once the Yelp dataset is downloaded into the local desktop, You need to remotely access your BigInsights that you executed in your Bluemix account using *ssh*. Below are the shell commands required to have to push these files onto the ssh from the local desktop(windows- cmd). You can upload the data files to remote node of BigInsights using **pscp** shell commands.

```
C:\Users\priya> pscp
C:\Users\priya\Downloads\yelp_dataset_challenge_academic_dataset.tar
cali@bi-hadoop-prod-4280.bi.services.us-
south.bluemix.net:/home/cali/project
```

You need to untar **yelp_dataset_challenge_academic_dataset.tar**, in order to retrieve the yelp data files.

```
$ tar -zxvf yelp_dataset_challenge_academic_dataset.tar
```

```
-bash-4.1$ pwd
/home/cali/project
-bash-4.1$ ls
businessjson.hql          pig_1480965629728.log    reviewjson.pig           userjson.hql             yelp_dataset_challenge_academic_dataset.tar
business_table.hql       pig_1480965683793.log    sentimentanalysis.hql    yelp_academic_dataset_business.json  Yelp_Dataset_Challenge_Terms_round_8.pdf
Dataset_Challenge_Academic_Dataset_Agreement.pdf  pig_1480967292667.log    tipsjson.pig            yelp_academic_dataset_checkin.json
dictionary.hql           pig_1481004872332.log    tipsjson.pig            yelp_academic_dataset_review.json
pig_1480965434186.log    pig_1481005727936.log    touristanalysis.hql     yelp_academic_dataset_tip.json
pig_1480965549569.log    review.hql               yelp_academic_dataset_user.json
```

Now we have to extract these json files in order to perform the analysis. To query on the json file we have adopted two techniques: PIG and Hive.

Before we upload these files to HDFS we have to create a directory in HDFS. Run the following HDFS commands to create to store these files into a directory in HDFS.

```
$ hdfs dfs -mkdir /user/cali/priyanka/yelp_review_new
$ hdfs dfs -mkdir /user/cali/priyanka/yelp_new_tips
```

Using Pig we have extracted 2 data files of yelp – Reviews and tips data files. Below are the shell commands required for extraction and cleaning these data files in PIG. The code for extraction and cleaning has been written in pig script.

```
$ vi reviewjson.pig
```

Now enter the below mentioned pig code into reviewjson.pig script using vi editor. The script contains code to load the json file with the schema provided in the **JsonLoader()**. The cleaning of data is done using attributes such as **FOREACH GENERATE, FLATTEN AND REPLACE**. The extracted json file is stored into HDFS as csv file using **CSVExcelStorage()**.

```
reviews = load
'/user/cali/priyanka/yelp_review/yelp_academic_dataset_review.json'
using
JsonLoader('votes:(funny:int,useful:int,cool:int),user_id:chararray,
review_id:chararray,stars:int,date:chararray,text:chararray,type:chararray,business_id:chararray');

review_final = FOREACH reviews generate FLATTEN(votes), user_id,
review_id, stars, REPLACE(REPLACE(text, '\n', ''), '\t', ''), date,
type, business_id;

STORE review_final INTO '/user/cali/priyanka/yelp_review_new' USING
org.apache.pig.piggybank.storage.CSVExcelStorage('#',
'YES_MULTILINE', 'WINDOWS', 'WRITE_OUTPUT_HEADER');
```

```

reviews = load '/user/cali/project/yelp_academic_dataset_review.json' using JsonLoader('votes:(funny:int,useful:int,cool:int),user_id:chararray,review_id:chararray,stars:int,date:chararray,text:chararray,type:chararray,business_id:chararray');

review_final = FOREACH reviews generate FLATTEN(votes), user_id, review_id, stars, REPLACE(REPLACE(text, '\n', ''), '\t', ''), date, type, business_id;

STORE review_final INTO '/user/cali/priyanka/yelp_review' USING org.apache.pig.piggybank.storage.CSVExcelStorage('#', 'YES_MULTILINE', 'WINDOWS', 'WRITE_OUTPUT_HEADER');

```

"reviewjson.pig" 6L, 569C

Now in order to execute the above-mentioned pig script use the following command

```
$ pig reviewjson.pig
```

Similarly a pig script has been written to extract and clean the yelp_academic_dataset_tips.json using the vi editor . The cleaning of data is done using attributes such as **FOREACH GENERATE, FLATTEN AND REPLACE**. The extracted json file is stored into HDFS as csv file using **CSVExcelStorage()**.

```
$ vi tipjson.pig
```

```

tips = LOAD
'/user/cali/priyanka/yelp_tip/yelp_academic_dataset_tip.json' using
JsonLoader
('user_id:chararray,text:chararray,business_id:chararray,likes:int,date:chararray,type:chararray');

tips_final = FOREACH tips GENERATE user_id, REPLACE(REPLACE(text,
'\n', ''), '\t', ''),business_id,likes,date,type;

STORE tips_final INTO '/user/cali/priyanka/yelp_new_tips' USING
org.apache.pig.piggybank.storage.CSVExcelStorage('#',
'YES_MULTILINE', 'WINDOWS', 'WRITE_OUTPUT_HEADER');

```



```

CREATE TABLE IF NOT EXISTS business_table_new (business_id
STRING,full_address STRING,open STRING,categories STRING,city
STRING,review_cnt STRING,name STRING,longitude STRING, state STRING, stars
STRING,latitude STRING,live_music STRING,background_music STRING,karaoke
STRING,video STRING,dj STRING,jukebox STRING,park_lot STRING, street
STRING,garage STRING,valet STRING,validated STRING,Happy_Hour
STRING,Price_Range STRING,Wi_Fi STRING,Ages_Allowed STRING, Open_24_Hours
STRING,vegan STRING,dairy_free STRING,halal STRING,gluten_free
STRING,kosher STRING,soy_free STRING,vegetarian STRING);

```

```

FROM yelp_business
INSERT OVERWRITE TABLE business_table_new
SELECT get_json_object(json_response,
'$.business_id'),REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_respons
e, '$.full_address'),'\\n',' '), '\\t',' '), get_json_object(json_response,
'$.open'), get_json_object(json_response,
'$.categories'),get_json_object(json_response,
'$.city'),get_json_object(json_response, '$.review_count'),
REGEXP_REPLACE(get_json_object(json_response, '$.name'),'\\t','
'),get_json_object(json_response,
'$.longitude'),get_json_object(json_response, '$.state'),
get_json_object(json_response, '$.stars'),get_json_object(json_response,
'$.latitude'),get_json_object(json_response, '$.attributes.Music.live'),
get_json_object(json_response,
'$.attributes.Music.background_music'),get_json_object(json_response,
'$.attributes.Music.karaoke'), get_json_object(json_response,
'$.attributes.Music.video'),get_json_object(json_response,
'$.attributes.Music.dj'), get_json_object(json_response,
'$.attributes.Music.jukebox'),get_json_object(json_response,
'$.attributes.Parking.lot'),get_json_object(json_response,
'$.attributes.Parking.street'), get_json_object(json_response,
'$.attributes.Parking.garage'),get_json_object(json_response,
'$.attributes.Parking.valet'),get_json_object(json_response,
'$.attributes.Parking.validated'),get_json_object(json_response,
'$.attributes.Happy Hour'),get_json_object(json_response,
'$.attributes.Price Range'), get_json_object(json_response,
'$.attributes.Wi-Fi'),get_json_object(json_response, '$.attributes.Ages
Allowed'),get_json_object(json_response, '$.attributes.Open 24
Hours'),get_json_object(json_response, '$.attributes.Dietary
Restrictions.vegan'),get_json_object(json_response, '$.attributes.Dietary
Restrictions.dairy-free'), get_json_object(json_response,
'$.attributes.Dietary Restrictions.halal'),get_json_object(json_response,
'$.attributes.Dietary Restrictions.gluten-free'),
get_json_object(json_response, '$.attributes.Dietary
Restrictions.kosher'),get_json_object(json_response, '$.attributes.Dietary
Restrictions.soy-free'), get_json_object(json_response,
'$.attributes.Dietary Restrictions.vegetarian');

```

```
--DATA CLEANING - DONE FOR ONLY (LAS
VEGAS,PITTSBURGH,CHARLOTTE,URBANA,PHOENIX,MADISON) CITIES
CREATE TABLE IF NOT EXISTS final_yelp_business
AS SELECT * FROM business_table_new WHERE
(city LIKE concat('%','Las Vegas','%') OR
city LIKE concat('Pittsburg','%') OR
city LIKE concat('%','Charlot','%') OR
city LIKE concat('Phoenix','%') OR
city IN ('Madison','Urbana')) AND
state IN ('NV','PA','NC','IL','AZ','WI');

--BUSINESS TABLE STORED WITH ONLY RESTAURANTS/Foodchain - REMAINING
CATEGORIES REMOVED
CREATE TABLE IF NOT EXISTS business_final_new
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_new_business"
AS SELECT * FROM final_yelp_business WHERE categories LIKE
concat('%','Food','%')
OR categories LIKE concat('%','Restaurant','%');
```

The above-mentioned code included cleaning of the business data file using hive. Here in our project we have filtered out the files based on the category and the cities. We have only considered 5 US cities and the Food and Restaurant business types. Remaining business types are not considered for our analysis.

```
CREATE EXTERNAL TABLE IF NOT EXISTS yelp_business_2 (
    json_response STRING)
STORED AS TEXTFILE
LOCATION "/user/calipriyanka/yelp";

CREATE TABLE IF NOT EXISTS business_table_new (
    business_id STRING,full_address STRING,open STRING,categories STRING,city STRING,review_cnt STRING,longitude STRING,state STRING,
    stars STRING,latitud STRING,live_music STRING,background_music STRING,karaoke STRING,video STRING,dj STRING,jukebox STRING,park_lot STRING,
    street STRING,garage STRING,valet STRING,validated STRING,Happ Hour STRING,Price_Range STRING,Wi_Fi STRING,Ages_Allowed STRING,Open_24_Hours STRING,vegan STRING,dairy_free STRING,hala STRING,gluten_free
    STRING,kosher STRING,soy_free STRING,vegetarian STRING);

FROM yelp_business
INSERT OVERWRITE TABLE business_table_new
SELECT get_json_object(json_response, '$.business_id'),REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_response, '$.full_address'),'\\n',' '), '\\t',' '),get_json_object(json_response, '$.open'),
get_json_object(json_response, '$.categories'),get_json_object(json_response, '$.city'),get_json_object(json_response, '$.review_count'),
REGEXP_REPLACE(get_json_object(json_response, '$.name'),'\\\\t',' '),get_json_object(json_response, '$.longitude'),get_json_object(json_response, '$.state'),get_json_object(json_response, '$.stars'),get_json_
object(json_response, '$.latitude'),get_json_object(json_response, '$.attributes.Music.Live'),
get_json_object(json_response, '$.attributes.Music.background_music'),get_json_object(json_response, '$.attributes.Music.karaoke'),
get_json_object(json_response, '$.attributes.Music.video'),get_json_object(json_response, '$.attributes.Music.dj'),get_json_object(json_response, '$.attributes.Music.jukebox'),get_json_object(json_respon
se, '$.attributes.Parking.lot'),get_json_object(json_response, '$.attributes.Parking.street'),
get_json_object(json_response, '$.attributes.Parking.garage'),get_json_object(json_response, '$.attributes.Parking.valet'),get_json_object(json_response, '$.attributes.Parking.validated'),get_json_obje
ct(json_response, '$.attributes.Happy Hour'),get_json_object(json_response, '$.attributes.Price Range'),
get_json_object(json_response, '$.attributes.Wi-Fi'),get_json_object(json_response, '$.attributes.Ages Allowed'),get_json_object(json_response, '$.attributes.Open 24 Hours'),get_json_object(json_response,
'$ .attributes.Dietary Restrictions.vegan'),get_json_object(json_response, '$.attributes.Dietary Restrictions.dairy-free'),
get_json_object(json_response, '$.attributes.Dietary Restrictions.hala'),get_json_object(json_response, '$.attributes.Dietary Restrictions.gluten-free'),
get_json_object(json_response, '$.attributes.Dietary Restrictions.kosher'),get_json_object(json_response, '$.attributes.Dietary Restrictions.soy-free'),
get_json_object(json_response, '$.attributes.Dietary Restrictions.vegetarian');

--DATA CLEANING - DONE FOR ONLY (LAS VEGAS,PITTSBURGH,CHARLOTTE,URBANA,PHOENIX,MADISON) CITIES
CREATE TABLE IF NOT EXISTS final_yelp_business AS
SELECT * FROM business_table_new WHERE
(city LIKE concat('%','Las Vegas','%') OR
city LIKE concat('Pittsburg','%') OR
city LIKE concat('%','Charlot','%') OR
city LIKE concat('Phoenix','%') OR
city IN ('Madison','Urbana')) AND
state IN ('NV','PA','NC','IL','AZ','MT');

--BUSINESS TABLE STORED WITH ONLY RESTAURANTS - REMAINING CATEGORIES REMOVED
CREATE TABLE IF NOT EXISTS business_final_new
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
STORED AS TEXTFILE
LOCATION "/user/calipriyanka/yelp_new_business"
AS
SELECT * FROM final_yelp_business WHERE categories LIKE concat('\','Food','\')
OR categories LIKE concat('\','Restaurant','\');
```

User Data file:

```
CREATE EXTERNAL TABLE IF NOT EXISTS yelp_user(
json_response STRING)
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_user";

CREATE TABLE IF NOT EXISTS users (
yelping_yr INT,yelping_mon INT,funny INT,cool INT,useful INT,review_count
INT,name STRING,user_id STRING,friends STRING, fans INT,average_stars
DECIMAL,type STRING,profile INT,cute INT,cfunny INT,plain INT,writer
INT,photos INT,hot INT,ccool INT, zmore INT,elite STRING);

--Extraction of User.json file and data cleaning
FROM yelp_user
INSERT OVERWRITE TABLE users
SELECT SUBSTR(get_json_object(json_response, '$.yelping_since'),1,4),
SUBSTR(get_json_object(json_response, '$.yelping_since'),6,8),
get_json_object(json_response, '$.votes.funny'),
get_json_object(json_response, '$.votes.cool'),
get_json_object(json_response, '$.votes.useful'),
get_json_object(json_response, '$.review_count'),
get_json_object(json_response, '$.name'),
get_json_object(json_response, '$.user_id'),
get_json_object(json_response, '$.friends'),
get_json_object(json_response, '$.fans'),
get_json_object(json_response, '$.average_stars'),
get_json_object(json_response, '$.type'),
get_json_object(json_response, '$.compliments.profile'),
get_json_object(json_response, '$.compliments.cute'),
get_json_object(json_response, '$.compliments.funny'),
get_json_object(json_response, '$.compliments.plain'),
get_json_object(json_response, '$.compliments.writer'),
get_json_object(json_response, '$.compliments.photos'),
get_json_object(json_response, '$.compliments.hot'),
get_json_object(json_response, '$.compliments.cool'),
get_json_object(json_response, '$.compliments.more'),
get_json_object(json_response, '$.elite');

--New table uploaded
CREATE TABLE IF NOT EXISTS user_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_new_user"
tblproperties ("skip.header.line.count"="0")
AS
SELECT * FROM users;
```



```

CREATE EXTERNAL TABLE IF NOT EXISTS yelp_user(
  json_response STRING)
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_user";

CREATE TABLE IF NOT EXISTS users (
  yelping_yr INT,yelping_mon INT,funny INT,cool INT,useful INT,review_count INT,name STRING,user_id STRING,friends STRING,fans INT,average_stars DECIMAL,type STRING,profile INT,cute INT,cfunny INT,plain INT
,writer INT,photos INT,hot INT,ccool INT,zmore INT,elite STRING);

--Extraction of User.json file and data cleaning
FROM yelp_user
INSERT OVERWRITE TABLE users
SELECT SUBSTR(get_json_object(json_response, '$.yelping_since'),1,4),
SUBSTR(get_json_object(json_response, '$.yelping_since'),6,8),
get_json_object(json_response, '$.votes.funny'),
get_json_object(json_response, '$.votes.cool'),
get_json_object(json_response, '$.votes.useful'),
get_json_object(json_response, '$.review_count'),
get_json_object(json_response, '$.name'),
get_json_object(json_response, '$.user_id'),
get_json_object(json_response, '$.friends'),
get_json_object(json_response, '$.fans'),
get_json_object(json_response, '$.average_stars'),
get_json_object(json_response, '$.type'),
get_json_object(json_response, '$.compliments.profile'),
get_json_object(json_response, '$.compliments.cute'),
get_json_object(json_response, '$.compliments.funny'),
get_json_object(json_response, '$.compliments.plain'),
get_json_object(json_response, '$.compliments.writer'),
get_json_object(json_response, '$.compliments.photos'),
get_json_object(json_response, '$.compliments.hot'),
get_json_object(json_response, '$.compliments.cool'),
get_json_object(json_response, '$.compliments.more'),
get_json_object(json_response, '$.elite');

--New table uploaded
CREATE TABLE IF NOT EXISTS user_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "g"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_new_user"
OP hive tables if existing
DROP TABLE yelp_user;
DROP TABLE users;
DROP table user_final;

```

The hql script for the business and user files can be executed by using the following shell command

```

$ hive -f business_table.hql

$ hive -f business_table.hql

```

Note: An **alternate** method to extract data from **yelp_academic_dataset_review.json** is also mentioned below. We have already extracted data using pig but the below method shows how we can extract using hive.

```

CREATE EXTERNAL TABLE IF NOT EXISTS yelp_review(
  json_response STRING)
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_review";

CREATE TABLE IF NOT EXISTS review_new
(
  funny INT,
  cool INT,
  useful INT,
  user_id STRING,
  review_id STRING,
  stars INT,
  dates STRING,
  text STRING,
  type STRING,
  business_id STRING);

```

```

FROM yelp_review
INSERT OVERWRITE TABLE review_new
SELECT get_json_object(json_response, '$.votes.funny'),
get_json_object(json_response, '$.votes.cool'),
get_json_object(json_response, '$.votes.useful'),
get_json_object(json_response, '$.user_id'),
get_json_object(json_response, '$.review_id'),
get_json_object(json_response, '$.stars'),
get_json_object(json_response, '$.date'),
REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_response,
'$.text'), '\n', ' '), '\t', ' '),
get_json_object(json_response, '$.type'),
get_json_object(json_response, '$.business_id');

CREATE TABLE IF NOT EXISTS review_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_review_hive"
tblproperties ("skip.header.line.count"=" ")
AS
SELECT * FROM review_new;

```

```

CREATE TABLE IF NOT EXISTS review
( funny INT,
cool INT,
useful INT,
user_id STRING,
review_id STRING,
stars INT,
dates STRING,
text STRING,
type STRING,
business_id STRING);

FROM yelp_review
INSERT OVERWRITE TABLE review
SELECT get_json_object(json_response, '$.votes.funny'),
get_json_object(json_response, '$.votes.cool'),
get_json_object(json_response, '$.votes.useful'),
get_json_object(json_response, '$.user_id'),
get_json_object(json_response, '$.review_id'),
get_json_object(json_response, '$.stars'),
get_json_object(json_response, '$.date'),
REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_response, '$.text'), '\n', ' '), '\t', ' '),
get_json_object(json_response, '$.type'),
get_json_object(json_response, '$.business_id');

CREATE TABLE IF NOT EXISTS review_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_review_hive"
tblproperties ("skip.header.line.count"=" ")
AS
SELECT * FROM review;

```

The hql script for the review data file can be executed by using the following shell command.

```
$ hive -f review.hql
```

Note: When you run the 3 hive scripts, the following tables are created into hive. These are the tables that used to perform all the analysis. The yelp_review, yelp_user, yelp_business contains the json data in the unstructured format and the review_final, user_final, business_final_review contains tables that are in structured format. Hence enter the below shown command to see if all these tables exists in the hive database.

- yelp_review , review_new,review_final,
- yelp_user, users,user_final,
- yelp_business, business_table_new,final_yelp_business,business_final_new

```
$ hive
hive> show tables;
```

Note: In the hive shell, you can compare the content of the tables and think about why **structured table** is better for querying than the **unstructured table** by using the following command:

```
$ select * from yelp_review limit 1;
$ select * from review_final limit 10;
```

Analysis 1: In which year Yelp has got maximum number of users?

Analysis have been made to find in which year yelp has got maximum users using the yelp app or yelp.com to give their reviews. Below query has been written to achieve the desired results using hive query.

```
hive> select yelping_since, count(user_id) as c1 from user_final group by
yelping_since order by yelping_since;
```

Analysis 2: Forecast of users joining Yelp in coming years.

Analysis have been made to perform a forecast to see the no of new yelp users over the next 3 years. Below query has been written to achieve the desired results using hive query.

```
hive> select yelping_since, count(user_id) as c1 from user_final group by
yelping_since order by yelping_since;
```

Analysis 3: Who are the most active users on Yelp ?

Analysis have been made to find who are the most active users, i.e. who have written maximum number of reviews.

```
hive> select r.user_id, u.name, count(r.review_id) as r1 from review r
join userxyz u on r.user_id=u.userid group by user_id,name order by r1
desc limit 100;
```

Analysis 4: Which users' review are most popular on Yelp ?

Analysis have been made to find which users' reviews are most popular based on the votes they've got.

```
hive> select u.name, r.review_id, r.total from review r join userxyz u  
on r.user_id=u.userid order by r.total desc limit 100;
```

Analysis 5: Which city has the maximum no of closed business?

This analysis was performed to find which city has the maximum no of closed businesses. This would be an advantage for those budding entrepreneurs who thinking to start up a new business, they can easily choose which are the cities in US they shouldn't starting a new business. Below query has been written to achieve the desired results using hive query.

```
hive> select city, count(open) as c1 from business_final_new where  
open="FALSE" group by city order by c1 desc limit 20;
```

Analysis 6: Sentiment Analysis performed on the top 8 business chains in US based on user reviews.

User reviews are the most valuable asset on yelp. Yelp thrives on the reviews given by users and hence we have used the reviews given by users on the top 8 food chains in US. Below are the steps followed in order to achieve the desired results.

The entire code has been written in hive script. The important aspect here is we need to download a dictionary with the command shown below in order to match the positive words and negative words from the review text. You need to download the dictionary using the **wget** shell command.

```
$ cd project  
$ wget -O dictionary.tsv  
https://s3.amazonaws.com/hipicdatasets/dictionary.tsv
```

Once we have downloaded we have to create a directory in HDFS in order to store the dictionary file. Use the following commands to upload dictionary to a new directory in HDFS and to list the contents.

```
$ hdfs dfs -mkdir /user/cali/priyanka/senti_test/data/tables/dict  
$ hdfs dfs -put dictionary.tsv  
/user/cali/priyanka/senti_test/data/tables/dict/  
$ hdfs dfs -ls /user/cali/priyanka/senti_test/data/tables/dict
```

Once we have uploaded the dictionary file we have to convert it into a structured table by defining the schema in hive. The code for this has been written in the hive script **dictionary.hql**.

```
$ vi dictionary.hql
```

```
CREATE EXTERNAL TABLE if not exists dictionary (  
  type string,  
  length int,  
  word string,  
  pos string,  
  stemmed string,  
  polarity string)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t'  
STORED AS TEXTFILE  
LOCATION '/user/cali/priyanka/senti_test/data/tables/dict';
```

Now we need to create a table which contains only the top 8 food chains and also join the business table and the review table in order to find the review text for those business ids. We will join the business table and the review table using the business id field which is present in both – review and business table. We also need to create a structured table for the dictionary since it is easier to query a structured table than a unstructured one. The below hive query is written in hive script - **sentianalysis.hql**.

```
$ vi sentianalysis.hql
```

```
--Analysis done only for top 8 food chains in US  
CREATE TABLE IF NOT EXISTS business_top AS  
SELECT * FROM business_final_new WHERE name LIKE  
CONCAT('%','starbucks','%')  
OR name LIKE CONCAT('%','Starbucks','%')  
OR name IN ('Burger King','Burger King ','Wendys','Wendy's','Taco Bell',  
'Pizza Hut','KFC','McDonald's','McDonalds','McDonald's - MGM Grand The  
District Food Court','Subway');
```

```

--Review table and business table joined to obtain review text for sentiment
analysis
CREATE TABLE IF NOT EXISTS business_review
AS SELECT
b.business_id,b.city,b.review_cnt,b.name,b.longitude,b.state,b.latitude,
r.user_id,r.review_id,r.dates,r.text
FROM business_top b INNER JOIN review_new r
ON b.business_id = r.business_id;

--Create views to separate the words and sentences in the review text
create view IF NOT EXISTS br1
AS SELECT business_id,text1
FROM business_review
lateral view explode(sentences(lower(text))) dummy as text1;

--Create view to seperate the group of words
create view IF NOT EXISTS br2
AS SELECT business_id,text
FROM br1
lateral view explode( text1 ) dummy as text;

--Compare the reviews words with dictionary words and store the polarity
create view IF NOT EXISTS br3
AS SELECT business_id,br2.text,
case d.polarity
  when 'negative' then -1
  when 'positive' then 1
  else 0 end as polarity
from br2 left outer join dictionary d on br2.text = d.word;

--The Optimized Row Columnar (ORC) file format provides a highly efficient way
to store Hive data. It was designed to overcome limitations of the other Hive
file formats. Using --ORC files improves performance when Hive is reading,
writing, and processing data

--Summation of polarity to understand the whole review
create table IF NOT EXISTS review_sentiment
stored as orc
AS SELECT business_id,
case
  when sum( polarity ) > 0 then 'positive'
  when sum( polarity ) < 0 then 'negative'
  else 'neutral' end as sentiment
from br3 group by business_id;

--The Optimized Row Columnar (ORC) file format provides a highly efficient way
to store Hive data. It was designed to overcome limitations of the other Hive
file formats. Using --ORC files improves performance when Hive is reading,
writing, and processing data

```

```

--Summation of polarity to understand the whole review
create table IF NOT EXISTS review_sentiment
stored as orc
AS SELECT business_id,
  case
  when sum( polarity ) > 0 then 'positive'
  when sum( polarity ) < 0 then 'negative'
  else 'neutral' end as sentiment
from br3 group by business_id;

--Output the join table based on business id for the top 8 Food chains
CREATE TABLE IF NOT EXISTS yelp_sentiment
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_sentiment"
AS SELECT
  br.business_id,br.city,br.name,br.longitude,br.state,br.latitude,
  br.user_id,br.review_id,
  case rs.sentiment
  when 'positive' then 2
  when 'neutral' then 1
  when 'negative' then 0
  end as sentiment
FROM business_review br LEFT OUTER JOIN review_sentiment rs
on br.business_id = rs.business_id;

```

Run the **dictionary.hql** and **sentianalysis.hql** scripts by using the below mentioned shell commands

```

$ hive -f dictionary.hql
$ hive -f sentianalysis.hql

```

The analyzed data has been saved into the **yelp_sentiment** table on which visualization is done using the Excel 3D map.

Analysis 7: Best suited restaurants for tourists

This analysis is done to help tourist to find the best restaurant in a new place. We made our analysis with respect to the price range available on the yelp dataset for the 5 cities. Then we checked only for those restaurants that are currently open with has parking and wifi.

The code for this analysis is written in **touristanalysis.hql** script. Run the below mentioned code to receive the desired output for the analysis.

```
$ vi touristanalysis.hql
```

```
CREATE TABLE IF NOT EXISTS tourist
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_tourist_analysis"
AS SELECT name,open,categories,city,review_cnt,state,park_lot,
street,garage,valet,validated,Happy_Hour,Price_Range,Wi_Fi
FROM business_final_new WHERE park_lot IS NOT NULL AND
street IS NOT NULL AND
garage IS NOT NULL AND
valet IS NOT NULL AND
validated IS NOT NULL AND
Happy_Hour IS NOT NULL AND
Price_Range IS NOT NULL AND
Wi_Fi IS NOT NULL;
```

Run the **touristanalysis.hql** script by using the below mentioned shell commands

```
$ hive -f touristanalysis.hql
```

```
CREATE TABLE IF NOT EXISTS tourist
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
STORED AS TEXTFILE
LOCATION "/user/calipriyanka/yelp_tourist_analysis/"
SELECT name,open,categories,city,review_cnt,state,park_lot,
street,garage,valet,validated,HappY_Hour,Price,Range,Wi_Fi
FROM business_final_new WHERE park_lot IS NOT NULL AND
street IS NOT NULL AND
garage IS NOT NULL AND
valet IS NOT NULL AND
validated IS NOT NULL AND
HappY_Hour IS NOT NULL AND
Price_Range IS NOT NULL AND
Wi_Fi IS NOT NULL;
```

--Upload Tourist Analysis

~~~~~

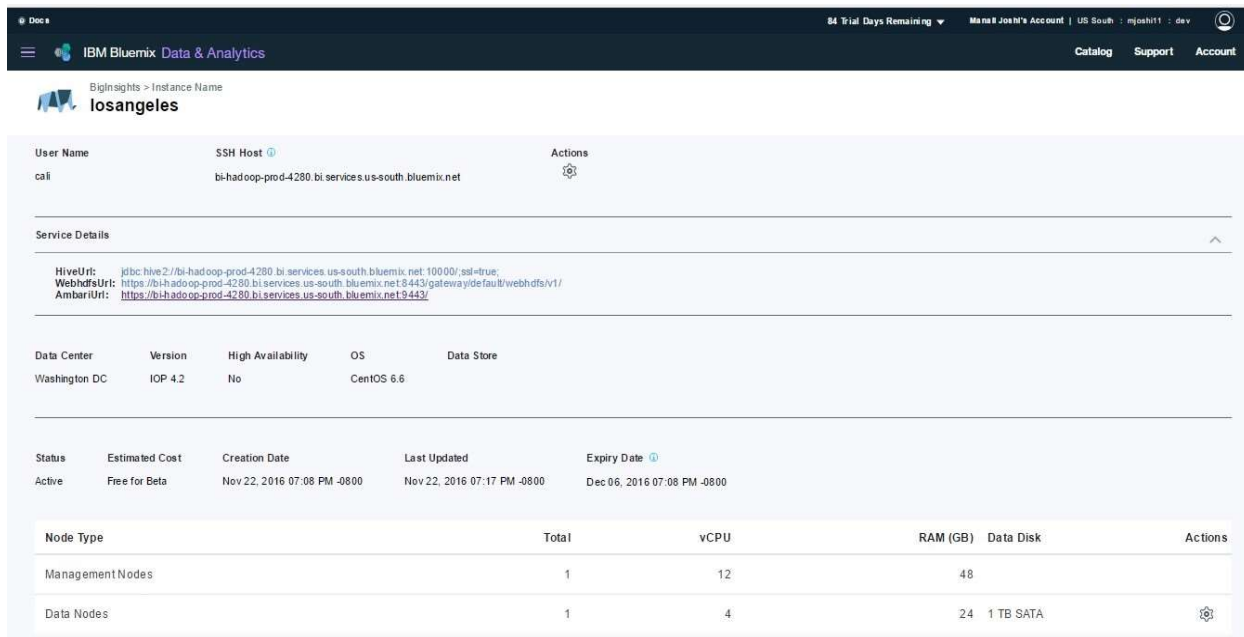
```
"touristanalysis.hql" 18L, 528C
```

The analyzed data has been visualized using tableau.



## Loading Data into Excel:

Now go to the BigInsights cluster page to open Ambari web page.



The screenshot displays the IBM Bluemix Data & Analytics console interface. At the top, the header shows 'IBM Bluemix Data & Analytics' and 'BigInsights > Instance Name losangeles'. Below this, there's a table with columns: User Name, SSH Host, and Actions. The first row shows 'cali' as the user name and 'bi-hadoop-prod-4280.bi.services.us-south.ibm.com' as the SSH host. Below this table is a 'Service Details' section with a dropdown arrow. It contains three lines of text: 'HiveUrl: jdbc:hive2://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:10000/?ssl=true;', 'WebhdfsUrl: https://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:8443/gateway/default/webhdfs/v1/', and 'AmbariUrl: https://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:8443/'. Below this is another table with columns: Data Center, Version, High Availability, OS, and Data Store. The first row shows 'Washington DC', 'IOP 4.2', 'No', 'CentOS 6.6', and an empty cell. Below this is a table with columns: Status, Estimated Cost, Creation Date, Last Updated, and Expiry Date. The first row shows 'Active', 'Free for Beta', 'Nov 22, 2016 07:08 PM -0800', 'Nov 22, 2016 07:17 PM -0800', and 'Dec 06, 2016 07:08 PM -0800'. At the bottom is a table with columns: Node Type, Total, vCPU, RAM (GB), Data Disk, and Actions. The first row shows 'ManagementNodes', '1', '12', '48', and an empty cell. The second row shows 'Data Nodes', '1', '4', '24', '1 TB SATA', and an empty cell.

| User Name | SSH Host                                         | Actions |
|-----------|--------------------------------------------------|---------|
| cali      | bi-hadoop-prod-4280.bi.services.us-south.ibm.com |         |

Service Details

HiveUrl: jdbc:hive2://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:10000/?ssl=true;  
WebhdfsUrl: https://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:8443/gateway/default/webhdfs/v1/  
AmbariUrl: https://bi-hadoop-prod-4280.bi.services.us-south.ibm.com:8443/

| Data Center   | Version | High Availability | OS         | Data Store |
|---------------|---------|-------------------|------------|------------|
| Washington DC | IOP 4.2 | No                | CentOS 6.6 |            |

| Status | Estimated Cost | Creation Date               | Last Updated                | Expiry Date                 |
|--------|----------------|-----------------------------|-----------------------------|-----------------------------|
| Active | Free for Beta  | Nov 22, 2016 07:08 PM -0800 | Nov 22, 2016 07:17 PM -0800 | Dec 06, 2016 07:08 PM -0800 |

| Node Type       | Total | vCPU | RAM (GB) | Data Disk | Actions |
|-----------------|-------|------|----------|-----------|---------|
| ManagementNodes | 1     | 12   | 48       |           |         |
| Data Nodes      | 1     | 4    | 24       | 1 TB SATA |         |

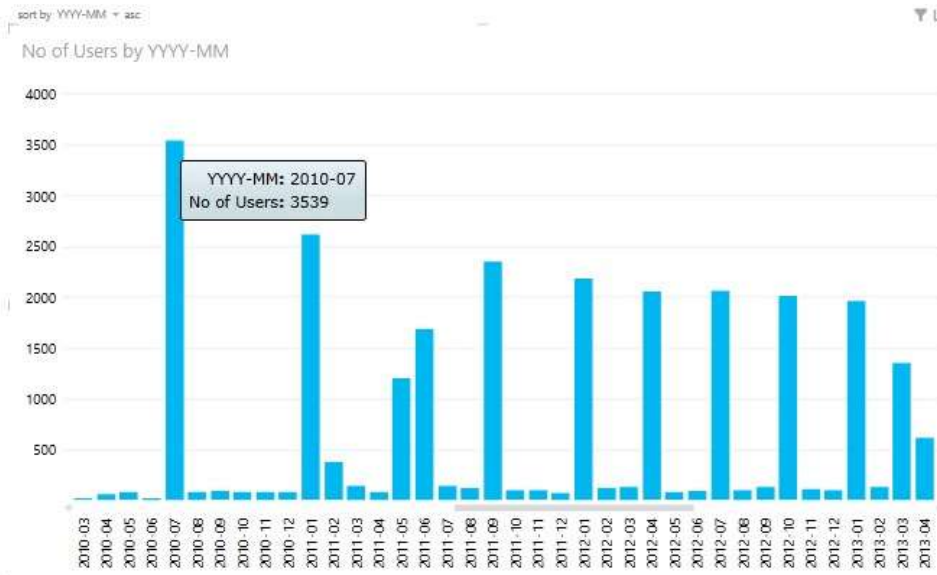
And, you need to go and open “File Browser” to find out the analyzed data in the respective directories as mentioned in the codes. You need to click on the file to download to your local computer in order to open it in Excel.

You have to open the file in Excel with delimiter (separator) specified while creating the analyzed data file on HDFS.

## Analysis 1: In which year Yelp has got maximum number of users?

The downloaded file is opened in Excel and we have used the Power View column under the Insert tab for visualization. After that select stacked columns.

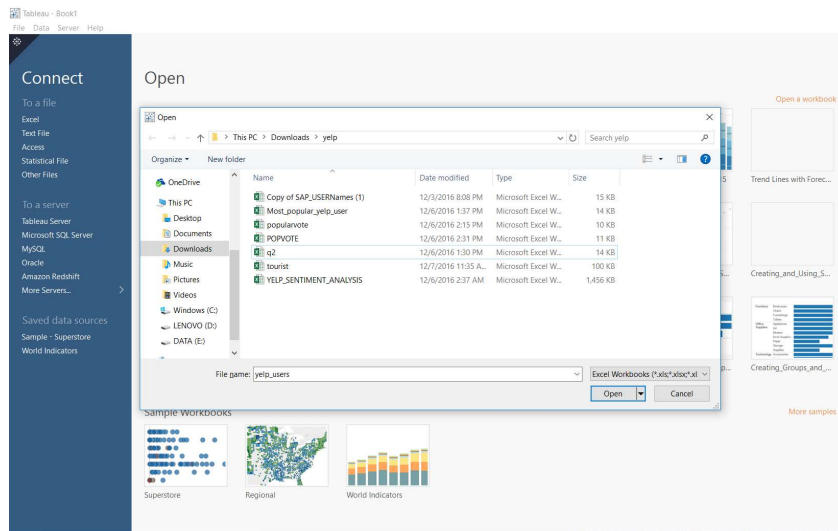
The output is shown in a vertical bar graph to represent the maximum user over the years from 2004-2016.



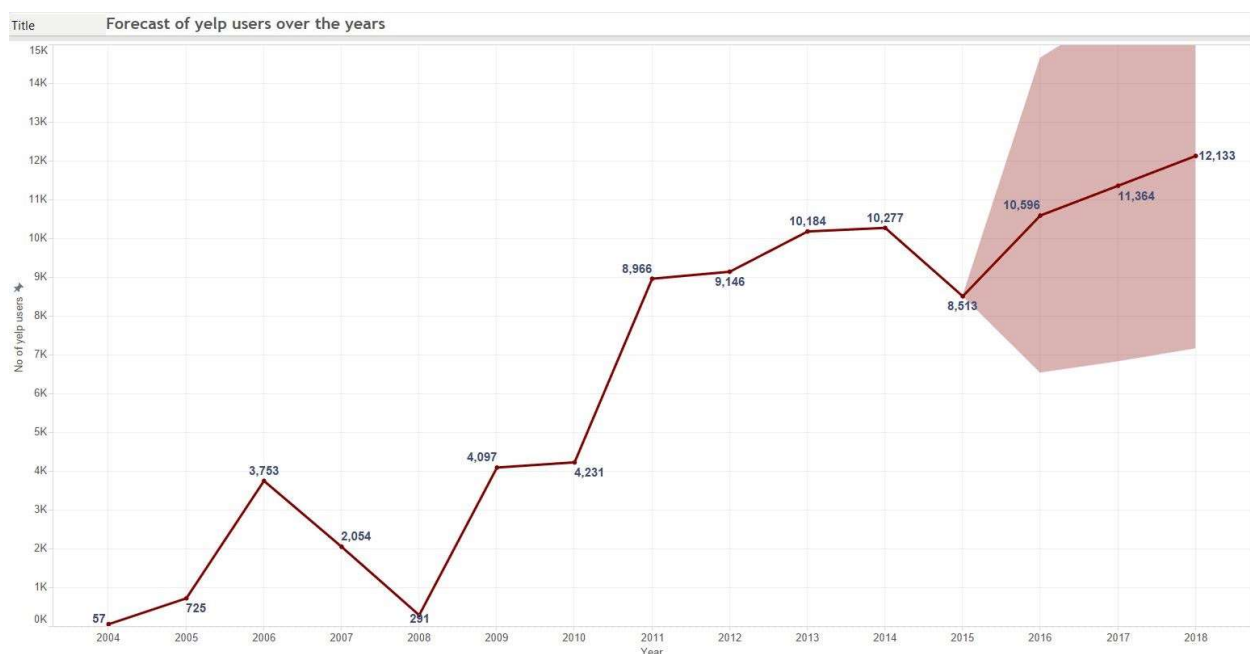
## Analysis 2: Forecast of users joining Yelp in coming years.

The forecast was done using tableau. The downloaded Excel file was opened in tableau in the following manner and a forecast was made to find the no of yelp users in the upcoming years. Forecast was made for the next 3 years. Here 2016 is also included in the forecast because we have data for 2016 up to June and while performing forecast the remaining 6 months are also considered.

The below step shows how to import the excel file with the analyzed data to Tableau.



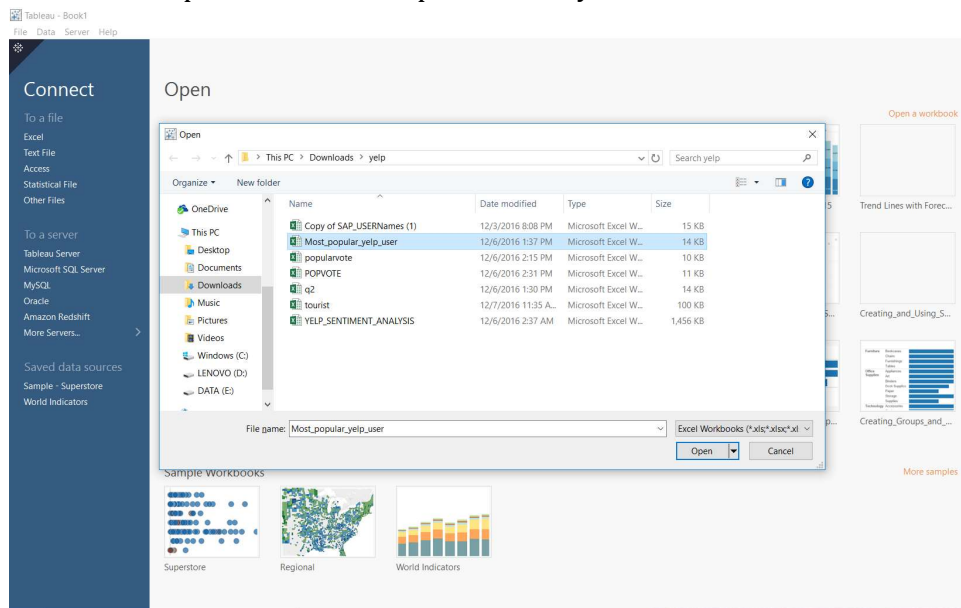
The forecast for the no of yelp users for the next 3 years are as follows:



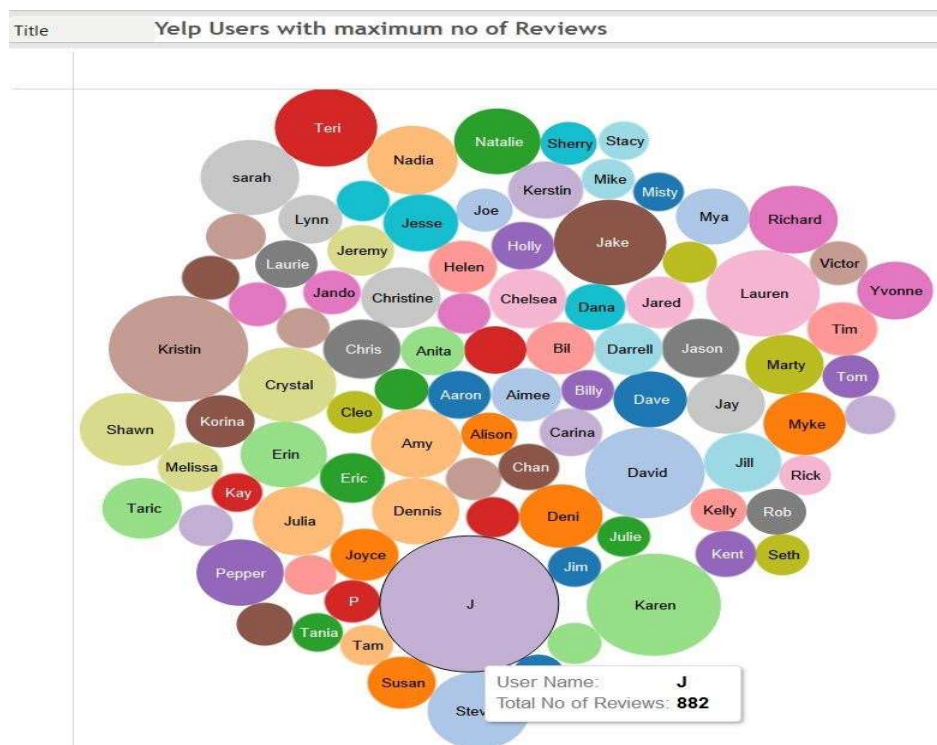
## Analysis 3: Who are the most active users on Yelp?

The desired file was downloaded to Excel and opened in the tableau as shown below. In tableau, we have used the bubble chart to represent the most active users. The bigger the size of the bubble the most active the user is.

The below step shows how to import the analyzed excel file to tableau

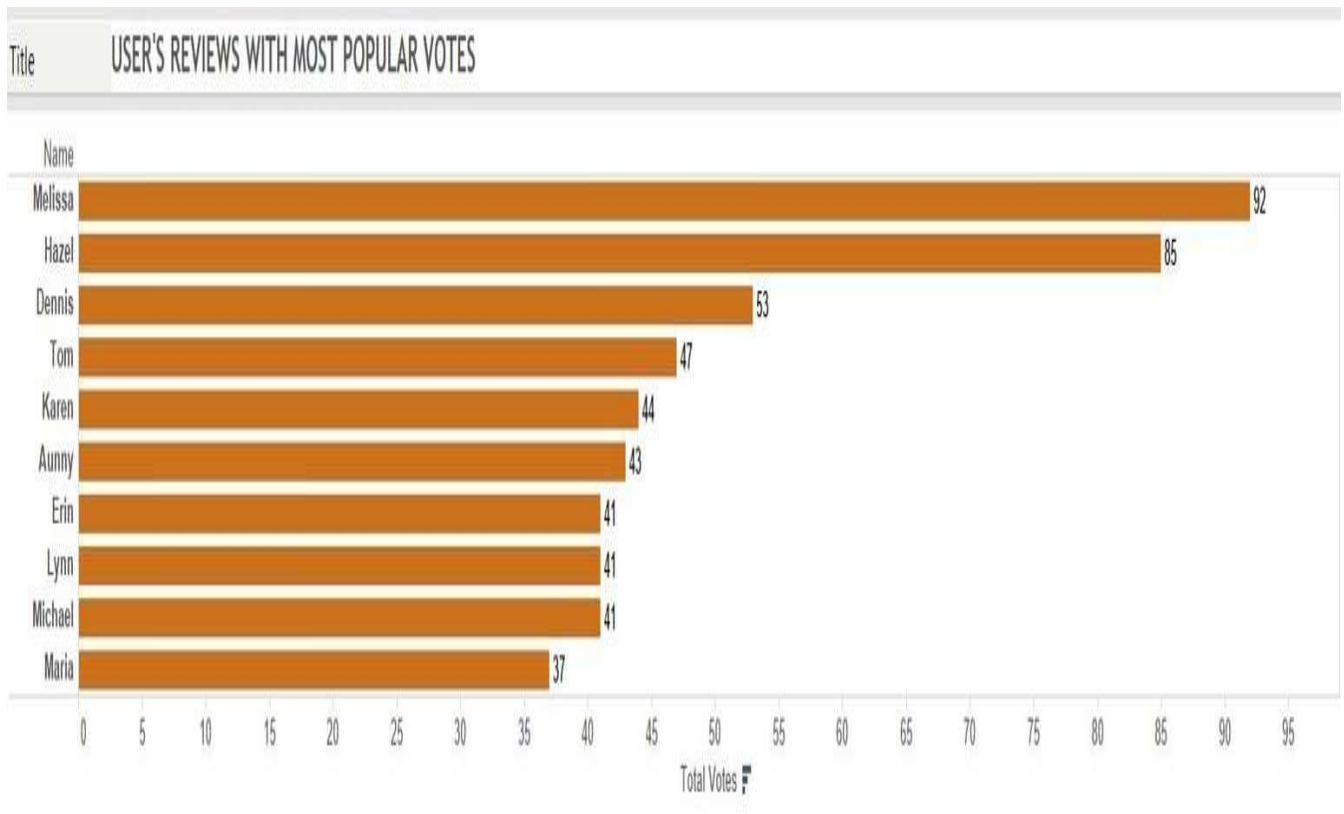


The bubble chart output shows that user “J” is the most active user with 882 reviews:



## Analysis 4: Which users' reviews are the most popular?

Analysis have been made to find which users' reviews are most popular based on the votes they've got. The desired file was downloaded to Excel and opened in the tableau as shown below. In tableau, we have used the horizontal bar chart to represent the most popular user's review. Here, we can see that user Melissa's review has got highest no. of votes i.e. 92, followed by Hazel by 85 and Dennis by 53, and so on.



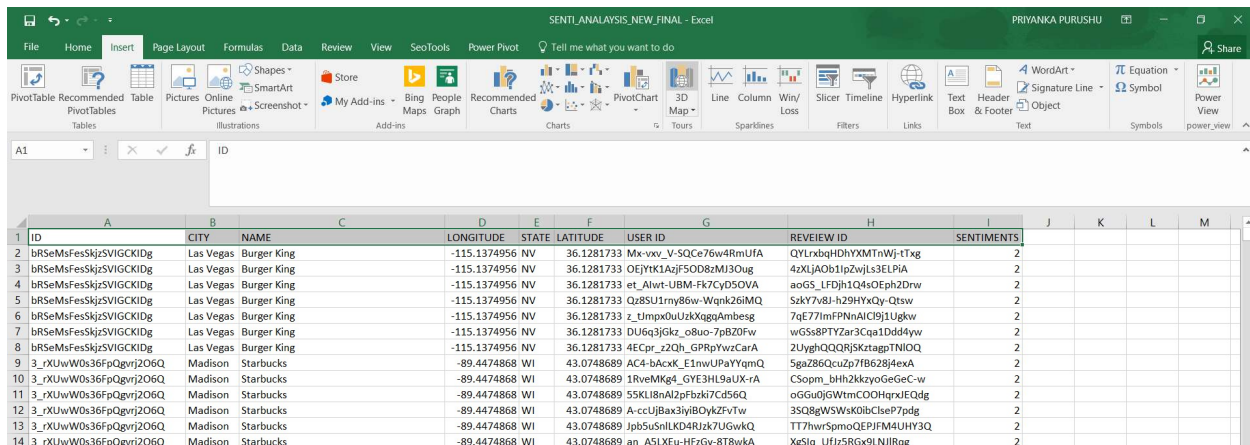
## Analysis 5: Which city has the maximum no of closed business?

Open the downloaded Analysis table in Excel with delimiter (separator) “comma”. Then, go to “insert” tab to find out the menu “3D Maps” enabled. You need to click and open “3D Map”. If it complains that 3D Map cannot be open, then you need to make sure if you insert headers into the first row. Now you select columns you want to filter.



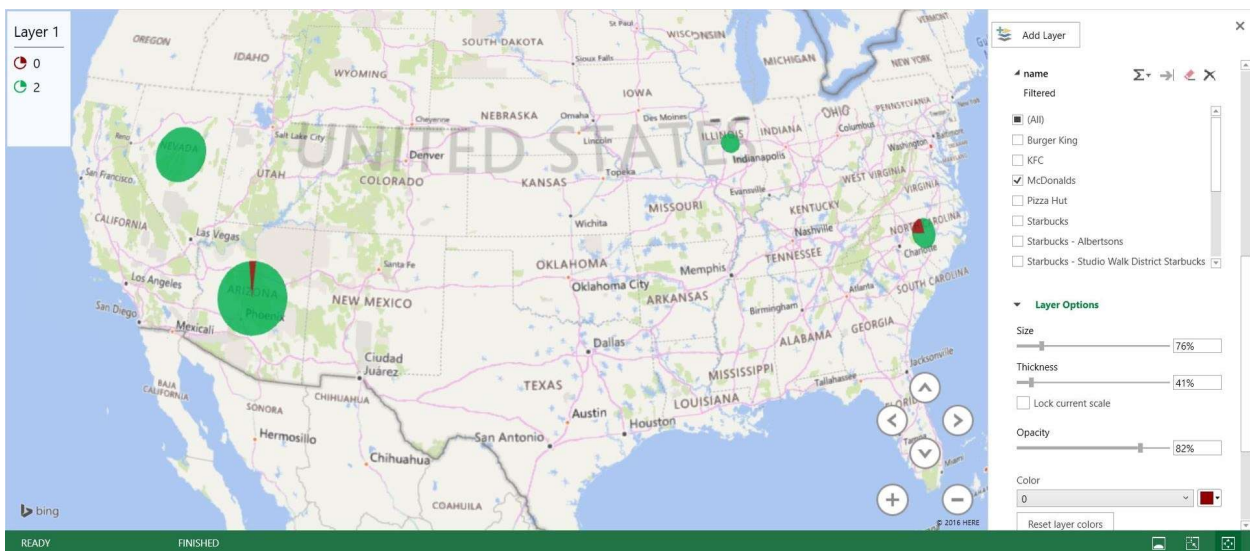
## Analysis 6: Sentiment analysis based on review text for the top 8 food chains in US.

Open the yelp\_sentiment table in Excel with delimiter (separator): “#”. Then, go to “insert” tab, click and open “3D Map”. Now you select columns you want to filter.



| ID | CITY      | NAME        | LONGITUDE    | STATE | LATITUDE   | USER_ID                | REVIEW ID              | SENTIMENTS |
|----|-----------|-------------|--------------|-------|------------|------------------------|------------------------|------------|
| 1  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | Mx-vw_V-SQCe76w4RmUfA  | CYLnbqHdHYXMTnWj-tTxg  | 2          |
| 2  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | 0EjYKIAzJF50D8zMI3Oug  | 4xXJAOb31pZwJLs3ELPIA  | 2          |
| 3  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | et_Alwt-UBM-Fk7CyD5OVA | aoGS_LFDjt1Q4wOEph2Dnw | 2          |
| 4  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | Qz8SU1rny86w-Wqmk26IMQ | SdKY7v8J-h29HYxQy-Qtsw | 2          |
| 5  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | _tTmnpDuUzKXaggAmbesg  | 7qE77lmFPNnAIC9j1Ugkw  | 2          |
| 6  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | DUEq3jGkz_o8u-7pBZ0Fw  | wGSs8PTYZar3Cqa1Ddd4yw | 2          |
| 7  | Las Vegas | Burger King | -115.1374956 | NV    | 36.1281733 | 4ECpr_x2Qh_GPRpYwzCarA | 2UyghQQQRjSKrtappTNIOQ | 2          |
| 8  | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | AC4-bAcxK_E1nwUPaYYqmQ | SgaZ86Qcu27fB628j4exA  | 2          |
| 9  | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | 1RveMKg4_GYE3Hl9aUX-rA | CScpm_bHh2kkyoGeGeC-w  | 2          |
| 10 | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | 55KLI8nAl2pFbk7Cd56Q   | oGGuJGWtmCOOHqrxJEQdg  | 2          |
| 11 | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | A-ccUjBax3iy8OykZfvTw  | 3S08gWSWsk0ibCseP7pdg  | 2          |
| 12 | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | Jpb5uSnLKD4Rjzk7UGwkQ  | TT7hwrSpmoQEPJFM4UHY3Q | 2          |
| 13 | Madison   | Starbucks   | -89.4474868  | WI    | 43.0748689 | an_ASIXEu-HFrGy-8T8wkA | XgSlq_Ufjz5RGx9LNIJrag | 2          |

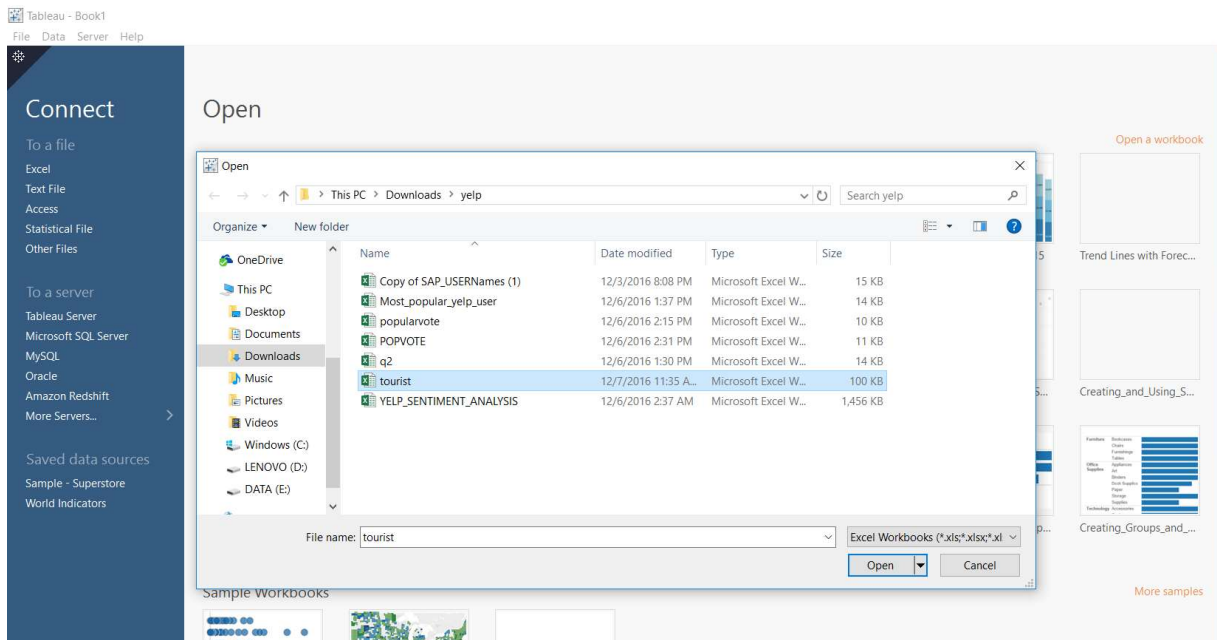
You will see the following 3D map:



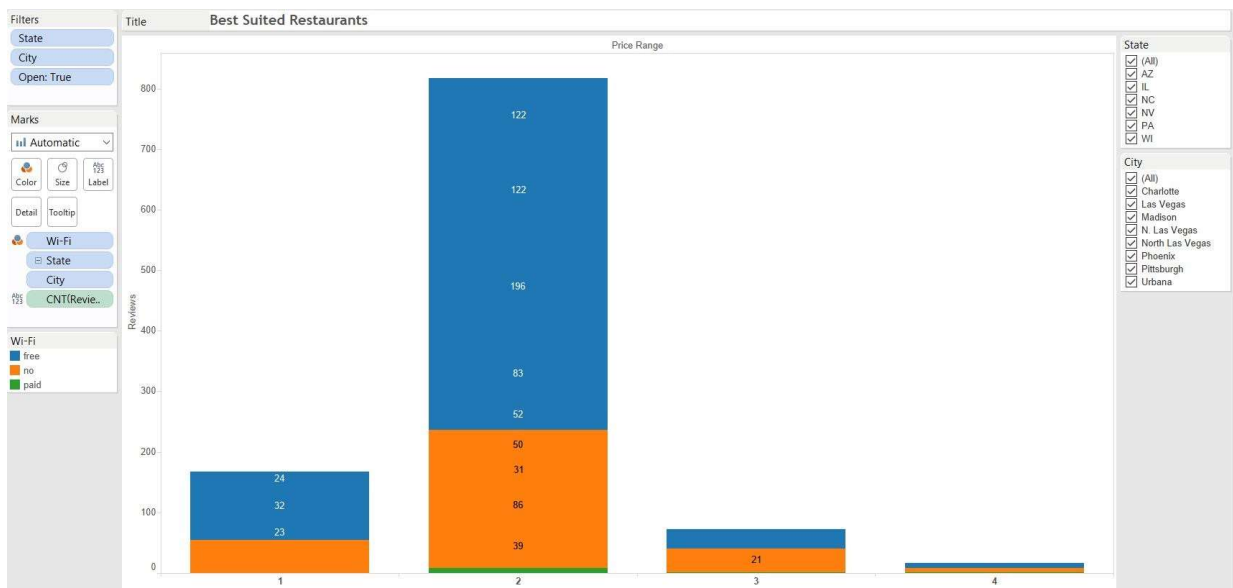


## Analysis 7: Best suited restaurants for tourists

Open the tourist table in Excel with delimiter (separator): “#” and save the file. Open tableau and download the file as show and the visualization was done using a bar stacked bar graph. The below step shows how to import the analyzed excel file to tableau.



The output of our analysis was in the horizontal stacked bar graph.





## Summary:

During our analysis, we learned that, Yelp got the max no of users in the month of July 2010. Further, we displayed a forecast of new Yelp users joining in the coming years. We displayed Top 100 most active users in the Bubble chart. Top 10 user reviews which got more popular on Yelp. On the 3D Maps, we visualized that, Las Vegas city has the max no of closed businesses. Sentiment analysis which includes positive, neutral & negative reviews about top 5 restaurants based on geo location. We only displayed for McDonalds. List of Best suited restaurants for tourists in terms of Price range, City, State, Wi-Fi Connection and Parking.

## Github URL:

<https://github.com/mjoshi1110/cali/>

## References:

1. <http://hortonworks.com/hadoop-tutorial/how-to-refine-and-visualize-sentiment-data/>
2. <https://calstatela.edu/jwoo5/classes/2016/fall/cis5200/>
3. [https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge)
4. <https://console.ng.bluemix.net/catalog/services/BigInsightss-for-apache-hadoop>
5. <https://www.tableau.com>