David P. Stonehouse

University of Maryland University College

DATA 650 – Big Data Analytics

Fall 2016, Section 9040, Professor Gortcheva

Assignment 3: BigSQL with Yelp Data

## Lessons Learned from the Walk-through exercise

The BigSQL walk-through demonstrated the power of using standard structured query language (SQL) commands to interact with a Hadoop repository. Hadoop, especially in combination with MapReduce, is designed to support unstructured data. Unstructured data is data generally not a format friendly for machines to read. Using MapReduce jobs, unstructured data can be distilled into a format useful for analysis. For example, MapReduce works well to do textual analysis, transforming human readable words into document term vectors that are suitable for machine processing. In contrast, structured data is for reading directly using something like SQL. This type of data is widely used in many industries. There are many, many users who can program proficiently using SQL, while the talent pool is still emerging for newer technologies related to Hadoop.

The architecture of Hadoop allows very large scale processing of data. Using BigSQL, users can take advantage of that scale using SQL to create and query tables. BigSQL’s syntax complies to standard SQL. It meets important business requirements like user-specific row and column access. IBM ran benchmarks that (as of 2014) showed performance better than Impala and Hive, which are Hadoop ecosystem technologies that provide an SQL interface. In the trial, BigSQL was shown to have richer SQL support, such that users did not need to rewrite their queries to change over to using BigSQL.

The walk-through exercise covered the steps of creating and querying data in BigSQL. The application runs “in the cloud”. After logging in, the first step is to create a table using an SQL CREATE HADOOP TABLE command. Other than the word “HADOOP” to identify the technology for the table, the remaining syntax complies with standard SQL. After running the command, BigSQL provides the execution status. Using the Explore Databases feature, you can see the structure of the data: its columns and attributes. There are multiple ways to add data to BigSQL. In normal use, a LOAD command is used to pull in data from various file formats or relational database management systems (RDBMs). For the walk-through (and for part 2), the INSERT SQL command was used since only a few rows were added. It was noted that values for character variables needed to be specified inside of single quotes.

Using standard SQL QUERY commands, the data was explored. BigSQL shows the results in the lower pane. During exploration of the results, rows can be filtered, and columns can be reordered or removed from view.

On a subsequent step, SQL commands were used to do a query against multiple tables using a JOIN operation. As a result, one row of response was given combining the values from the tables. This join statement was incorporated into a CREATE VIEW statement to create a view. Views are discussed further in a later section. A couple of queries were performed using functions. The first provided the aggregate value for the whole table. The second was done with the GROUP BY clause. For each distinct value for the variable selected by the GROUP BY clause, the rows were aggregated together. For example, for each distinct value of PRODUCT\_COLOR\_CODE, the AVG(GROSS\_PROFIT) value would be for the set of rows for each color. When the ROLLUP function is also added, the results will include the aggregate at the top level.

## Statement of Work

Yelp is a web service that allows users to share and find opinions on businesses, especially restaurants. Yelp has provided data for academic data scientists to study and mine for useful patterns. The data includes information about reviews, reviewers and businesses. Yelp hopes to turn this data into useful insights about reviewers and businesses and other trends in reviews. The Yelp marketing department is hoping to use trend information about reviews of restaurants for a particular town to assist with a service for restaurants to do local marketing. The Yelp application developers hope to make use of information about how a specific review compares with other reviews by the same reviewer and other reviews for the same business.

The academic team from UMUC has accepted the challenge to make this data available to Yelp in a useful way. The data on reviews, reviewers, and businesses will be loaded into Hadoop using BigSQL. Then initial queries will be performed to demonstrate the usefulness of the database. It is hoped that Yelp marketing and application developers will gain the advantage of flexible reporting using SQL to filter, aggregate and sort reviews according to information about reviews, reviewers and businesses.

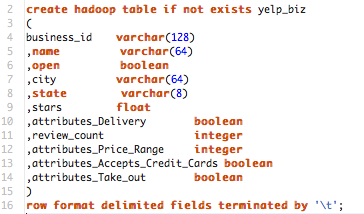
## Database Implementation

Yelp provided data that the team transformed into SQL commands to create, load and query tables. The original data was in JSON format, one file for reviewers (users), one for businesses and one for reviews. Yelp provided a convertor to convert JSON to comma separated value (CSV) format. The CSV files were a good starting point for SQL commands, but first the team had to do a series of transformations.

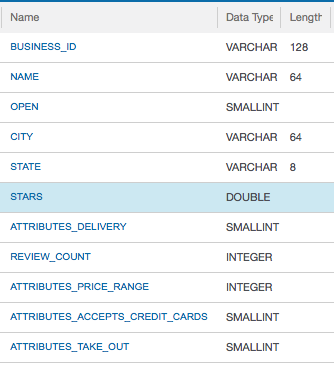
In Excel, the data sets were trimmed down to a subset of columns that would be useful for the original study. Each table had a primary key – business\_id, user\_id and review\_id. The review data set had two foreign keys: business\_id and user\_id. Some columns had embedded carriage returns and commas – those columns were removed. About eight columns were kept for each data set.

In a text editor, the variable names had periods and spaces. Those were replaced by underscores. The string “NULL” was inserted into positions with no value. All character values were surrounded by single quotes. The two date values in the data sets had to be transformed to a valid SQL format. The CSV file had “true” and “false” for Boolean values. Those values would work for many SQL implementations, but for BigSQL, the false/true values had to be converted to zeroes and ones.

The following code was used to create the table for the businesses identified in Yelp:



The key to the table is *business\_id*. The *open* column indicates if the business is open. Some key attributes are where the restaurant is located and the average rating given to it: the *stars* column. There are attributes tracking whether the restaurant provides delivery, take out, or accepts credit cards.

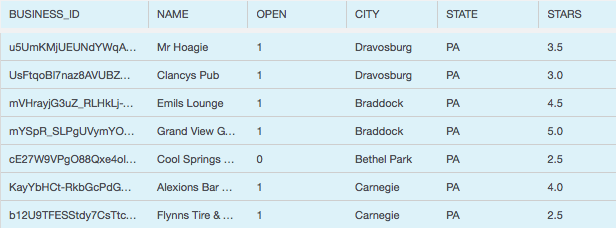


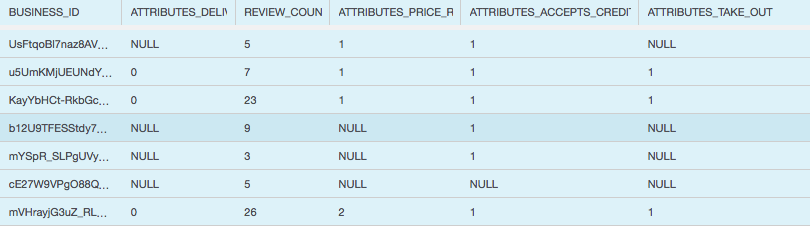
The following code added 7 records to the table. Note that all of these businesses are in Pennsylvania.



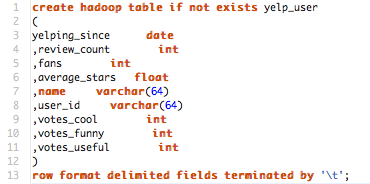
A couple of queries were performed to verify that the data was loaded properly:

../../Desktop/Screen%20Shot%202016-10-27%20at%205.55.08%20PM.png

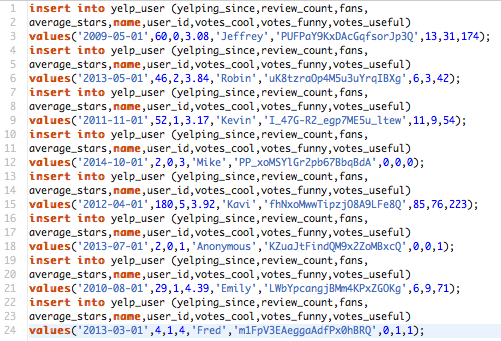
../../Desktop/Screen%20Shot%202016-10-30%20at%208.25.10%20PM.png



The second table is for yelp users. The key to the table is *user\_id.*  Some important information is *yelping\_since,* which indicates how long the user has been on Yelp. The row of information tracks the average of the start they gave in their reviews and the votes that their reviews have received for being cool, funny or useful.

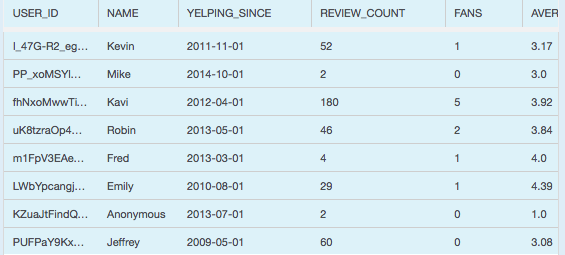


The following 8 rows were added to the table.

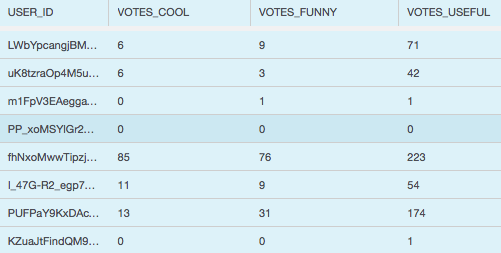


The following queries verify the data in the table. Note the wide range of values for review\_count. Jeffrey has almost as many votes\_useful as Kavi, but with a third as many reviews given.

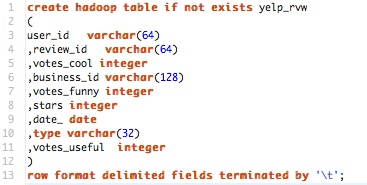
../../Desktop/Screen%20Shot%202016-10-27%20at%206.26.02%20PM.png



../../Desktop/Screen%20Shot%202016-10-27%20at%206.27.26%20PM.png



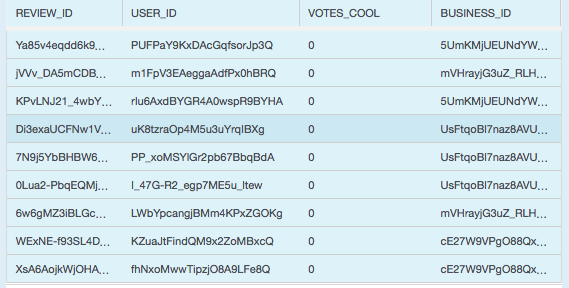
The last table contains information about reviews. The key to the table is *review\_id.* It has two foreign keys into the other two tables: business\_id and user\_id. Note that the *stars* and *votes\_useful* columns in this table are integers, because those are the values for one review, whereas the columns in the other two tables are of type float because they store the average of stars related to the business or given by the user.





The following SELECT commands validate that the INSERT commands were successful.

../../Desktop/Screen%20Shot%202016-10-27%20at%206.15.57%20PM.png



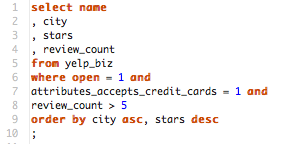
../../Desktop/Screen%20Shot%202016-10-27%20at%206.18.43%20PM.png

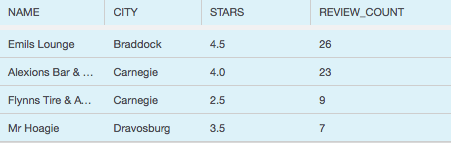


At one point, a row was accidentally added twice. Because BigSQL is based on Hadoop, BigSQL attempts to delete the extra row using DELETE..WHERE failed. In order to clean up that mistake, the whole table was dropped, recreated and reloaded.

## Using the database to retrieve data

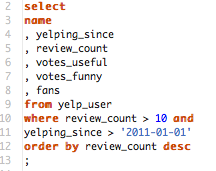
The first query was done against the business table to demonstrate the power of SQL to research businesses. This query filters out businesses are closed, ones that don’t have many reviews, and ones that don’t accept credit cards. These results were ordered by city and then by descending stars. This type of query response could be helpful for a user going through a state wanting to get an idea of the restaurants in places that yelp users frequented.

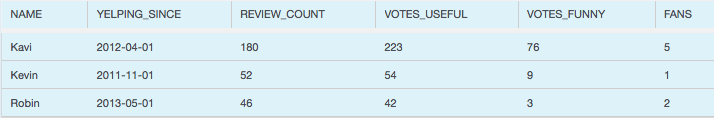




Looking at these results starts to read like a travel guide. This could help a yelp user planning to travel across a state like Montana and see what the best places are to stop, from a Yelp perspective.

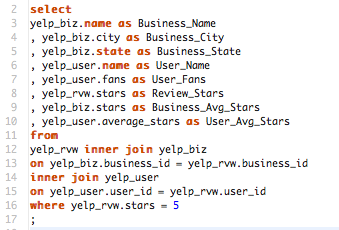
The second query can help the Yelp marketing department as they reach out to the Yelp community of reviewers.

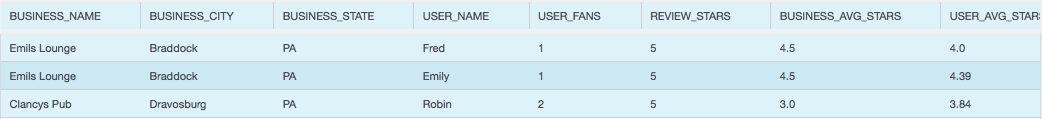




These results start to give a picture of which users are not only active, but maybe more importantly, who is more helpful (VOTES\_USEFUL) and developing the Yelp brand (VOTES\_FUNNY and FANS).

The next query was done with a join of the three tables. The review table has foreign keys to the other two tables. An inner join means that the rows for this result will include rows where the business\_id and user\_id in the review table have matching data in the other two tables. The application developers are considering some enhancements that help users to have some context for the review they are reading.

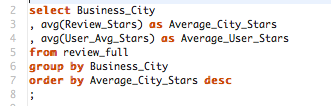


In this case, the user can see not only the stars for the review but whether the number of stars is unusually high for this user or this business. The person looking at the review could use this enhanced info to either skip this review if the user always gives 5 stars, or really pay attention if the business normally gets a 3.

An important feature of SQL is the ability to setup views. Mullins (1994), provides some guidance for how they should be used. Views provide a convenience to users so that a joined set of tables can be queried as if they were one table. Views can be used to implement security and limit access. They are a helpful way to make derived columns available to users. A drawback (or at least a consideration) with views is that they need to be updated when the tables they rely on are changed. The following view gives users a way to query for reviews as a simple table, pulling in information from the users and business table that would be quite normal for filtering reviews. For example, it would be good to compare Review\_Votes\_Cool next to User\_Votes\_Cool to understand if the review is consistent with other reviews. Also, in this case, most users are only interested in reviews for businesses that are open, so the WHERE clause is a nice benefit to users.



The following query against the view gives the marketing department the ability to work with towns to consider how to improve their image on Yelp.



The query results indicate that the average stars for businesses in Braddock are higher by almost a star than the average of each users’ average rating given. This isn’t a perfect measure of how they relate to the typical reviews given for restaurants, but is a pretty good indication.



## Summary, Recommendations and Future Research

By loading data into BigSQL, Yelp can take advantage of powerful query functionality to study data related to reviews. This data can help them provide more interesting information top their users while reading reviews inside the Yelp application. It also can provide the marketing department with key information to engage active reviewers and the chamber of commerce of various cities to work on their image on Yelp.

In terms of differentiators for BigSQL, IBM points to benchmark studies that would indicate that BigSQL is best in class for performing SQL queries on a Hadoop platform. It also provides some key capabilities related to row and column level access. This could be valuable for limiting what internal and external users of the database can see, particularly concerning attributes of users. There were not articles about the cost-effectiveness of BigSQL. The Hadoop infrastructure supports low-cost hardware implementations, so it would seem that BigSQL has potential to be lower cost than traditional data warehouse platforms. A drawback for BigSQL relative to RDBMSs is that it does not allow for flexible deletion or update of records. This is because of the limits of Hadoop as the underlying storage system.

Yelp could extend the initial study by looking at the relationship between some more of the columns. For example, it would be interesting to understand whether price range, delivery, takeout or accepting credit cards limits or increases the number of reviews for businesses on Yelp. This could be used by Yelp to improve how it encourages more reviews for, for example, businesses that do deliveries or take out. Also, Yelp could make use of the Yelping\_Since field in the user table and the date of reviews to understand the pattern for people creating reviews. Does the behavior tend to dwindle over time? Is there often an initial period of activity and then no more reviews?

## References

IBM, “IBM Knowledge Center - Data types that are supported by Big SQL”. Retrieved from <http://www.ibm.com/support/knowledgecenter/SSPT3X_4.0.0/com.ibm.swg.im.infosphere.biginsights.dev.doc/doc/biga_numbers.html>

IBM, 2014, “IBM Hadoop-DS Benchmark Report - 30TB”, retrieved from <http://www.slideshare.net/GJSissons/ibm-hadoop-ds-benchmark-report-30-tb?cm_mc_uid=13877598593914741480505&cm_mc_sid_50200000=1477866898>

<http://www.ibm.com/developerworks/library/bd-bigsql/>

Mullins, C.S., “Views on Views”, retrieved from <http://www.craigsmullins.com/viewnw.htm>

## 