CIS 4560 Term Project Tutorial

Authors: Mohsen Alam; Benjamin Cevallos; Oscar Flores; Randall Lunetto; Kotaro Yayoshi

Instructor: Jongwook Woo

Date: 12/09/2020

Lab Tutorial

UNAME (UNAME@calstatela.edu)

12/09/2020

Analyzing the Yelp Dataset: Descriptive and Semantic Analytics Using Hive in Oracle Cloud Big Data Compute Edition

---

Objectives

List what your objectives are. In this hands-on lab, you will learn how to:

- Download and upload big data with limited storage
- Process multiple files while understanding dataset owner's documentation and guidelines
- Combine multiple files into a coherent dataset
- Hive's different functions to produce sensible outputs that have business value

- Use Excel in combination to Hive's output to produce visual analytics

Platform Spec

- Oracle Big Data Compute Edition
- # of CPU cores: 12
- # of nodes: 3
- Total Memory Size: 180 GB
- Storage: 957 GB

We will use Oracle Cloud Big Data Compute Edition (BDCE) to use Hive for analyzing the Yelp Dataset. This dataset has five JSON files: business, checkin, review, tip, and user.

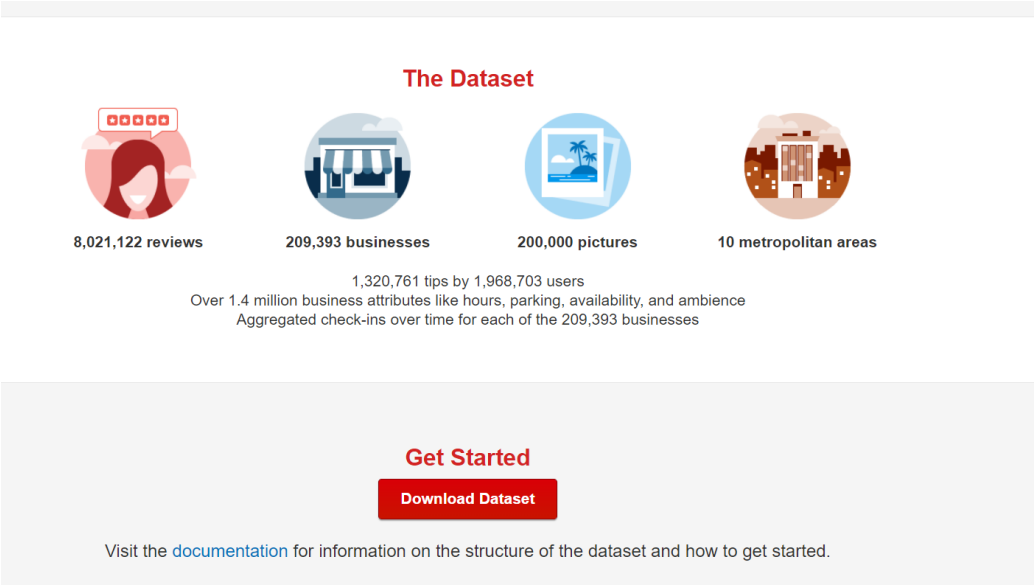
1. business.json: This file includes business information such as location data, categories, and attributes.
2. checkin.json: This file holds all check-ins dates and times for businesses.
3. review.json: This file includes reviews with reviewers' and business' information.
4. tip.json: Tips are short feedbacks from users, which are recorded in this file.
5. user.json: This file holds information about reviewers.

We will use Hive's JOIN feature to combine these files to produce results that have business value. Also, we will use Hive's text processing features to understand reviewers' sentiment and how they vary in different regions.

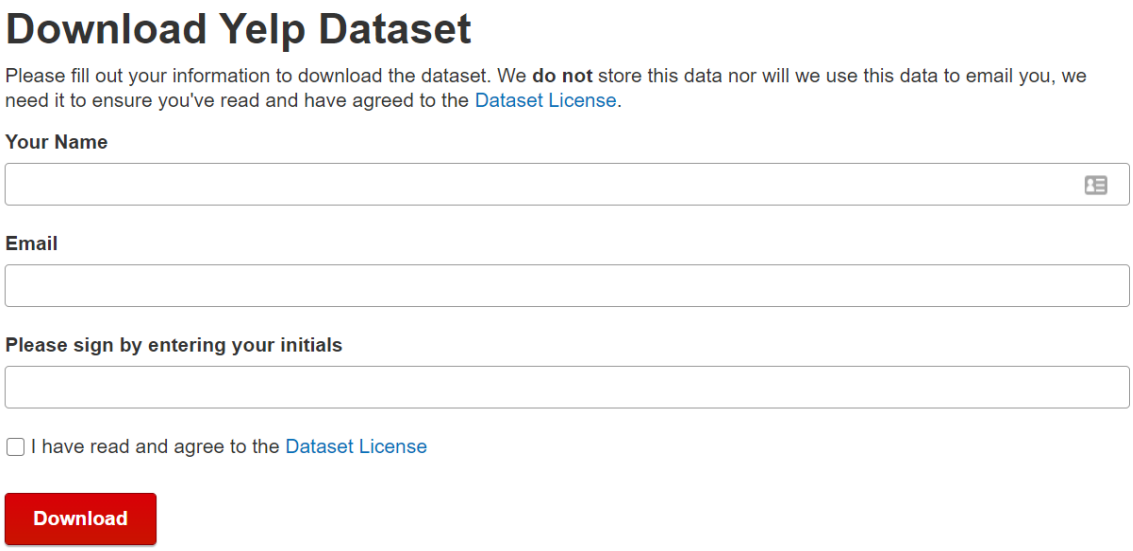
Step 1: Downloading and Preparing Dataset

We will set up our lab environment in this stage. Throughout the lab, we assume that your BDCE server's and Hive Server's usernames are the same. In this tutorial, the username is denoted as UNAME, please replace UNAME with your username.

1. Use a web browser in your lab or personal computer to go to this link: https://www.yelp.com/dataset/ and click on the "Download Dataset" button.



2. Fill up your name, email, and initials and agree to the Dataset License, then click on the "Download" button



3. Click on the "Download JSON" button quickly since the link expires after 30 seconds. Download the "yelp_dataset.tar" file to your Downloads folder.

Download The Data

The links to download the data will be valid for **30 seconds.**

JSON	Photos	COVID-19 Data
Download JSON	Download photos	Download COVID-19 data
4.5GB compressed 9.8GB uncompressed	7.0GB compressed 7.2GB uncompressed	14MB compressed 69MB uncompressed
1 .tgz file compressed 1 .pdf file and 5 .json files uncompressed	1 .tar file compressed 1 .json file, 1 text file, 1 .pdf and 1 folder containing 200,000 photos	1 .tgz file compressed 1 folder containing 1 .json file and 1 .pdf
For more information on the JSON dataset, visit the main dataset documentation page.		For more information on the COVID-19 dataset, visit the Yelp Engineering Blog post.

4. Use the SCP command in Git Bash or other terminal to upload the downloaded file to BDCE server as below (IP address used here is 129.150.79.19, which may change. Use proper IP address):
   scp Downloads/yelp_dataset.tar UNAME@129.150.79.19:/home/UNAME/
5. Use web browser to browse this link and download the state_locations.txt file in your Downloads folder:
   https://drive.google.com/uc?id=1dFrIcQuBhaANRHHvnzbthfU3HHVDRy7Y&export=download
6. Use the SCP command in Git Bash or other terminal to upload the downloaded file to BDCE server as below:
   scp Downloads/state_locations.txt UNAME@129.150.79.19:/home/UNAME/
7. Open terminal and connect to the cloud BDCE server with appropriate password:
   ssh UNAME@129.150.79.19
8. Once successfully logged in, you should see something similar like this:

```
Last login: Fri Nov 6 20:04:55 2020 from xxxxx.spectrum.com
-bash-4.1$
```

9. Run this command to verify that you have the yelp_dataset.tar file:
   ls -hl
10. Run the following commands to create directories in HDFS filesystem:
    hdfs dfs -mkdir yelp
    hdfs dfs -mkdir yelp/business
    hdfs dfs -mkdir yelp/checkin
    hdfs dfs -mkdir yelp/review
    hdfs dfs -mkdir yelp/tip
    hdfs dfs -mkdir yelp/user
    hdfs dfs -mkdir yelp/states
    hdfs dfs -mkdir yelp/dictionary
11. Run these codes to extract individual files from the yelp_dataset.tar file and upload the file to HDFS filesystem
    tar -xvf yelp_dataset.tar ./yelp_academic_dataset_business.json
    hdfs dfs -put yelp_academic_dataset_business.json yelp/business
    rm yelp_academic_dataset_business.json

tar -xvf yelp_dataset.tar ./yelp_academic_dataset_checkin.json
hdfs dfs -put yelp_academic_dataset_checkin.json yelp/checkin
rm yelp_academic_dataset_checkin.json

tar -xvf yelp_dataset.tar ./yelp_academic_dataset_review.json
hdfs dfs -put yelp_academic_dataset_review.json yelp/review
rm yelp_academic_dataset_review.json

tar -xvf yelp_dataset.tar ./yelp_academic_dataset_tip.json
hdfs dfs -put yelp_academic_dataset_tip.json yelp/tip
rm yelp_academic_dataset_tip.json

tar -xvf yelp_dataset.tar ./yelp_academic_dataset_user.json
hdfs dfs -put yelp_academic_dataset_user.json yelp/
rm yelp_academic_dataset_user.json

12. Use these commands to verify whether the files are in the right place:
    hdfs dfs -ls -h yelp/business
    hdfs dfs -ls -h yelp/checkin
    hdfs dfs -ls -h yelp/review
    hdfs dfs -ls -h yelp/tip
    hdfs dfs -ls -h yelp/user
13. If the files are successfully uploaded to HDFS filesystem, remove the yelp_dataset.tar file:
    rm yelp_dataset.tar
14. Download dictionary.tsv file by using wget utility:
    wget https://s3.amazonaws.com/hipicdatasets/dictionary.tsv
15. Upload dictionary.tsv and state_locations.txt to HDFS filesystem:
    hdfs dfs -put dictionary.tsv yelp/dictionary
    hdfs dfs -put state_locations.txt yelp/states
16. Verify the files are in the right places:
    hdfs dfs -ls -h yelp/dictionary
    hdfs dfs -ls -h yelp/states
17. Run this command to see if all the files are in right place:
    hdfs dfs -ls -R -h yelp/

    The output should look like:

```
-bash-4.1$ hdfs dfs -ls -R -h yelp/
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:20 yelp/business
-rw-r--rw-   2 malam hdfs    145.8 M 2020-11-11 23:20 yelp/business/yelp_academic_dataset_business.json
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:20 yelp/checkin
-rw-r--rw-   2 malam hdfs    428.8 M 2020-11-11 23:20 yelp/checkin/yelp_academic_dataset_checkin.json
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:22 yelp/dictionary
-rw-r--rw-   2 malam hdfs    301.7 K 2020-11-11 23:22 yelp/dictionary/dictionary.tsv
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:21 yelp/review
-rw-r--rw-   2 malam hdfs      5.9 G 2020-11-11 23:21 yelp/review/yelp_academic_dataset_review.json
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:24 yelp/states
-rw-r--rw-   2 malam hdfs        761 2020-11-11 23:24 yelp/states/state_locations.txt
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:21 yelp/tip
-rw-r--rw-   2 malam hdfs    251.3 M 2020-11-11 23:21 yelp/tip/yelp_academic_dataset_tip.json
drwxr-xrwx   - malam hdfs          0 2020-11-11 23:22 yelp/user
-rw-r--rw-   2 malam hdfs      3.0 G 2020-11-11 23:22 yelp/user/yelp_academic_dataset_user.json
```

18. Create a result sub-directory in HDFS to hold all results within this directory:
    hdfs dfs -mkdir yelp/results
19. To allow Hive to work, we need to change permission:
    hdfs dfs -chmod -R o+w .

Step 2: Creating Primary Tables

In this step, we create the initial tables from the dataset's CSV files. The CSV files have data in JSON format. We use Hive's JSON parser to create these tables.

1. Enter beeline and connect to the beeline server

   beeline

   !connect jdbc:hive2://summer2020-bdcsce-1:2181,summer2020-bdcsce-2:2181,summer2020-bdcsce-3:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2?tez.queue.name=interactive bdcsce_admin

   If it is successful, the CLI should change to something like this:

```
0: jdbc:hive2://summer2020-bdcsce-1:2181>
```

2. Access your database and see existing tables

   use uname;
   show tables;

3. Now, run these block of codes one by one to create raw tables based on JSON files, and then a standard table with proper column names (Hive comments start with --, so the lines that begin with -- can be omitted)

   --Creating table raw_business FROM the yelp_academic_dataset_business.json file. This json file is saved in the /user/UNAME/yelp/business directory of HDFS file system
   CREATE EXTERNAL TABLE raw_business (json_response string) STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/business';

```
No rows affected (0.238 seconds)
```

   --Creating business table
   CREATE TABLE business (business_id string, bus_name string, bus_address string, bus_city string, bus_state string, bus_postal_code string, bus_latitude float, bus_longitude float, bus_stars float, bus_review_count int, bus_is_open tinyint, bus_attributes string, bus_categories string, bus_hours string);

```
No rows affected (0.201 seconds)
```

   --Populating business table FROM raw_business
   FROM raw_business INSERT OVERWRITE TABLE business SELECT get_json_object(json_response, '$.business_id'), get_json_object(json_response, '$.name'), get_json_object(json_response, '$.address'), get_json_object(json_response, '$.city'), get_json_object(json_response, '$.state'), get_json_object(json_response, '$.postal_code'), get_json_object(json_response, '$.latitude'), get_json_object(json_response, '$.longitude'),get_json_object(json_response, '$.stars'), get_json_object(json_response, '$.review_count'), get_json_object(json_response, '$.is_open'), cast(get_json_object(json_response, '$.attributes') as string),get_json_object(json_response, '$.categories'), get_json_object(json_response, '$.hours');

```
No rows affected (29.006 seconds)
```

--Creating state_locations table for efficient map rendering
CREATE EXTERNAL TABLE state_locations (bus_state string, state_names string, country_names string) row format delimited fields terminated by '\t' STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/states/';

```
No rows affected (0.315 seconds)
```

--Creating table raw_checkin
CREATE EXTERNAL TABLE raw_checkin (json_response string) STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/checkin';

```
No rows affected (0.179 seconds)
```

--Creating checkin table
CREATE TABLE checkin (business_id string, checkin_dates string);

```
No rows affected (0.318 seconds)
```

--Populating checkin table based on the raw_checkin table.
FROM raw_checkin INSERT OVERWRITE TABLE checkin SELECT get_json_object(json_response, '$.business_id'), get_json_object(json_response, '$.date');

```
No rows affected (13.083 seconds)
```

--Creating table raw_review
CREATE EXTERNAL TABLE raw_review (json_response string) STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/review';

```
No rows affected (0.277 seconds)
```

--Creating review table
CREATE TABLE review (review_id string, rev_user_id string, rev_business_id string, rev_stars int, rev_useful int, rev_funny int, rev_cool int, rev_text string, rev_timestamp string, rev_date date);

```
No rows affected (0.219 seconds)
```

--Populating review table FROM raw_review
FROM raw_review INSERT OVERWRITE TABLE review SELECT get_json_object(json_response, '$.review_id'), get_json_object(json_response, '$.user_id'), get_json_object(json_response, '$.business_id'), get_json_object(json_response, '$.stars'), get_json_object(json_response, '$.useful'), get_json_object(json_response, '$.funny'), get_json_object(json_response, '$.cool'), regexp_replace(regexp_replace(get_json_object(json_response,'$.text'), '\n', ' '), '\r', ' '), get_json_object(json_response, '$.date'), cast(substr(get_json_object(json_response, '$.date'),0,10) as date);

```
No rows affected (60.73 seconds)
```

--Creating table raw_tip
CREATE EXTERNAL TABLE raw_tip (json_response string) STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/tip';

```
No rows affected (0.192 seconds)
```

```
--Creating table 'tip'
CREATE TABLE tip (tip_user_id STRING, tip_business_id STRING, tip_text STRING,
tip_date date, tip_compliment_count int);
```

```
No rows affected (0.198 seconds)
```

```
--Populating tip table based on the raw_tip table
FROM raw_tip INSERT OVERWRITE TABLE tip SELECT
get_json_object(json_response,'$.user_id'),
get_json_object(json_response,'$.business_id'),
regexp_replace(get_json_object(json_response,'$.text'), '\n', ' '),
cast(substr(get_json_object(json_response,'$.date'),0,10) as date),
cast(get_json_object(json_response,'$.compliment_count') as int);
```

```
No rows affected (5.468 seconds)
```

```
--Creating a view tip_modified with an added column tip_id, which will act as a row
identifier/primary key;
CREATE VIEW tip_modified as SELECT row_number() over() tip_id, tip_user_id,
tip_business_id, tip_text, tip_date, tip_compliment_count FROM tip;
```

```
No rows affected (0.395 seconds)
```

```
--Creating table raw_user
CREATE EXTERNAL TABLE raw_user (json_response string) STORED AS TEXTFILE
LOCATION '/user/UNAME/yelp/user';
```

```
No rows affected (0.222 seconds)
```

```
--Creating table users
CREATE TABLE users (user_id string, user_name string, user_review_count int,
user_yelping_since string, user_friends string, user_useful int, user_funny int, user_cool
int, user_fans int, user_elite string, user_average_stars float, user_compliment_hot int,
user_compliment_more int, user_compliment_profile int, user_compliment_cute int,
user_compliment_list int, user_compliment_note int, user_compliment_plain int,
user_compliment_cool int, user_compliment_funny int, user_compliment_writer int,
user_compliment_photos int);
```

```
No rows affected (0.294 seconds)
```

```
--Populating users FROM raw_user
FROM raw_user INSERT OVERWRITE TABLE users SELECT
get_json_object(json_response, '$.user_id'), get_json_object(json_response, '$.name'),
get_json_object(json_response, '$.review_count'), get_json_object(json_response,
'$.yelping_since'), get_json_object(json_response, '$.friends'),
get_json_object(json_response, '$.useful'), get_json_object(json_response, '$.funny'),
get_json_object(json_response, '$.cool'), get_json_object(json_response, '$.fans'),
get_json_object(json_response, '$.elite'), get_json_object(json_response,
'$.average_stars'), get_json_object(json_response, '$.compliment_hot'),
get_json_object(json_response, '$.compliment_more'), get_json_object(json_response,
'$.compliment_profile'), get_json_object(json_response, '$.compliment_cute'),
get_json_object(json_response, '$.compliment_list'), get_json_object(json_response,
'$.compliment_note'), get_json_object(json_response, '$.compliment_plain'),
```

get_json_object(json_response, '$.compliment_cool'), get_json_object(json_response, '$.compliment_funny'), get_json_object(json_response, '$.compliment_writer'), get_json_object(json_response, '$.compliment_photos');

```
No rows affected (49.949 seconds)
```

4. Ensure that the tables are created

   show tables;

```
+-------------------------+--+
|         tab_name        |
+-------------------------+--+
| business                |
| checkin                 |
| raw_business            |
| raw_checkin             |
| raw_review              |
| raw_tip                 |
| raw_user                |
| review                  |
| state_locations         |
| tip                     |
| tip_modified            |
| users                   |
```

5. Using COUNT function to see if the parsed tables have the same numbers of entities mentioned by Yelp Documentation

   SELECT COUNT(business_id) FROM business;

```
+---------+--+
|   _c0   |
+---------+--+
| 209393  |
+---------+--+
1 row selected (15.996 seconds)
```

   SELECT COUNT(review_id) FROM review;

```
+----------+--+
|   _c0    |
+----------+--+
| 8021122  |
+----------+--+
1 row selected (14.061 seconds)
```

SELECT COUNT(tip_id) FROM tip_modified;

```
+----------+--+
|   _c0    |
+----------+--+
| 1320761  |
+----------+--+
1 row selected (17.307 seconds)
```

SELECT COUNT(user_id) FROM users;

```
+----------+--+
|   _c0    |
+----------+--+
| 1968703  |
+----------+--+
1 row selected (17.797 seconds)
```

6. Peek into the data using SELECT command

select * from business limit 2;

```
| f9NumwFMBDn751xgFiRbNA  | The Range At Lake Norman  | 10913 Bailey
Rd                | Cornelius             | NC                        |
28031                                 |  35.46272277832031        |  -
80.85261535644531     | 3.5                | 36                        |
1                                                                   |
{"BusinessAcceptsCreditCards":"True","BikeParking":"True","GoodForKids"
:"False","BusinessParking":"{'garage':    False,    'street':    False,
'validated':         False,         'lot':         True,         'valet':
False}","ByAppointmentOnly":"False","RestaurantsPriceRange2":"3"}      |
Active Life, Gun/Rifle Ranges, Guns & Ammo, Shopping                   |
{"Monday":"10:0-18:0","Tuesday":"11:0-20:0","Wednesday":"10:0-
18:0","Thursday":"11:0-20:0","Friday":"11:0-20:0","Saturday":"11:0-
20:0","Sunday":"13:0-18:0"}  |

| YzvjgOSayhoZgCljUJRF9Q  | Carlos Santo, NMD        | 8880 E Via Linda,
Ste  107   |  Scottsdale              |  AZ                           |
85258                                 |  33.56940460205078        |  -
111.89026641845703    | 5.0                | 4                         |
1                                                                   |
{"GoodForKids":"True","ByAppointmentOnly":"True"}


                              | Health & Medical, Fitness & Instruction,
Yoga,           Active           Life,           Pilates             |
NULL

              |

2 rows selected (0.1 seconds)
```

SELECT * FROM checkin LIMIT 1;

```
| --1UhMGODdWsrMastO9DZw  | 2016-04-26 19:49:16, 2016-08-30 18:36:57,
2016-10-15 02:45:18, 2016-11-18 01:54:50, 2017-04-20 18:39:06, 2017-05-
03 17:58:02, 2019-03-19 22:04:48  |


1 row selected (0.24 seconds)
```

SELECT * FROM review LIMIT 2;
SELECT * FROM tip_modified LIMIT 2;
SELECT * FROM users LIMIT 1;
SELECT * FROM state_locations LIMIT 5;

Step 4: Performance Analysis

We analyze how different Yelp features performed over time.

1. Use SPLIT function to split the checkin_dates strings to arrays of timestamps and EXPLODE function to create new rows for each timestamp. Then, save this data into a view.

   CREATE VIEW checkin_clean as SELECT business_id, CAST(SUBSTR(timestamps, 0, 10) as date) checkin_dates FROM checkin lateral view explode(split(checkin_dates, ', ')) dummy as timestamps;

```
No rows affected (0.377 seconds)
```

2. Create checkin_per_year table to count all check-ins per year

   CREATE TABLE checkin_per_year as SELECT checkin_year, count(business_id) checkin_count FROM (SELECT year(checkin_dates) checkin_year, business_id FROM checkin_clean) checkin_temp GROUP BY checkin_year ORDER BY checkin_year;

```
No rows affected (34.304 seconds)
```

3. Create review_per_year table

   CREATE TABLE review_per_year as SELECT rev_year, count(review_id) review_count FROM (SELECT year(rev_date) rev_year, review_id FROM review) review_temp GROUP BY rev_year ORDER BY rev_year;

```
No rows affected (33.461 seconds)
```

4. Creating tip_per_year table

   CREATE TABLE tip_per_year as SELECT tip_year, count(tip_id) tip_count FROM (SELECT year(tip_date) tip_year, tip_id FROM tip_modified) tip_summary GROUP BY tip_year ORDER BY tip_year;

```
No rows affected (26.076 seconds)
```

5. Create users_summary table to minimize the users table

   CREATE TABLE users_summary as SELECT user_id, size(split(user_friends, ',')) user_friends_count, CAST(ROUND(length(user_elite)/6) as int) user_elite_count, user_review_count, user_fans, ROUND(user_average_stars, 2) user_average_stars, CAST(SUBSTR(user_yelping_since, 0, 10) as date) user_yelping_since, (user_compliment_hot + user_compliment_more + user_compliment_profile + user_compliment_cute + user_compliment_list + user_compliment_note + user_compliment_plain + user_compliment_cool + user_compliment_funny +

user_compliment_writer + user_compliment_photos) user_compliment_total FROM users;

```
No rows affected (40.281 seconds)
```

6. Create user_new_per_year table to count newly added users per year

   CREATE TABLE user_new_per_year as SELECT user_year, count(user_id) new_users_count FROM (SELECT year(user_yelping_since) user_year, user_id FROM users_summary) users_temp GROUP BY user_year ORDER BY user_year;

```
No rows affected (12.451 seconds)
```

7. Create user_elite view to allow further sorting

   CREATE VIEW users_elite as SELECT user_id, user_elite_year FROM users lateral view explode(split(user_elite, ',')) dummy as user_elite_year;

```
No rows affected (0.209 seconds)
```

8. Create user_elite_per_year to count number of elite users per year

   CREATE TABLE user_elite_per_year as SELECT user_elite_year, count(user_id) elite_users_count FROM users_elite GROUP BY user_elite_year ORDER BY user_elite_year;

```
No rows affected (28.59 seconds)
```

9. Join all per_year tables to generate combined report

   CREATE TABLE yelp_per_year ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/results/yelp_per_year' as SELECT user_year years, new_users_count, review_count, elite_users_count, tip_count, checkin_count FROM user_new_per_year full outer join review_per_year on user_year = rev_year full outer join user_elite_per_year on user_year = user_elite_year full outer join tip_per_year on user_year = tip_year full outer join checkin_per_year on user_year = checkin_year where user_year is not null ORDER BY years;

```
No rows affected (12.43 seconds)
```

10. View the result

    SELECT * FROM yelp_per_year;

```
| 2004                    | 82                                  | 12
| NULL                              | NULL                      | NULL
|

| 2005                    | 1022                                | 875
| NULL                              | NULL                      | NULL
|

| 2006                    | 6052                                | 5030
| 896                               | NULL                      | NULL
|

| 2007                    | 17155                               | 21130
| 2368                              | NULL                      | NULL
|

| 2008                    | 34327                               | 56996
| 3592                              | NULL                      | NULL
|
```

```
| 2009                    | 68314                                | 100760
| 6369                              | 957                        | NULL
|

| 2010                    | 115106                               | 186752
| 10238                             | 41922                      | 393953
|

| 2011                    | 185076                               | 302523
| 12809                             | 146532                     | 1608736
|

| 2012                    | 203180                               | 367367
| 17362                             | 185961                     | 2233001
|

| 2013                    | 221380                               | 491678
| 18223                             | 167643                     | 2665596
|

| 2014                    | 250827                               | 702060
| 20508                             | 163943                     | 2742368
|

| 2015                    | 267267                               | 940603
| 26409                             | 130844                     | 2766769
|

| 2016                    | 241414                               | 1094154
| 32128                             | 145569                     | 2560414
|

| 2017                    | 158881                               | 1217292
| 38645                             | 151006                     | 2307315
|

| 2018                    | 122892                               | 1318054
| 43026                             | 107826                     | 2008051
|

| 2019                    | 75728                                | 1215836
| NULL                              | 78558                      | 1717574
|

+----------------------+--------------------------------+--------------
---------------+----------------------------------+--------------------
------+------------------------------+--+

16 rows selected (0.224 seconds)
```

11. Open another terminal to run these shell commands to copy the output file from previous step to Linux filesystem

    hdfs dfs -get yelp/results/yelp_per_year/0*
    cat 00* > yelp_per_year.csv
    rm 00*

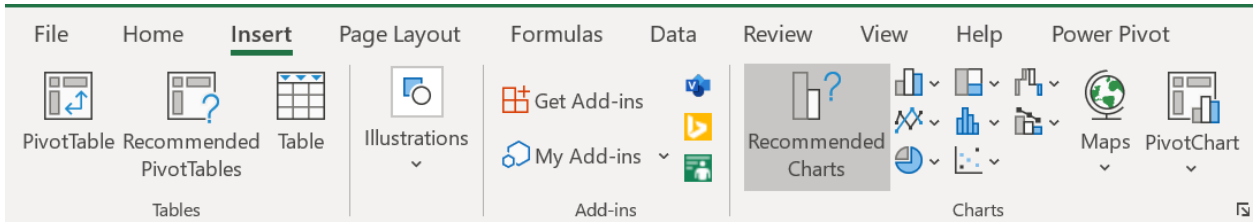
12. Then, use SCP utility in another terminal to download the file to your local machine

    scp UNAME@129.150.79.19:/home/UNAME/yelp_per_year.csv Downloads/yelp_per_year.csv

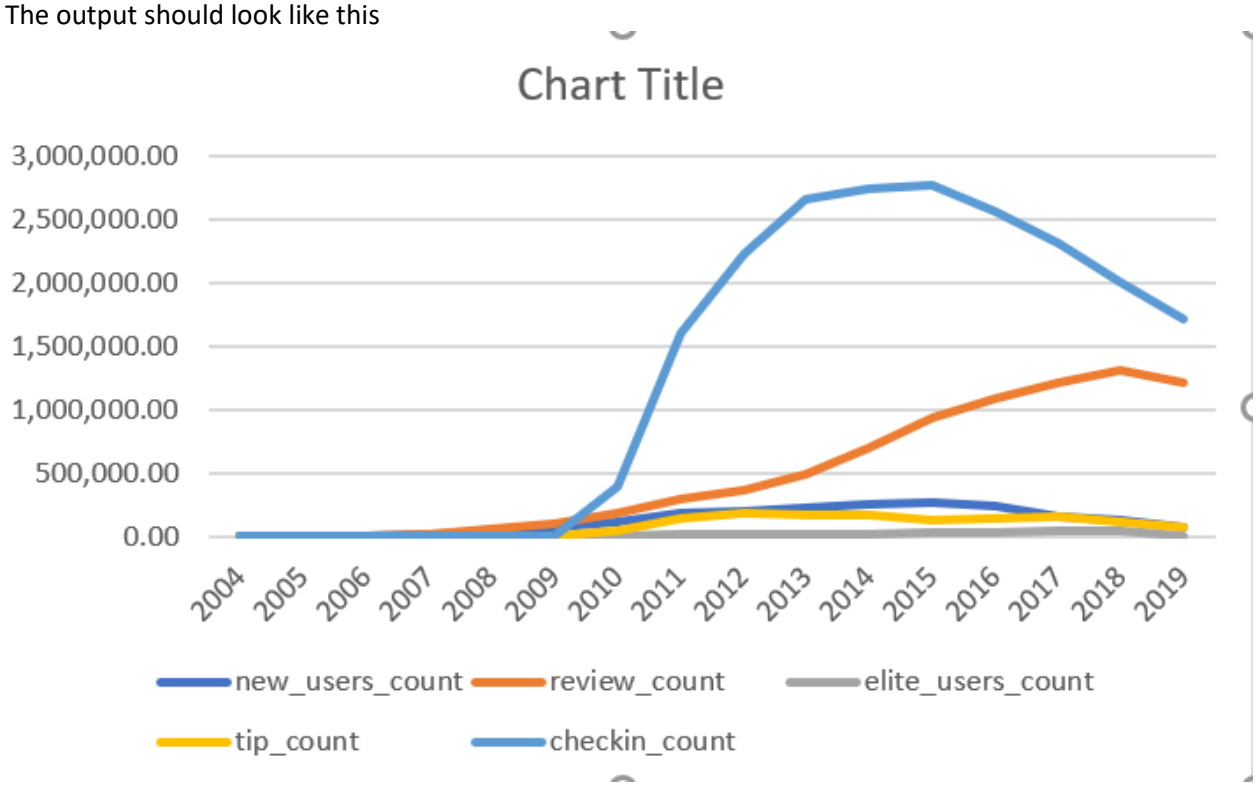
13. Open the downloaded file with Excel and insert a row at the beginning and add the column headers as follows

    years, new_users_count, review_count, elite_users_count, tip_count, checkin_count

14. Select all data, then click on Insert, then Recommended Charts, and select the first recommended chart



15. The output should look like this



Step 5: Tip Sentiment Analysis

Tips are short feedback that can describe a business's unique attribute or service quality or a reviewer's feelings about a particular business. In this part of the lab, we will conduct a sentiment analysis on tips and visualize the findings using geo-temporal visualization tools.

1. Create a dictionary table based on the dictionary.tsv file

   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/UNAME/yelp/dictionary';

```
No rows affected (0.286 seconds)
```

2. Create a view called L1_tip to break down tip sentences into new rows

   CREATE VIEW IF NOT EXISTS L1_tip as select tip_id, words from tip_modified lateral view explode(sentences(lower(tip_text))) dummy as words;

```
No rows affected (0.308 seconds)
```

Run a select command to peek into L1_tip:

select * from L1_tip limit 5;

```
| l1_tip.tip_id  |                                                    l1_tip.words
|

+----------------+-------------------------------------------------------
---------------------------------------------+--+

| 1|["such","a","fun","night"]|

|1 |["jarrod","was","awesome"]|

|1 | ["10","out","of","10","would","recommend"]|

|2                                                                      |
["great","healthy","food","and","great","service","must","stop","by","i
f","you're","in","town"]  |

|3 | ["oh","yea"]|

+----------------+-------------------------------------------------------
---------------------------------------------+--+

5 rows selected (216.116 seconds)
```

3. Create a new view to split each word into new row

   CREATE VIEW IF NOT EXISTS L2_tip as select tip_id, word from l1_tip lateral view explode(words) dummy as word;

```
No rows affected (0.198 seconds)
```

   SELECT * from L2_tip limit 5;

```
+----------------+--------------+--+

| l2_tip.tip_id  | l2_tip.word  |

+----------------+--------------+--+

| 1              | its          |

| 1              | not          |

| 1              | m            |

| 1              | but          |

| 1              | it'll        |

+----------------+--------------+--+

5 rows selected (32.117 seconds)
```

4. Join the L2_tip view with Dictionary table to classify each word

   CREATE VIEW IF NOT EXISTS l3_tip as select tip_id, l2_tip.word, case d.polarity when 'negative' then -1 when 'positive' then 1 else 0 end as polarity from l2_tip left outer join dictionary d on l2_tip.word = d.word;

```
No rows affected (0.295 seconds)
```

   SELECT * from L3_tip limit 5;

```
+----------------+--------------+------------------+--+
| l3_tip.tip_id  | l3_tip.word  | l3_tip.polarity  |
+----------------+--------------+------------------+--+
| 1              | its          | 0                |
| 1              | not          | 0                |
| 1              | m            | 0                |
| 1              | but          | 0                |
| 1              | it'll        | 0                |
+----------------+--------------+------------------+--+
```

5. Create tip_sentiment table to aggregate sentiments of all words for individual tip

   CREATE TABLE tip_sentiment as SELECT tip_id, case when sum(polarity) > 0 then 'positive' when sum(polarity) < 0 then 'negative' else 'neutral' end as tip_sentiment from l3_tip GROUP BY tip_id ORDER BY tip_id;

```
No rows affected (65.608 seconds)
```

   SELECT * from tip_sentiment limit 5;

```
+-----------------------+-----------------------------+--+
| tip_sentiment.tip_id  | tip_sentiment.tip_sentiment  |
+-----------------------+-----------------------------+--+
| 1                     | neutral                     |
| 2                     | positive                    |
| 3                     | neutral                     |
| 4                     | positive                    |
| 5                     | neutral                     |
+-----------------------+-----------------------------+--+
5 rows selected (0.234 seconds)
```

6. Join tip_sentiment, tip_modifed, business, and state_locations tables to create a table with aggregated location, time, and sentiment information

   CREATE TABLE tip_sentiment_summary ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/results/tip_sentiment_summary' as SELECT country_names, state_names, tip_date, tip_sentiment, count(ts.tip_id) FROM tip_modified tm JOIN tip_sentiment ts ON tm.tip_id=ts.tip_id JOIN business ON tm.tip_business_id= business.business_id JOIN state_locations sl ON business.bus_state = sl.bus_state GROUP BY country_names, state_names, tip_date, tip_sentiment ORDER BY country_names, state_names, tip_date;

```
No rows affected (49.423 seconds)
```

   SELECT * FROM tip_sentiment_summary LIMIT 3;

```
 tip_sentiment_summary.country_names                                    |
tip_sentiment_summary.state_names   | tip_sentiment_summary.tip_date   |
tip_sentiment_summary.tip_sentiment  | tip_sentiment_summary._c4  |

|  Canada                              | Alberta                          |
2009-06-04                        | neutral                          |
1                            |

|  Canada                              | Alberta                          |
2009-08-05                        | positive                         |
1                            |

|  Canada                              | Alberta                          |
2009-08-30                        | positive                         |
1                            |

3 rows selected (0.311 seconds)
```

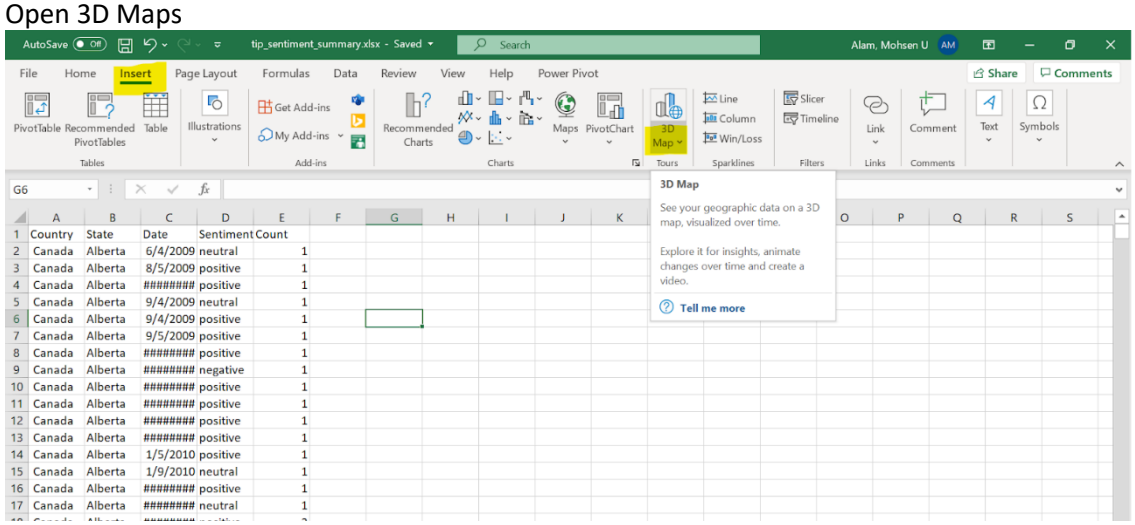
7. Open another terminal; after connecting to the Oracle server, run these commands to copy the output file

   #Delete previously copied Hive output files
   rm 000*
   #Copy output file from HDFS filesystem
   hdfs dfs -get yelp/results/tip_sentiment_summary/0*
   #Convert to a .csv file
   cat 000000_0 > tip_sentiment_summary.csv
   #exit to use SCP utility
   exit
   #Copy the rating_category_count.csv file to local workstation
   scp UNAME@129.150.64.74:/home/UNAME/tip_sentiment_summary.csv Downloads/

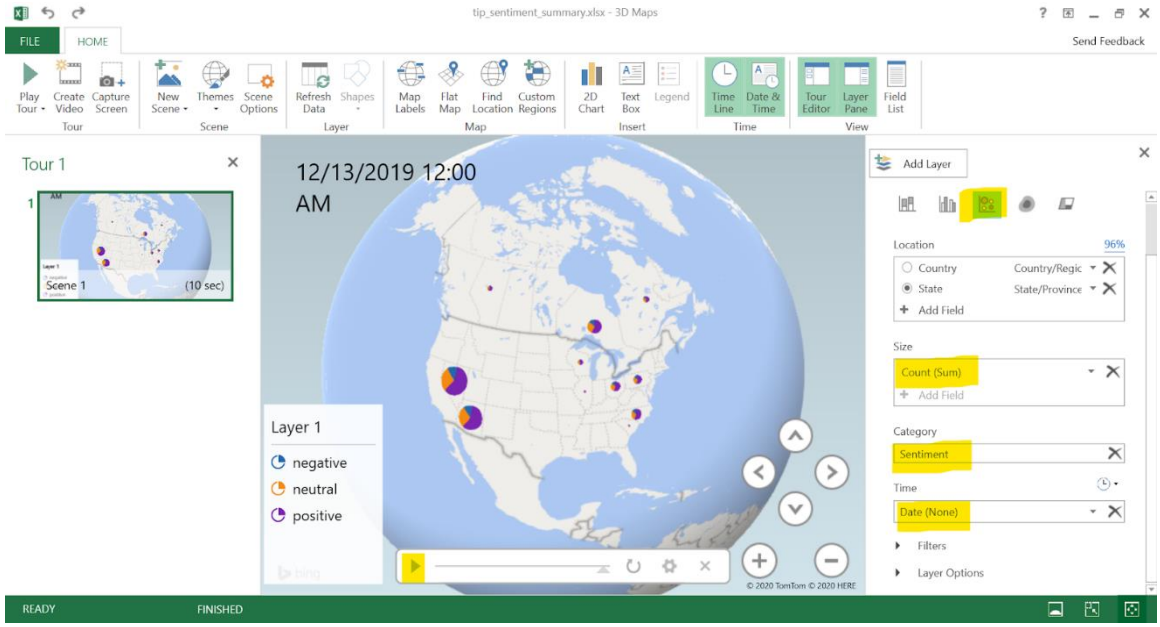
8. Open the downloaded CSV file with Excel. Add a row at the top and put these as column header:

   Country State Date Sentiment Count

9. Save the file as an Excel workbook (.xlsx), then select all data, then click on Insert > 3D Map > Open 3D Maps



10. In the 3D Map window, choose Bubble visualization and then choose Count as Size, Sentiment as Category, and Date as Time



11. Hit the Play button to see the change in count over the time.

Step 6: Star-Rating Analysis

We analyze the distribution of star-ratings in different states in this step.

1. Join the business and review table to gather some necessary data

   CREATE VIEW review_location as SELECT b.business_id, bus_latitude, bus_longitude, bus_city, bus_state, rev_stars, rev_text, rev_date FROM business b JOIN review r ON b.business_id = r.rev_business_id;

```
No rows affected (0.28 seconds)
```

2. Extracting month and year FROM review_date and create a pseudo-date for each month in each year

   CREATE VIEW review_location_yyyymm as SELECT business_id, bus_latitude, bus_longitude, bus_city, bus_state, rev_stars, cast(concat(year(rev_date), '-', month(rev_date), '-', 1) as date) rev_yyyy_mm FROM review_location;

```
No rows affected (0.232 seconds)
```

3. Count rating stars per state

   CREATE TABLE rating_category_count ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/UNAME/yelp/results/rating_category' as SELECT s.country_names, s.state_names, r.rev_yyyy_mm, r.rev_stars, count(business_id) FROM review_location_yyyymm r JOIN state_locations s where r.bus_state = s.bus_state group by s.country_names, s.state_names, r.rev_yyyy_mm, r.rev_stars order by s.country_names, s.state_names, r.rev_yyyy_mm, r.rev_stars;

```
No rows affected (56.343 seconds)
```

4. Peek into the created table

   SELECT * FROM rating_category_count LIMIT 5;

```
|            rating_category_count.country_names                          |
rating_category_count.state_names  | rating_category_count.rev_yyyy_mm  |
rating_category_count.rev_stars   | rating_category_count._c4   |


|  Canada                              | Alberta                           |
2008-08-01                            | 1                                 |
1                           |

|  Canada                              | Alberta                           |
2008-08-01                            | 3                                 |
5                           |

|  Canada                              | Alberta                           |
2008-08-01                            | 4                                 |
24                          |

|  Canada                              | Alberta                           |
2008-08-01                            | 5                                 |
21                          |

|  Canada                              | Alberta                           |
2008-09-01                            | 1                                 |
3                           |

+-------------------------------------+------------------------------
-----+-------------------------------------+----------------------------
------+----------------------------+--+

5 rows selected (0.144 seconds)
```

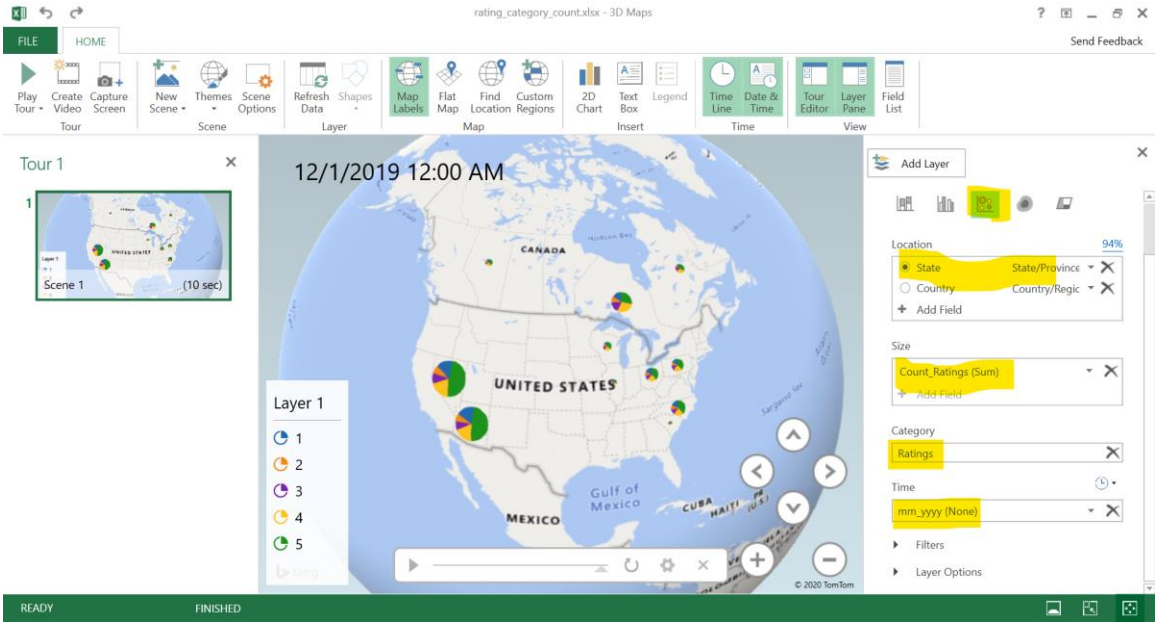
5. Open another terminal; after connecting to the Oracle server, run these commands to copy the output file

   #Delete previously copied Hive output files
   rm 000*
   #Copy output file from HDFS filesystem
   hdfs dfs -get yelp/results/rating_category/0*
   #Convert to a .csv file
   cat 000000_0 > rating_category_count.csv
   #exit to use SCP utility
   Exit
   #Copy the rating_category_count.csv file to local workstation
   scp UNAME@129.150.64.74:/home/UNAME/ rating_category_count.csv.csv Downloads/

6. Open the downloaded CSV file with Excel. Add a row at the top and put these as column header:

   Country State mm_yyyy Ratings Count_Ratings

7. Save the file as an Excel workbook (.xlsx), then select all data, then click on Insert > 3D Map > Open 3D Maps
8. In the 3D Map window, choose Bubble visualization and then choose Count_Ratings(sum) as Size, Ratings as Category, and mm_yyyy as Time



9. Hit the Play button to see the change in count over the time.

References

1. Yelp Dataset: https://www.yelp.com/dataset/download

2. Dictionary.tsv: https://s3.amazonaws.com/hipicdatasets/dictionary.tsv

3. State_locations.txt:

   https://drive.google.com/uc?id=1dFrIcQuBhaANRHHvnzbthfU3HHVDRy7Y&export=download

4. GitHub: https://github.com/CIS-4560-Team-2/Hive-on-Yelp

5. Griffo, U. (2016). Step by step Tutorial on Twitter Sentiment Analysis and n-gram with Hadoop and Hive SQL. https://gist.github.com/umbertogriffo/a512baaf63ce0797e175