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

Objective

We will use Hive to explore the Yelp Dataset to find out how different features of Yelp have evolved over the years and how reviewers from different regions rank and feel about local businesses.

Introduction

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

Objective of this lab is to 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

This lab has four sections:

1. Evaluation of Yelp features over the time
2. Sentiment analysis of tips with geo-spatial visualization
3. Rating distributions in different regions

4. Exploring Child and Pet related categories

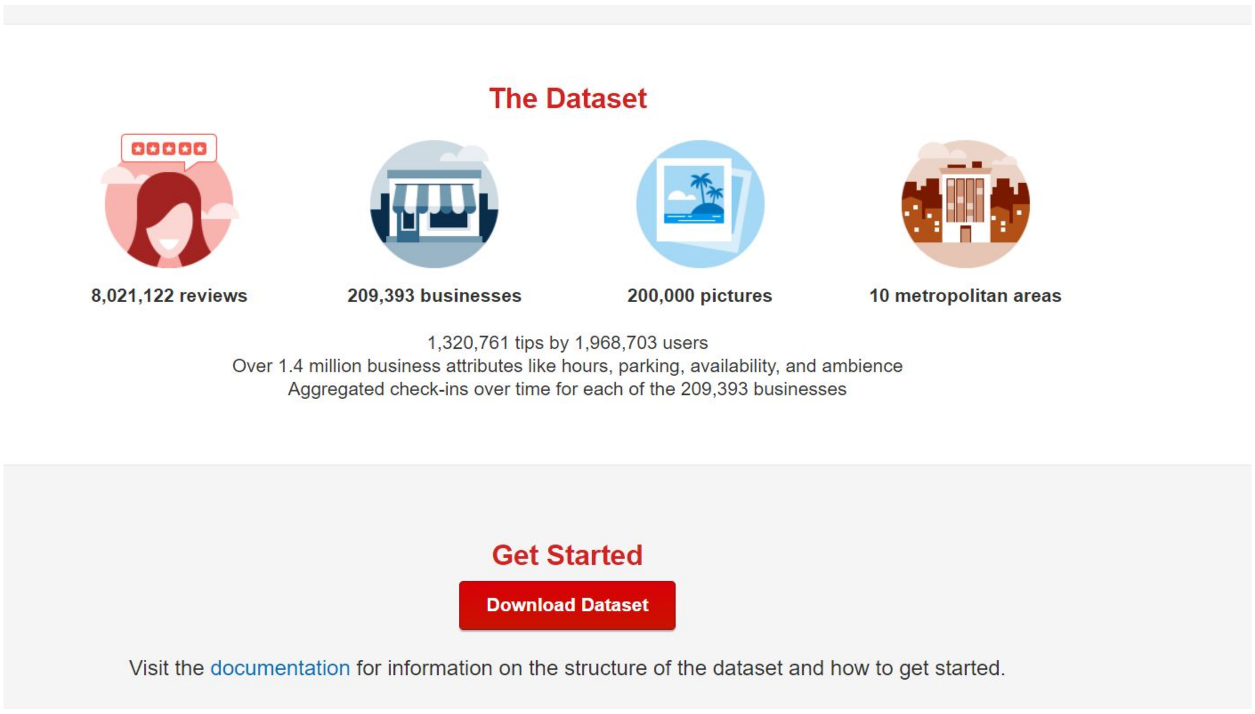
Prerequisites

- Access to a latest version of Excel such as Excel 2016 or Excel 2019
- 3D Map enabled in Excel
- Any terminal utility such as Git Bash or Putty

Lab Setup: Data Loading

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.

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


The Dataset

8,021,122 reviews

209,393 businesses

200,000 pictures

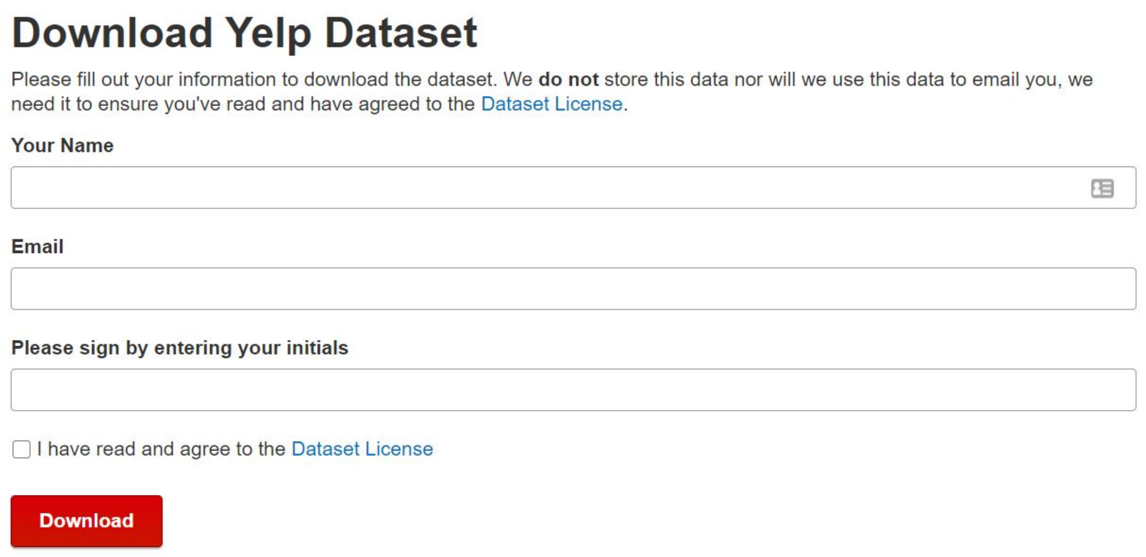
10 metropolitan areas

1,320,761 tips by 1,968,703 users
Over 1.4 million business attributes like hours, parking, availability, and ambience
Aggregated check-ins over time for each of the 209,393 businesses

Get Started

Visit the documentation for information on the structure of the dataset and how to get started.

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



Download Yelp Dataset

Please fill out your information to download the dataset. We **do not** store this data nor will we use this data to email you, we need it to ensure you've read and have agreed to the Dataset License.

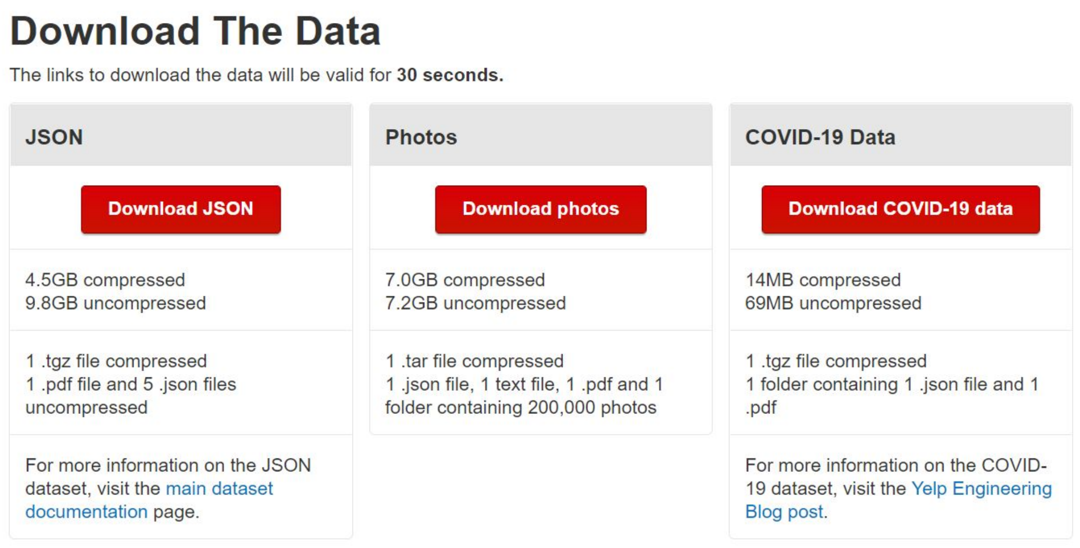
Your Name

Email

Please sign by entering your initials

☐ I have read and agree to the Dataset License

8. Click on the "Download JSON" button quickly since the link may be disabled 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.

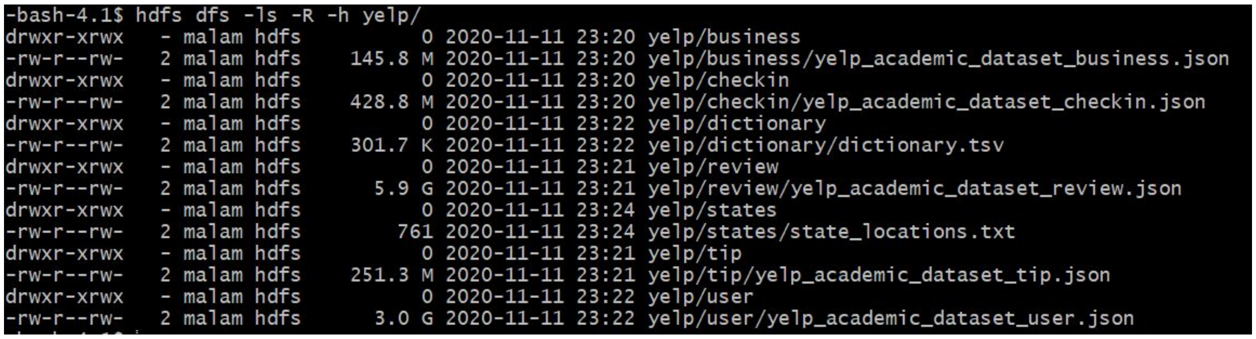
9. Use the SCP command in Git Bash or other terminal to upload the downloaded file to BDCE server as below (IP address used here is 129.150.79.19, which may change. Use proper IP address):

```
scp Downloads/yelp_dataset.tar UNAME@129.150.79.19:/home/UNAME/
```

10. Use web browser to browse this link and download the state_locations.txt file in your Downloads folder:
```
https://drive.google.com/uc?id=1dFrIcQuBhaANRHHvnzbthfU3HHVDRy7
Y&export=download
```
11. 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/
```
12. Open terminal and connect to the cloud BDCE server with appropriate password:
```
ssh UNAME@129.150.79.19
```
13. 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$
```
14. Run this command to verify that you have the yelp_dataset.tar file: `ls -hl`
15. 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
```
16. 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
```

17. 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
```
18. If the files are successfully uploaded to HDFS filesystem, remove the yelp_dataset.tar file:
```
rm yelp_dataset.tar
```
19. Download dictionary.tsv file by using wget utility:
```
wget https://s3.amazonaws.com/hipicdatasets/dictionary.tsv
```
20. 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
```
21. Verify the files are in the right places:
```
hdfs dfs -ls -h yelp/dictionary
hdfs dfs -ls -h yelp/states
```
22. Run this commands to see if all the files are in right place:
```
hdfs dfs -ls -R -h yelp/
```
This should show output like this:



```
-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
```

23. Run this command to hold all results within this directory:
```
hdfs dfs -mkdir yelp/results
```
24. To allow Hive to work, we need to change permission:
```
hdfs dfs -chmod -R o+w .
```

Lab Setup: Creating Primary Tables

1. Now we can start creating tables in the beeline environment. Run this command

```
beeline
```

2. Now enter this command to connect to a Hive server (this command may change; check with instructor to verify the link)
   ```
   !connect
   jdbc:hive2://summer2020-bdcsce-1:2181,summer2020-bdcsce-2:2181,
   summer2020-bdcsce-3:2181/;serviceDiscoveryMode=zooKeeper;zooKee
   perNamespace=hiveserver2?tez.queue.name=interactive
   bdcsce_admin
   ```
3. If it is successful, the CLI should have something like this:
   ```
   0: jdbc:hive2://summer2020-bdcsce-1:2181>
   ```
4. To use your database, run this:
   ```
   use UNAME;
   ```
5. Run this command to see existing tables, if any:
   ```
   show tables;
   ```
6. 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';

   --Output: 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);

   --Output: 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');

--Output: 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/';

--Output: 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';

--Output: No rows affected (0.179 seconds)

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

--Output: 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');

--Output: 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';

--Output: 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);

--Output: 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);

--Output: 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';

--Output: 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);

--Output: 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);

--Output: 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;

--Output: 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';

--Output: 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);

--Output: 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');

--Output: No rows affected (49.949 seconds)
```

7. Run this command again to see if all the tables are created
   `show tables;`
   This command should show at least these tables in you database:

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

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

```
SELECT COUNT(business_id) FROM business;

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


SELECT COUNT(review_id) FROM review;

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

SELECT COUNT(tip_id) FROM tip_modified;
```

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

```
SELECT COUNT(user_id) FROM users;
```

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

9. Run SELECT command to peek into data:

```
SELECT * FROM business LIMIT 2;
```

--Output:

```
| 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","Busin
essParking":"{'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":"11:0-2
0:0","Friday":"11:0-20:0","Saturday":"11:0-20:0","Sunday":"13:0-18:0"}  |
| Yzvjg0SayhoZgCljUJRF9Q  | 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;
```

```
--Output:
| --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;
```

```
--Output:

| xQY8N_XvtGbearJ5X4QryQ  | OwjRMXRC0KyPrIlcjaXeFQ  | -MhfebM0QIsKt87iDN-FNw  | 2
| 5                     | 0                     | 0                     | As someone who has
worked with many museums, I was eager to visit this gallery on my most recent trip to
Las Vegas. When I saw they would be showing infamous eggs of the House of Faberge from
the Virginia Museum of Fine Arts (VMFA), I knew I had to go!  Tucked away near the
gelateria and the garden, the Gallery is pretty much hidden from view. It's what real
estate agents would call "cozy" or "charming" - basically any euphemism for small.
That being said, you can still see wonderful art at a gallery of any size, so why the
two *s you ask? Let me tell you:  * pricing for this, while relatively inexpensive for
a Las Vegas attraction, is completely over the top. For the space and the amount of
art you can fit in there, it is a bit much. * it's not kid friendly at all. Seriously,
don't bring them. * the security is not trained properly for the show. When the
curating and design teams collaborate for exhibitions, there is a definite flow. That
means visitors should view the art in a certain sequence, whether it be by historical
period or cultural significance (this is how audio guides are usually developed). When
I arrived in the gallery I could not tell where to start, and security was certainly
not helpful. I was told to "just look around" and "do whatever."   At such a *fine*
institution, I find the lack of knowledge and respect for the art appalling.
| 2015-04-15 05:21:16   | 2015-04-15       |
| UmFMZ8PyXZTY2QcwzsfQYA  | nIJD_7ZXHq-FX8byPMOkMQ  | lbrU8StCq3yDfr-QMnGrmQ  | 1
| 1                     | 1                     | 0                     | I am actually horrified
this place is still in business. My 3 year old son needed a haircut this past summer
and the lure of the $7 kids cut signs got me in the door. We had to wait a few minutes
as both stylists were working on people. The decor in this place is total garbage. It
is so tacky. The sofa they had at the time was a pleather sofa with giant holes in it.
And my son noticed ants crawling all over the floor and the furniture. It was
disgusting and I should have walked out then. Actually, I should have turned around
and walked out upon entering but I didn't. So the older black male stylist finishes
the haircut he was doing and it's our turn. I tell him I want a #2 clipper around the
back and sides and then hand cut the top into a standard boys cut. Really freaking
simple, right? WRONG! Rather than use the clippers and go up to actually cut the hair,
he went down. Using it moving downward doesn't cut hair, it just rubs against it. How
does this man who has an alleged cosmetology license not know how to use a set of
freaking clippers??? I realized almost immediately that he had no idea what he was
doing. No idea at all. After about 10 minutes of watching this guy stumble through it,
I said "you know what? That's fine.", paid and left. All I wanted to do was get out of
that scummy joint and take my son to a real haircut place.  Bottom line: DO NOT GO
HERE. RUN THE OTHER WAY!!!!!  | 2013-12-07 03:16:52   | 2013-12-07       |
2 rows selected (0.076 seconds)
```

```
SELECT * FROM tip_modified LIMIT 2;
```

--Output:

```
      | 1                    | bXzM19nTkRQIQE2hFDr5fQ    | VMC8JY6jZ3uYlgnrgBKPEw
| Try to come early but if not it doesn't start on time so don't panic! It starts
about 20-30 minutes behind schedule  | 2016-06-15             | 0
|
      | 2                    | a7pT6vyAIXb5Zqt1dcwyYQ    | X3W-ddwbnZ2uAyYOxcYu3A
| Every day at sunset they have bagpipers playing - awesome :)
| 2012-06-05             | 0                                    |
      2 rows selected (23.494 seconds)
```

```
      SELECT * FROM users LIMIT 1;
```

```
      --Output:
      | ntlvfPzc8eglqvk92iDIAw  | Rafael           | 553                         |
2007-07-06 03:27:11       | oeMvJh94PiGQnx_6GlndPQ, wm1z1PaJKvHgSDRKfwhfDg,
IkRib6Xs91PPW7pon7VVig, A8Aq8f0-XvLBcyMk2GJdJQ, eEZM1kogR7eL4GOBZyPvBA,
e1o1LN7ez5ckCpQeAab4iw, _HrJVzFaRFUhPva8cwBjpQ, pZeGZGzX-ROT_D5lam5uNg,
0S6EI51ej5J7dgYz3-O0lA, woDt8raW-AorxQM_tIE2eA, hWUnSE5gKXNe7bDc8uAG9A,
c_3LDSO2RHwZ94_Q6j_O7w, -uv1wDiaplY6eXXS0VwQiA, QFjqxXn3acDC7hckFGUKMg,
ErOqapICmHPTN8YobZIcfQ, mJLRvqLOKhqEdkgt9iEaCQ, VKX7jlScJSA-ja5hYRw12Q,
ijIC9w5PRcj3dWVlanjZeg, CIZGlEw-Bp0rmkP8M6yQ9Q, OC6fT5WZ8EU7tEVJ3bzPBQ,
UZSDGTDpycDzrlfUlyw2dQ, deL6e_z9xqZTIODKqnvRXQ, 5mG2ENw2PylIWElqHSMGqg,
Uh5Kug2fvDd51RYmsNZkGg, 4dI4uoShugD9z84fYupelQ, EQpFHqGT9Tk6YSwORTtwpg,
o4EGL2-ICGmRJzJ3GxB-vw, s8gK7sdVzJcYKcPv2dkZXw, vOYVZgb_GVe-kdtjQwSUHw,
wBbjgHsrKr7BsPBrQwJf2w, p59u2EC_qcmCmLeX1jCi5Q, VSAZI1eHDrOPRWMK4Q2DIQ,
efMfeI_dkhpeGykaRJqxfQ, x6qYcQ8_i0mMDzSLsFCbZg, K_zSmtNGw1fu-vmxyTVfCQ,
5IM6YPQCK-NABkXmHhlRGQ, U_w8ZMD26vnkeeS1sD7s4Q, AbfS_oXF8H6HJb5jFqhrLw,
hbcjX4_D4KIfonNnwrH-cg, UKf66_MPz0zHCP70mF6p1g, hK2gYbxZRTqcqlSiQQcrtQ,
2Q45w_Twx_T9dXqlE16xtQ, BwRn8qcKSeA77HLaOTbfiQ, jouOn4VS_DtFPtMR2w8VDA,
ESteyJabbfvqas6CEDs3pQ  | 628                | 225                | 227                |
14               |                    | 3.569999933242798        | 3
| 2                          | 1                               | 0
| 1                          | 11                           | 15
| 22                         | 22                            | 10
| 0                            |

      1 row selected (0.095 seconds)
```

```
      SELECT * FROM state_locations LIMIT 5;
```

```
      --Output:
```

```
      | AB                          | Alberta                          |  Canada
|
      | AK                          | Alaska                           |  USA
|
      | AL                          | Alabama                          |  USA
|
      | AR                          | Arkansas                         |  USA
|
      | AZ                          | Arizona                          |  USA
|

      5 rows selected (0.058 seconds)
```

Part 1: Evaluation of Yelp features over the time

In this step, we will see how different Yelp features have evolved over time.

1. 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;
   ```
2. Create review_per_year table to count reviews per year
   ```
   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;
   ```
3. Create tip_per_year table to count tips per year
   ```
   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;
   ```
4. 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;
   ```
5. 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;
```

6. 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;
```

7. Join the tables crated above to generate a 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;
```

8. 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*
```

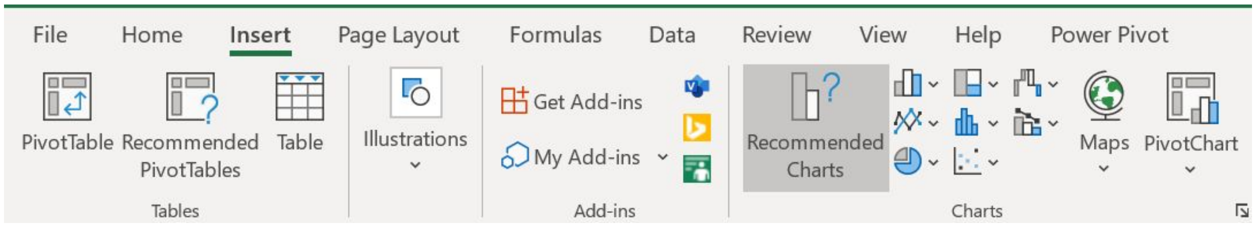
9. Then, use SCP utility in another terminal to download the file to your local machine

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

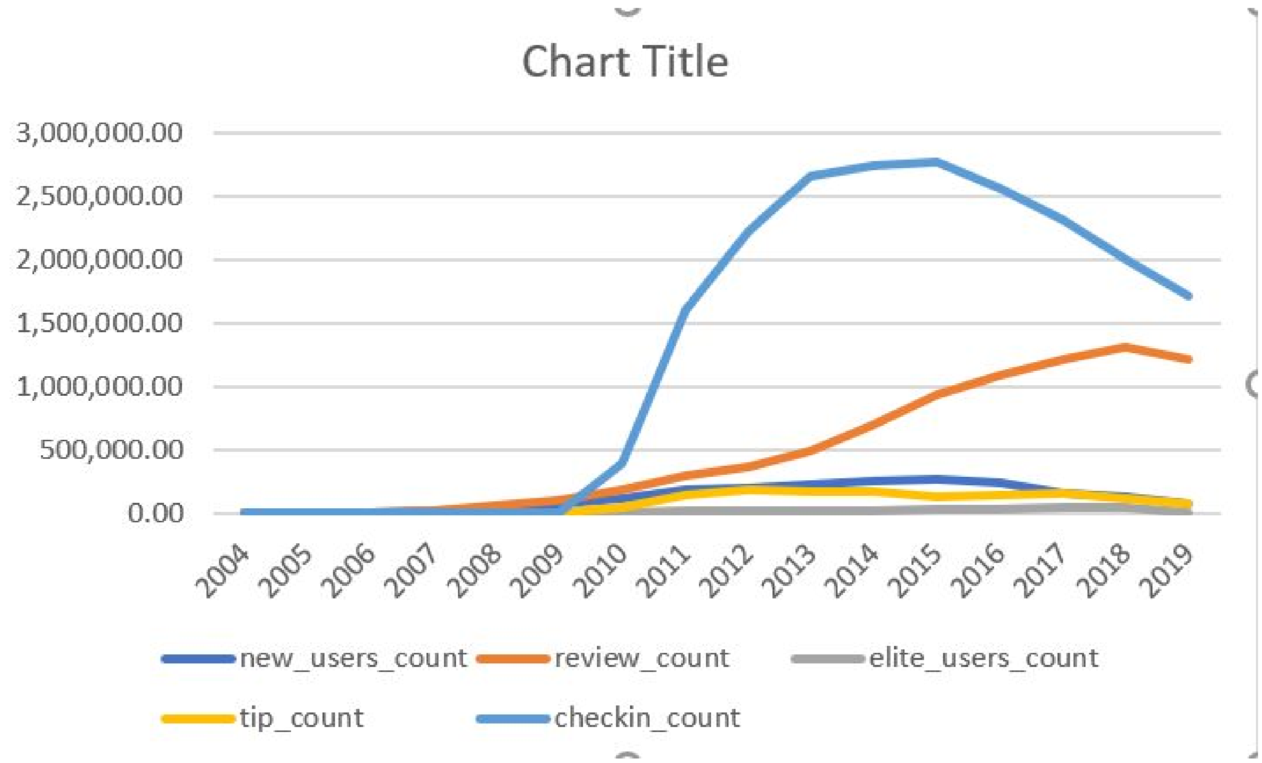
10. Open the downloaded file with Excel and insert a row at the beginning.
11. Add the column headers as follows:

```
years, new_users_count, review_count, elite_users_count,
tip_count, checkin_count
```

12. Select all data.
13. Click on Insert, then Recommended Charts, and select the first recommended chart



14. The output should look like this:



15. This chart shows that check-ins are dropping since 2014 unlike other entities.

Part 2: Sentiment Analysis of Tips with Geo-Spatial Visualization

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 snf visualize the findings using geo-temporal visualization tools.

1. Create a dictionary table

```
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';
```

```
--Output: 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;
```

```
--Output: No rows affected (0.308 seconds)
```

Run a select command to peek into L1_tip:

```
select * from L1_tip limit 5;
```

```
--Output:
+----------------+------------------------------------------------------------
------------------------------------+--+

| 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","if","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;

--Output: No rows affected (0.198 seconds)

SELECT * from L2_tip limit 5;

--Output: +----------------+--------------+--+

| 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;
```

```
--Output: No rows affected (0.295 seconds)
```

```
SELECT * from L3_tip limit 5;
```

```
--Output: +----------------+-------------+-----------------+--+

| 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 tips

```
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;
```

```
--Output: No rows affected (65.608 seconds)
```

```
SELECT * from tip_sentiment limit 5;
```

```
--Output: +-----------------------+------------------------------+--+

| 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;
```

```
--Output: No rows affected (49.423 seconds)
```

```
SELECT * FROM tip_sentiment_summary LIMIT 3;
```

```
--Output:  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)
```

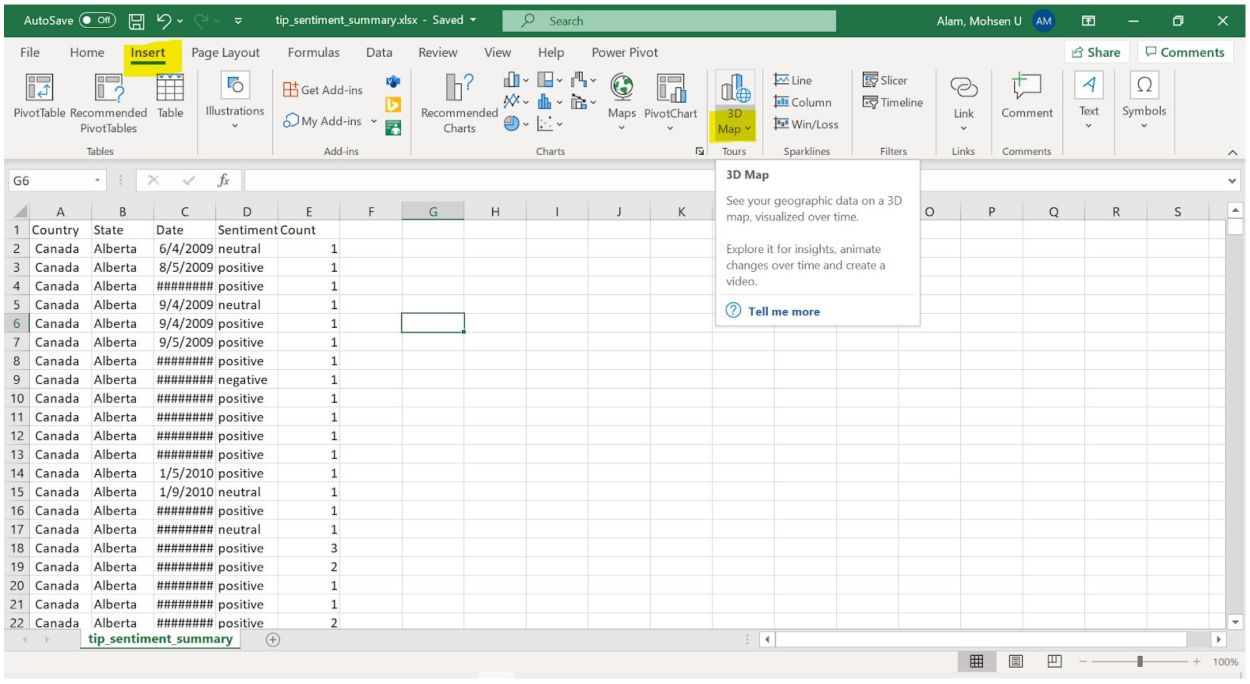
7. Open another terminal; after connecting to the Oracle server, run these commands to copy the output file

```
#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

scp malam@129.150.64.74:/home/malam/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)
10. Press Ctrl+a to select all data, then click on Insert > 3D Map > Open 3D Maps



3D Map

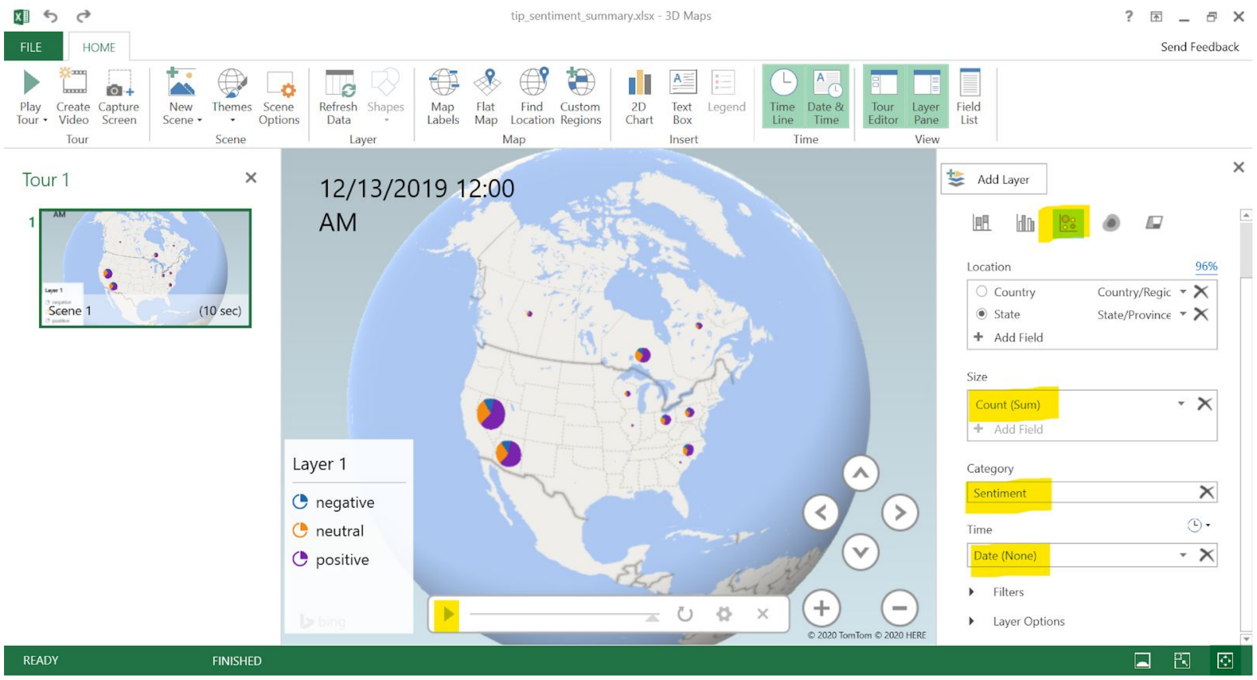
See your geographic data on a 3D map, visualized over time.

Explore it for insights, animate changes over time and create a video.

Tell me more

	A	B	C	D	E
1	Country	State	Date	Sentiment	Count
2	Canada	Alberta	6/4/2009	neutral	1
3	Canada	Alberta	8/5/2009	positive	1
4	Canada	Alberta	########	positive	1
5	Canada	Alberta	9/4/2009	neutral	1
6	Canada	Alberta	9/4/2009	positive	1
7	Canada	Alberta	9/5/2009	positive	1
8	Canada	Alberta	########	positive	1
9	Canada	Alberta	########	negative	1
10	Canada	Alberta	########	positive	1
11	Canada	Alberta	########	positive	1
12	Canada	Alberta	########	positive	1
13	Canada	Alberta	########	positive	1
14	Canada	Alberta	1/5/2010	positive	1
15	Canada	Alberta	1/9/2010	neutral	1
16	Canada	Alberta	########	positive	1
17	Canada	Alberta	########	neutral	1
18	Canada	Alberta	########	positive	3
19	Canada	Alberta	########	positive	2
20	Canada	Alberta	########	positive	1
21	Canada	Alberta	########	positive	1
22	Canada	Alberta	########	positive	2

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



Add Layer

Location: 96%
- Country: Country/Region
- State: State/Province
- Add Field

Size: Count (Sum)

Category: Sentiment

Time: Date (None)

Filters

Layer Options

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

References

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