ECE656 COURSE PROJECT
Data Cleaning & Indexing and data analysis and imposing permission on YELP Dataset

Jihua Xu 20751990
Jiatong He 20738590
Xin Chen 20705271
University of Waterloo
200 University Avenue West Waterloo, ON, Canada N2L 3G1

Abstract— This report will analyze the Yelp dataset to see what knowledge can be gleaned from it. Before starting an analysis, this project will firstly construct a new E-R diagram, clean and index the dataset, and then use different classification methods to analyse the data in the dataset. We also create application which allows different users to have different access to the database.

Keywords— YELP dataset, Data cleaning, Data indexing

I. Introduction

Yelp was founded in 2004 by former PayPal employees Russel Simmons and Jeremy Stoppelman. Yelp grew quickly and raised several rounds of funding. By 2010 it had $30 million in revenues and the website had published more than 4.5 million crowd-sourced reviews. From 2009 to 2012, Yelp expanded throughout Europe and Asia. In 2009 it entered several negotiations with Google for a potential acquisition. Yelp became a public company in March 2012 and became profitable for the first time two years later. As of 2016, Yelp.com has 135 million monthly visitors and 95 million reviews. The company's revenues come from businesses advertising. The goal of this project is to extract some subsets of the whole dataset, and then clean and index the samples using several methods, and finally analyze what will influence the business performance.

II. Dataset and Problems

A. Background about Yelp dataset

The Yelp dataset is a subset of our businesses, reviews, and user data for use in personal, educational, and academic purposes. Available in both JSON and SQL files, use it to teach students about databases, to learn NLP, or for sample production data while you learn how to make mobile apps.
For our project, we used the dataset provided by Yelp which can be found online. All data is provided in sql format and is divided into commerce, registration, comments, tips and user information.
It is easy to find what content in dataset from their names. For example, locations, average rating and reviews can be found in business dataset. As for the user dataset, it contains all user information registered on the Yelp. The private information has been removed from the dataset to protect users' privacy. However, we still can get users' names, the number of comments, users' time using Yelp, and other interesting information, such as different kinds of votes that users send to other people's tips and comments. In addition, relationships between users are reflected in the dataset.

In this project, we use a relational database to analyze the data.

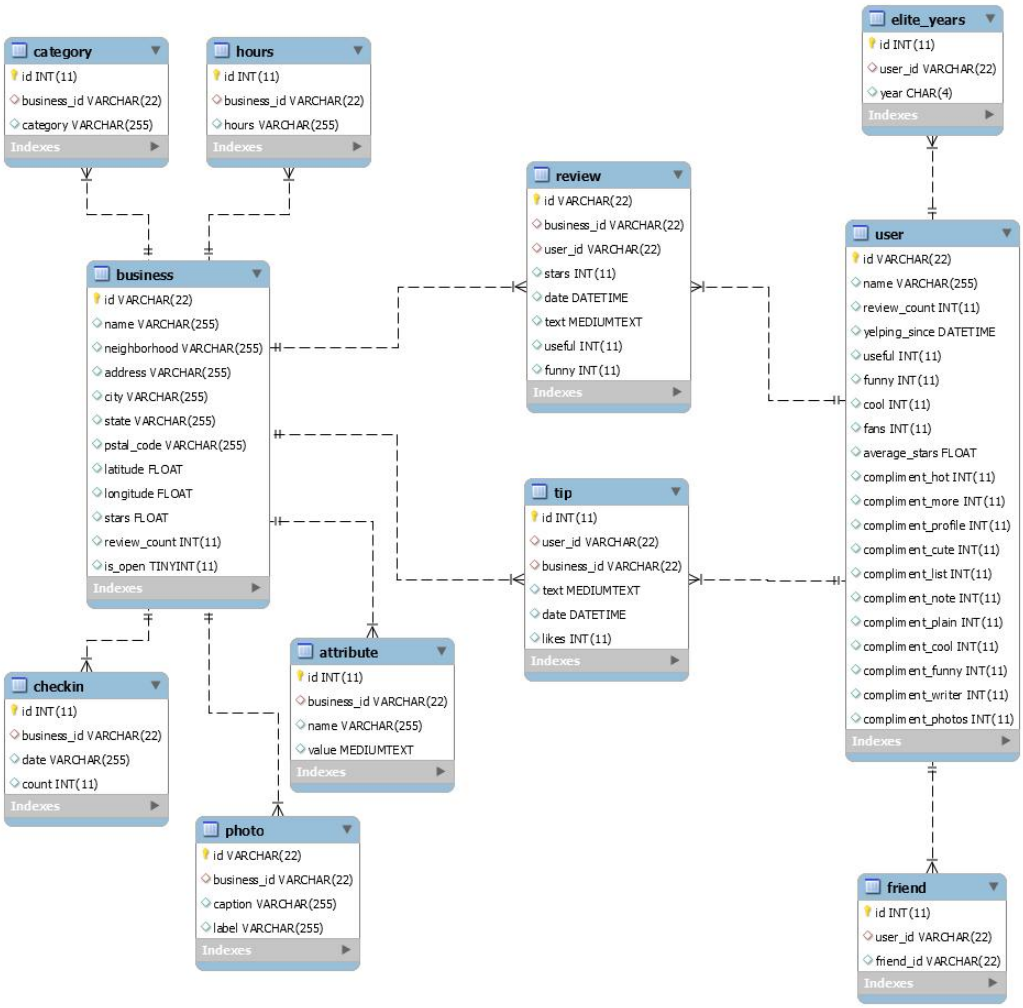

Fig. 1. ER Diagram of the Original YELP Dataset

B. Problems

Before starting an analysis, we had to clean the data. Because there were a lot of data in the database, while some on them were garbage data like some missing value or error value. In order to remove these useless data, cleaning is the first thing we have to do.

III. Data Preprocessing

Cleaning and indexing the data was the first thing we had to do for decreasing the operating time.

A. Foreign key

The table we focused on were the table business and the table user. In order to improve the efficiency of data processing, we selected some key data from the original database and them insert them into a new one. How to add foreign keys just like the following:

```
ALTER TABLE hours ADD FOREIGN KEY ( business_id ) REFERENCES business ( id ) ;
ALTER TABLE checkin ADD FOREIGN KEY ( business_id ) REFERENCES business ( id ) ;
ALTER TABLE tip ADD FOREIGN KEY ( business_id ) REFERENCES business ( id ) ;
ALTER TABLE tip ADD FOREIGN KEY ( user_id ) REFERENCES user ( id ) ;
```

ALTER TABLE review ADD FOREIGN KEY (business_id) REFERENCES business (id) ;
ALTER TABLE review ADD FOREIGN KEY (user_id) REFERENCES user (id) ;
ALTER TABLE photo ADD FOREIGN KEY (business_id) REFERENCES business (id) ;
ALTER TABLE friend ADD FOREIGN KEY (user_id) REFERENCES user (id) ;
ALTER TABLE elite_years ADD FOREIGN KEY (user_id) REFERENCES user (id) ;
ALTER TABLE category ADD FOREIGN KEY (business_id) REFERENCES business (id) ;
ALTER TABLE attribute ADD FOREIGN KEY (business_id) REFERENCES business (id) ;

```
mysql> ALTER TABLE photo ADD FOREIGN KEY ( business_id ) REFERENCES business ( i
d ) ;
Query OK, 206949 rows affected (1 hour 7 min 54.97 sec)
Records: 206949  Duplicates: 0  Warnings: 0
```

Fig. 2. Adding Foreign Key

While in the process of selecting data, we found that the table hours was not in the form of 1NF and then split the attribute into more varchars like Monday_start, Monday_end and so on, the new ER model is shown following.

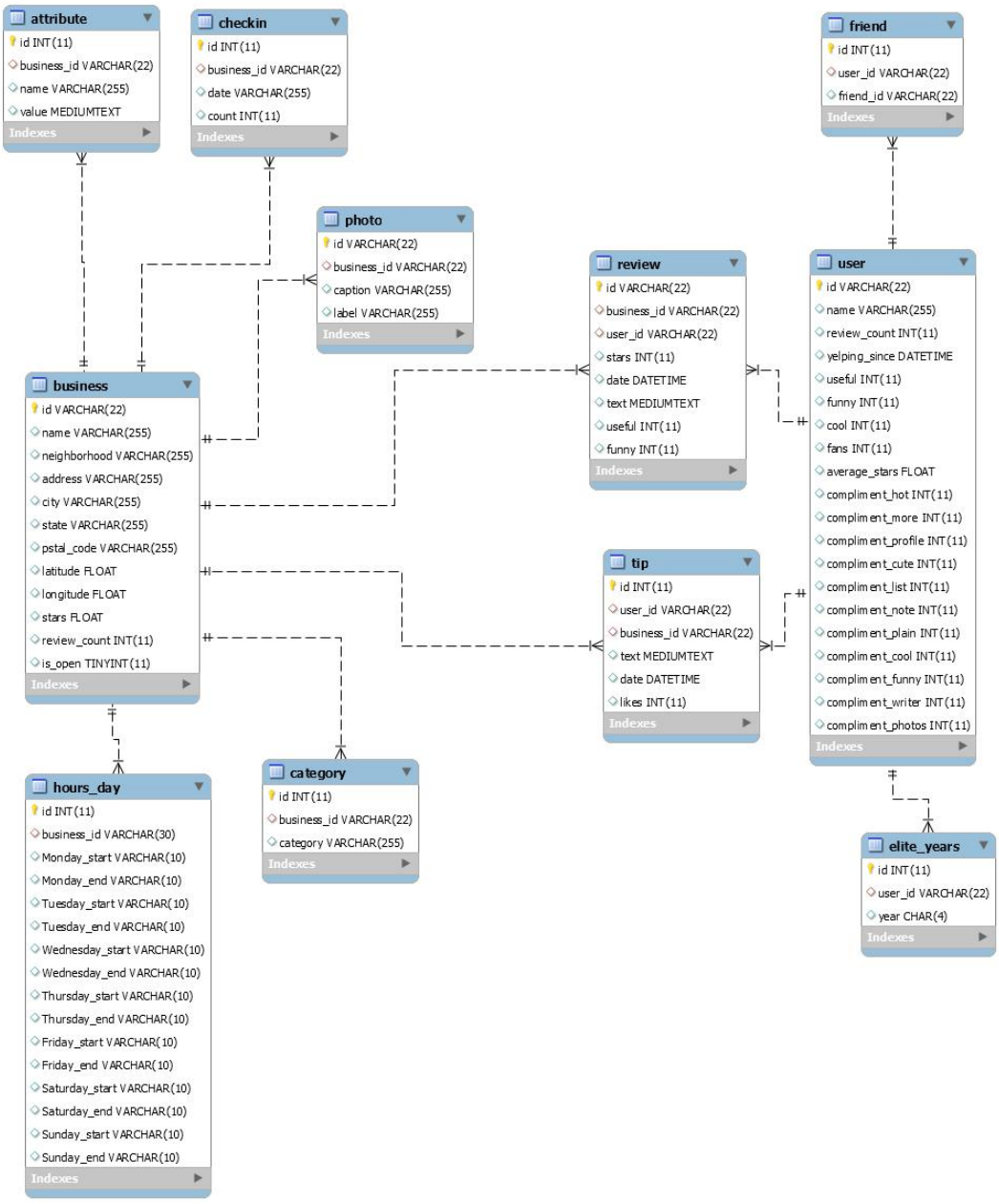

Fig. 3. Modified ER Diagram

B. Data Cleaning

There was an extremely large quantity of data stored in the database, therefore there must be some garbage data which does not satisfy basis sanity checks. To handle such problems, that is, data generally has to be viewed as correct, we have to cleaning the data before further analyzing.
When executing the data cleaning, there is an error because MYSQL cannot delete or update the parent row, so we run the following sql statement to turn off foreign key detection:

```
SET FOREIGN_KEY_CHECKS = 0;
```

Business table:

It would be time consuming to scan the whole database, so we delete some data which will not influence the following data analysis. We first delete the entities in which the businesses are not open anymore, because when we want to recommend some business to other customers, we only choose open businesses. The sql statement and the query result is shown following:

```
mysql> delete from business where is_open = 0;
Query OK, 27865 rows affected (4.50 sec)
```

Fig. 4

The average star in the review table for a business can not be much different from the star in the business table. And if the review_count in the business table and the total number of the review for the same business are different, update the review count to the smaller number.

User table:

We cleaned the entities in the user table in which year is greater than 2018 and less than 2004. The query statement and result is shown following.

```
mysql> delete from user WHERE YEAR(yelping_since) < 2004;
Query OK, 0 rows affected (2.30 sec)

mysql> delete FROM user WHERE YEAR(yelping_since) >2018;
Query OK, 0 rows affected (2.33 sec)
```

Fig. 5

The average star in the review table written by a user can not be much different from the average star in the user table. The number of reviews written by a user as recorded in the User table cannot be smaller than the number of reviews in the sample set.

Other tables:

The user id and business id appearing in the tables must appear in the sample set. One of the queries to clean the data is following:

```
mysql> delete from photo where business_id not in (select id from business);
Query OK, 24758 rows affected (43 min 4.75 sec)
```

Fig. 6

We also clean the incorrect data, for example, the year which is before 2004 and after 2018, because yelp is founded in 2004 and data for the future cannot appear.

C. Data indexing

Another problem was that since we only need use some attributes to do the analysis, it would be time consuming to scan the whole database. We constructed a new database to simply the original data together with several indexes for fastening operations.

Category table:

A B-Tree index was created for column business_id. The queries of creating index and select statement are shown following.

```
ALTER TABLE `category` ADD INDEX `idx_category_businessid` (`business_id`) USING BTREE;
SELECT * FROM category where substr(business_id,1,1) = 'X';
```

```
mysql> ALTER TABLE `category` ADD INDEX `idx_category_businessid` (`business_id
) USING BTREE;
Query OK, 0 rows affected (32.57 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig. 7. Adding Index in Category Table

After adding the BTREE index, the queries searching for category records based on the condition of column business_id increases obviously. The following pictures show the comparison between the same query time before and after the creation of the index.

```
21134 rows in set (9.98 sec)
```

Fig. 8. Before Adding the Index

```
21134 rows in set (0.88 sec)
```

Fig. 9. After Adding the Index

Attribute table:

The query of creating a BTree index for business_id and the corresponding query of checking the result of the index are shown following.

```
ALTER TABLE `attribute` ADD INDEX `idx_attribute_businessid` (`business_id`) USING BTREE;
SELECT * FROM attribute where substr(business_id,1,1) = 'X';
```

The following pictures show the comparison between the query time before and after creating the index.

```
41382 rows in set (35.23 sec)
```

Fig. 10. Before Adding the Index

```
41382 rows in set (1.98 sec)
```

Fig. 11. After Adding the Index

Checkin table:

The query of creating a BTree index for business_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `checkin` ADD INDEX `idx_checkin_businessid` (`business_id`) USING BTREE;
SELECT * FROM checkin where substr(business_id,1,1) = '7';
```

The following pictures show the comparison between the query time before and after creating the index.

```
60514 rows in set (39.97 sec)
```

Fig. 12. Before Adding the Index

```
60514 rows in set (3.86 sec)
```

Fig. 13. After Adding the Index

Tip table:

The query of creating a BTree index for user_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `tip` ADD INDEX `idx_tip_userid` (`user_id`) USING BTREE;
SELECT * FROM tip where user_id='blrWvPePSv87aU9hV1Zd8Q';
```

The following pictures show the comparison between the query time before and after creating the index.

```
236 rows in set (10.75 sec)
```

Fig. 13. Before Adding the Index

```
236 rows in set (0.23 sec)
```

Fig. 14. After Adding the Index

The query of creating a BTree index for business_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `tip` ADD INDEX `idx_tip_businessid` (`business_id`) USING BTREE;
SELECT * FROM tip where business_id='k7WRPbDd7rztjHcGGkEjlw';
```

The following pictures show the comparison between the query time before and after creating the index.

```
11 rows in set (10.50 sec)
```

Fig. 15. Before Adding the Index

Fig. 16. After Adding the Index

Review table:

The query of creating a BTree index for business_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `review` ADD INDEX `idx_review_businessid` (`business_id`) USING BTREE;
select count(distinct business_id) from review;
```

The following pictures show the comparison between the query time before and after creating the index.

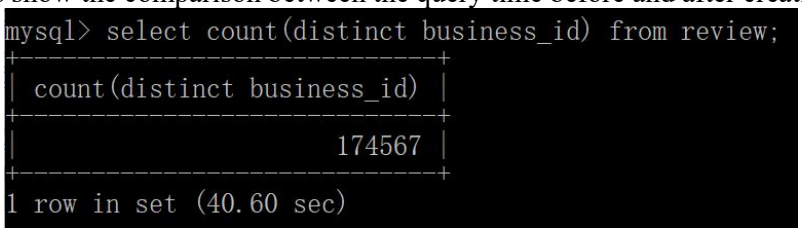

Fig. 17. Before Adding the Index

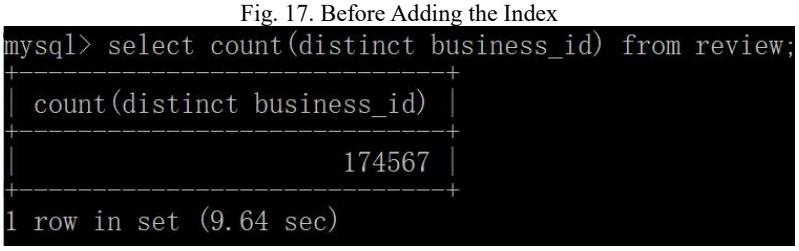

Fig. 18. After Adding the Index

The query of creating a BTree index for user_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `review` ADD INDEX `idx_review_userid` (`user_id`) USING BTREE;
select count(distinct user_id) from review;
```

The following pictures show the comparison between the query time before and after creating the index.

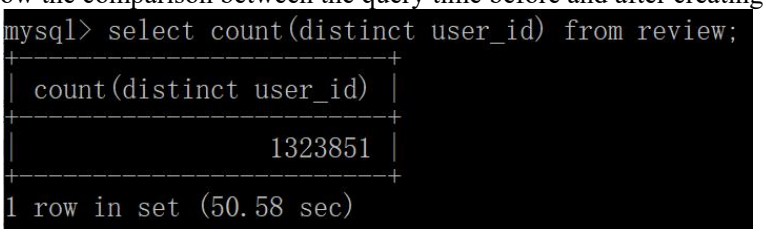

Fig. 19. Before Adding the Index

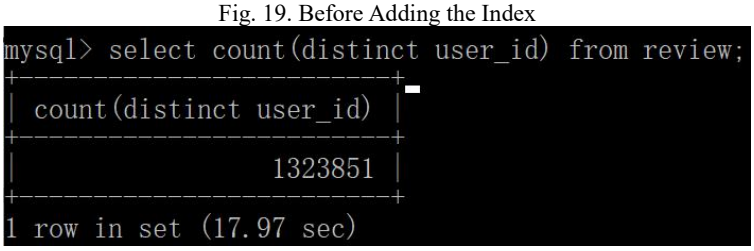

Fig. 20. After Adding the Index

Friend table:

The query of creating a BTree index for user_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `friend` ADD INDEX `idx_friendid_userid` (`friend_id`) USING BTREE;
select count(distinct friend_id) from friend;
```

The following pictures show the comparison between the query time before and after creating the index.

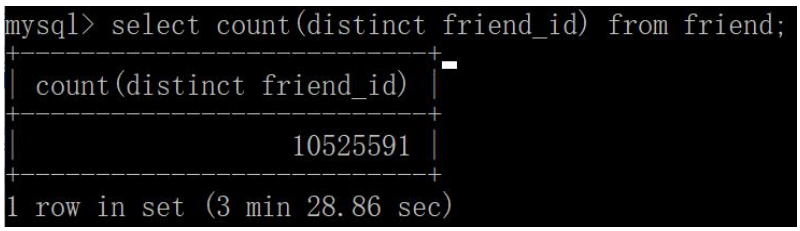

Fig. 21. Before Adding the Index

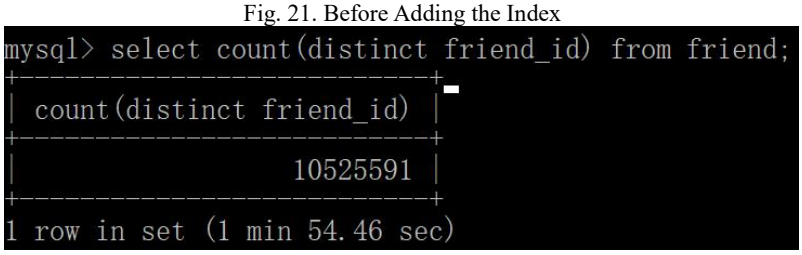

Fig. 22. After Adding the Index

The query of creating a BTree index for user_id and the corresponding query of checking the result of the index are shown in following.

```
ALTER TABLE `friend` ADD INDEX `idx_friend_userid` (`user_id`) USING BTREE;
  select count(distinct user_id) from friend;
```

The following pictures show the comparison between the query time before and after creating the index.

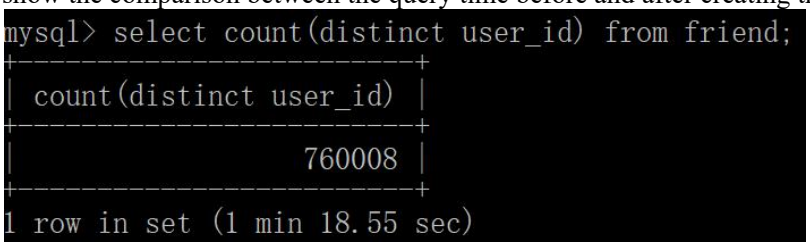

Fig. 23. Before Adding the Index

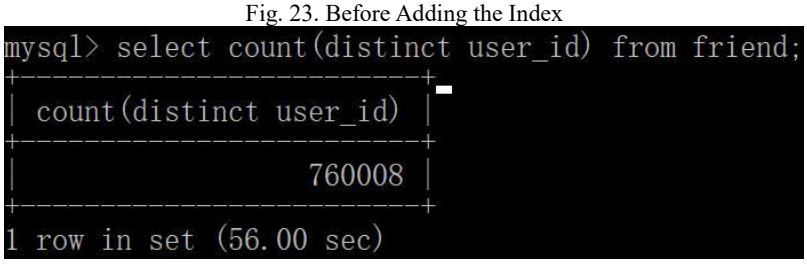

Fig. 24. After Adding the Index

Hours_day table:

The query of creating a BTree index for business_id and the corresponding query of checking the result of the index are shown in following:

```
ALTER TABLE `hours_day` ADD INDEX `idx_hours_day_businessid` (`business_id`) USING BTREE;
  (SELECT id
   FROM hours_day
     LEFT JOIN `business`
       ON hours_day.business_id= `business`.`id`
   WHERE `business`.`id` IS NULL
  );
```

When executing the query of joins, the time exceeds more than 1 hour, so we terminate the query early and the following picture show the query time after creating the index.

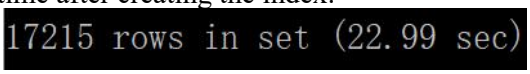

Fig. 25. After Adding the Index

Summary

The comparison between the query time before and after adding index is shown above. We can see that after adding indexes, the queries can be much faster, especially some queries whose time exceed some reasonable thresholds (e.g. sizable joins when indexes are not present).

IV. Data Analysis

A. Based on the data, determine if a business is declining or improving in its ratings

To analyze whether a business is declining or improving its ratings, we simply choose a specific business and get all its rating in time sequence, after which we renew the rating day by day. This process record two values, average rating up to now and total number of days in record. Every time we get a new data of a latest day, we calculate the average as:

$$\text{average}_{\text{up\_to\_now}} = \frac{\text{average}_{\text{before\_today}} * \text{number}_{\text{record\_before}} + \text{average}_{\text{today}} * \text{number}_{\text{today}}}{\text{number}_{\text{record\_before}} + \text{number}_{\text{today}}}$$

To judge whether the business is getting a better fame, we simply compare $\text{average}_{\text{up\_to\_now}}$ and $\text{average}_{\text{before\_today}}$:

$$\text{average}_{\text{up\_to\_now}} - \text{average}_{\text{before\_today}} = \frac{(\text{average}_{\text{today}} - \text{average}_{\text{before\_today}}) * \text{number}_{\text{today}}}{\text{number}_{\text{record\_before}} + \text{number}_{\text{today}}}$$

If the red part is greater than 0, then we could claim that the business is now providing a better service than it used to be and vice versus. However, this assumption holds when the data set is large enough to exclude some outliers. If some users rate the business randomly and dataset is not large enough then this method fails to determine whether the business is improving its service due to a large part of anomalies. Thus, we applied this method to business with large number of records in relation Review:

(1) Choosing business with certain amount of records:

```
mysql> select business_id, count(*) from review group by business_id order by count(*) desc limit 10
+------------------------+----------+
| business_id            | count(*) |
+------------------------+----------+
| 4JNXUYY8wbaaDmk3BPzlWw |     6978 |
| RESDUcs7fIiihp38-d6_6g |     6412 |
| K7lWdNUhCbcnEvIONhGewg |     5633 |
| cYwJA2A6I12KNkm2rtXd5g |     5431 |
| DkYS3arLOhA8si5uUEmHOw |     4790 |
| f4x1YBxkLrZg652xt2KR5g |     4371 |
| eoHdUeQDNgQ6WYEnP2aiRw |     3913 |
| 2weQS-RnoOBhb1KsHKyoSQ |     3873 |
| KskYqH1Bi7Z_61pH6Om8pg |     3839 |
| ujHiaprwCQ5ewziu0Vi9rw |     3698 |
+------------------------+----------+
10 rows in set (4.53 sec)
```

Fig. 26. Result of the Query for business_id with the Most Records in Review

(2) Plotting the average rating in time sequence:

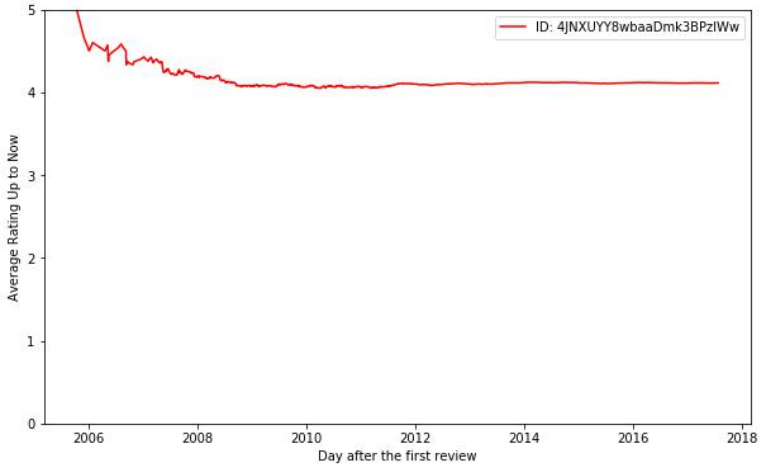

Fig. 27. The Average Rating Updated Every Day

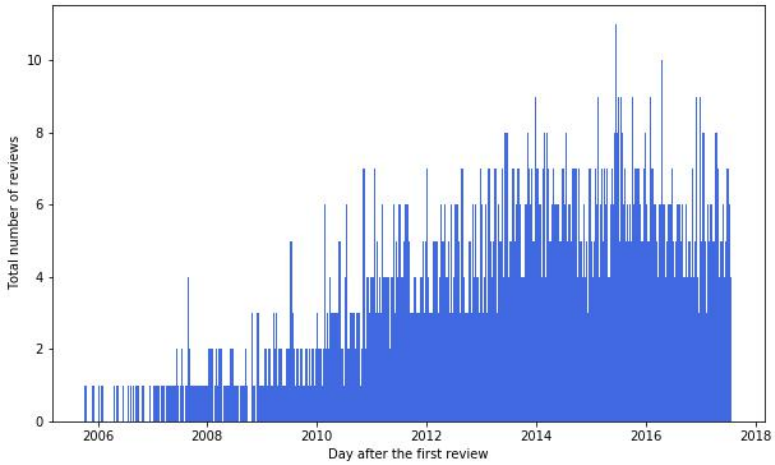

Fig. 28. Number of Reviews for A Certain Business Every Day

As we could see, in early period, the average rating fluctuates since during this period, the number of reviews it gets is quite small(around 1 review per day) and thus every rating it got might have a large influence on the average rating. After around 2 years in 2008, the rating of the business gets stable around 4, and since then the number of reviews it got every is no less than 3 thus we could evaluate whether the business is getting better in the short run. Fluctuating upward means the restaurant is providing better service during that period while downward a relatively less satisfying service. As we could see from the figure, this restaurant is keeping its high level service all the time. Based on the large number of reviews it gets and its relatively stable historical rating level, we could deem it an ideal place for dinner.

B. Predicting what a user might rate a business

Normally people would rate a business based on not only his favor but reviews of others. Assuming that we get the historical data of a user including the average rating of the business that the user has ever rated and the rating given by the user, we wanted to learn from the history and predict what the user would rate a certain business given the average rating of the business.

Priori Algorithm is adopted during this task. The process of the prediction is as followed:

(1) Collecting all records of a certain user as [average rating, (user_rating1, user_rating2…)][4.5, (5.0, 4.0…)] means for business which the user has ever visited is rated 4.5, the user has ever rated it 5.0, 4.0 and so on.

(2) Calculating the confidence level of each rating given by the user based on a certain average rating. (given 4.5, confidence level of (4.5, 5.0) is 0.2 means 20% of ratings is 5.0)

(3) Given a number between 0 and 5, find the closest average rating and list all ratings given by the user with the confidence level higher than the pre-determined level (for instance 0.7)

```
Average Rating: 4.8    User Rating: 4.0   Frequency:   8   Confidence Level: 0.800000
Average Rating: 4.8    User Rating: 5.0   Frequency:   2   Confidence Level: 0.200000
Average Rating: 3.8    User Rating: 1.0   Frequency:   2   Confidence Level: 0.009259
Average Rating: 3.8    User Rating: 2.0   Frequency:   9   Confidence Level: 0.041667
Average Rating: 3.8    User Rating: 3.0   Frequency: 117   Confidence Level: 0.541667
Average Rating: 3.8    User Rating: 4.0   Frequency:  86   Confidence Level: 0.398148
Average Rating: 3.8    User Rating: 5.0   Frequency:   2   Confidence Level: 0.009259
Average Rating: 2.4    User Rating: 2.0   Frequency:  10   Confidence Level: 0.277778
Average Rating: 2.4    User Rating: 3.0   Frequency:  25   Confidence Level: 0.694444
Average Rating: 2.4    User Rating: 4.0   Frequency:   1   Confidence Level: 0.027778
Average Rating: 2.9    User Rating: 4.0   Frequency:  12   Confidence Level: 0.108108
Average Rating: 2.9    User Rating: 2.0   Frequency:  24   Confidence Level: 0.216216
Average Rating: 2.9    User Rating: 3.0   Frequency:  72   Confidence Level: 0.648649
Average Rating: 2.9    User Rating: 1.0   Frequency:   3   Confidence Level: 0.027027
Average Rating: 2.8    User Rating: 4.0   Frequency:   5   Confidence Level: 0.045872
Average Rating: 2.8    User Rating: 2.0   Frequency:  21   Confidence Level: 0.192661
Average Rating: 2.8    User Rating: 3.0   Frequency:  79   Confidence Level: 0.724771
Average Rating: 2.8    User Rating: 1.0   Frequency:   4   Confidence Level: 0.036697

Given the average rating of this restaurant:4.5
According to the record of the user
The user might rate the business as: 4 with the probability of 70 %

Given the average rating of this restaurant:2.8
According to the record of the user
The user might rate the business as: 3 with the probability of 72 %
```

Fig. 29. Process of the Prediction Based on Historical Information

Based on the prediction, we could set the lower bound for the confidence level and the average rating of the business so that we could get all satisfied businesses. Naturally, we should also take the location and the preference of the user into consideration. In the end, we could push these business to the user as recommendations.

C. Relation between operating hours and ratings

To determine the relation between the operating hours and its ratings, we first simply calculated the Pearson Correlation Coefficient as followed:

The correlation between Operating Hours and Average_Rating is -0.189361, and the confidence level is 0.00

The correlation between operating hours and average rating is not obvious enough, therefore we could not simply predict the rating of a business based only on the operating hours. However, we could try to find something more detailed within a certain operating length like 80-90 hours/week. We want to know the percentage of each rating within one operating length.

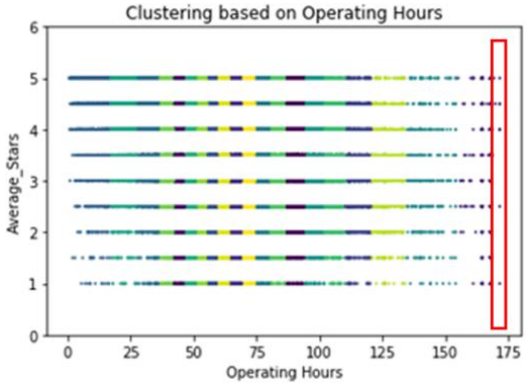

Fig. 30. Distribution of the Operating Hours and Their Ratings

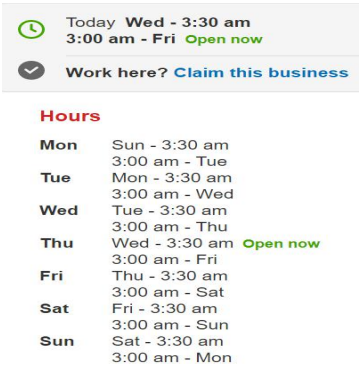

Fig. 31. Operating Time Exceeding 168 Hours/Week

Before we analyze the data, we notice that there should be operating hours over 168 hours (7*24), which is impossible in reality since there is only 168 hours a week. According to the information in there website, we assume that it works 24 hours a day and thus set it to be 168 when its operating hours exceeds 168.

Besides, we set a specific time step and collect all businesses within one time step as a group. For instance, if we set time step as 50, operating hours shorter than 50 should be in the same group and operating hours between 50 and 100 should be in the same group, after which we calculate the distribution of each rating within one time span:

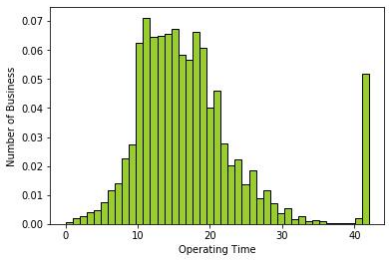

Fig. 32. Distribution of different operating hours

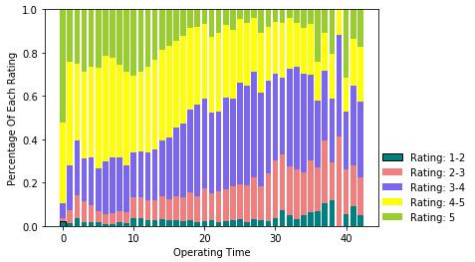

Fig. 33. Percentage of Each Rating within Every Time Range

The time step of the result shown above is 4. As we could see, the distribution of the operating hours is closed to normal distribution. Most of the businesses are open for 40-80 hours a week. And around 5% of businesses are open for almost the whole week 168 hours.

Based on the second figure shown above, a longer operating hours does not guarantee a higher rating. In case that the data are not representative enough, we simply choose the range between 10 and 20. Focusing on the yellow part and green part, with a longer operating time, the less likely that a business will get a rating higher than 4.0 since both these two parts are shrinking. However, the possibility to get higher than 3.0 is relatively stable given the same height of the accumulation of the bottom two parts. The optimal operating hours, indicated by the figure, should be 40-44 hours per week with 30% of the businesses within this range rated as 5.0 and 35% of them rated between 4.0 and 5.0. Those who operate for almost 24 hours every day has a higher lower bound as well with only 20% of which are lower than 3.0.

D. Relation between the length of the review and how people perceive the review

There are three operations for a certain review, useful, funny and cool. Alike the task c, we first calculate the person correlation coefficient between these three factors and the length of the text, and the result is as followed:

The correlation between Useful and Text Length is 0.814828, and the confidence level is 0.000000

The correlation between Funny and Text Length is 0.609083, and the confidence level is 0.000747

The correlation between Cool and Text Length is 0.294650, and the confidence level is 0.135712

Determination of the relation between these three factors and the text length does not seem to work, therefore we tried to find out what could be concluded within a fixed range. Rather than exploring the relation between the text length and the total number of each factors, what we want to know is that how likely is it that people would get at least one reply with a certain text length.

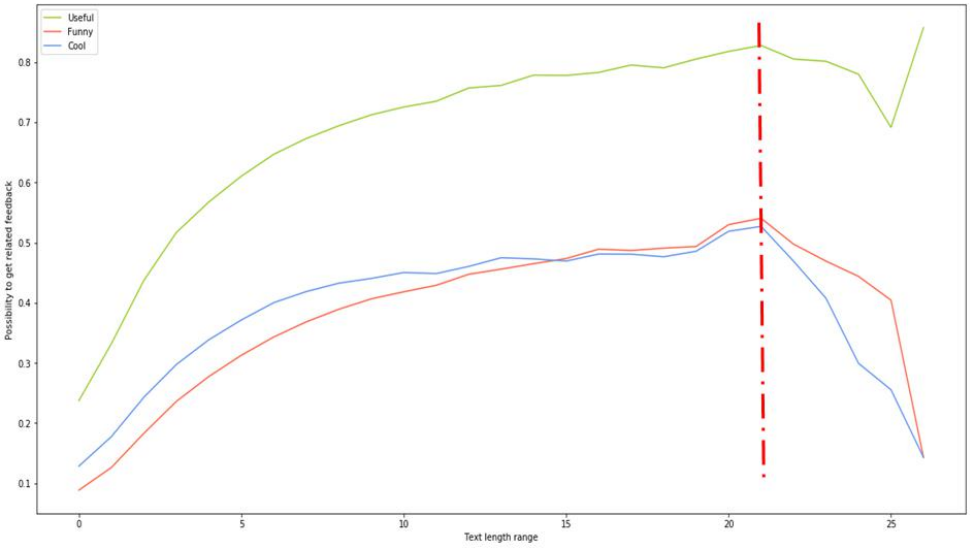

Figure. 34. Possibility to get at least one reply from others

The green curve shows the possibility that people would get at least one ‘useful’ when writing a review with that length. As it indicates, the longer the text is, the more likely that it will be noticed by other users and get at least one ‘useful’. The text length step adopted here is 40, thus if you write a review which has at least 800(20*40)

words. In most cases(around 80%) you would get at least one ‘useful’.

Similar phenomenon could be observed for ‘Funny’ and ‘Cool’ even though things are slightly different when length exceeds 840. Both curves reached the highest level when the length is around 840 with the possibility to be 50%. However, unlike the curve of ‘Useful’, which keeps in high possibility level all the time, both ‘Funny’ curve and ‘Cool’ curve drop sharply when the length exceed 840. They both drop down to 15% with the length of more than 1000 words. Which means that the longer the review, the less likely that people will regard it as a funny or cool one in this range.

The optimal length, concluded from the curves above, should be around 800 words once the user hopes that his review could get at least one reply from others.

V. Impose restrictions

Restrictions we use in this part are to assign different permission to users so that different users have different access to the database. We create five kinds of users, a casual user, a logged in user, a analyst user, a developer user and database admin, who have different permission to the yelp database.

A casual user:

A casual user uses the application to browse search results. These users do not need to have an account and they cannot submit reviews, so their permission is only to SELECT on the tables related to the business with no privilege to create view and other operations on tables, including UPDATE, DELETE and INSERT. They also cannot create new tables and alter tables.

(Select~~, Update, Insert, Drop, Create, Create View, Alter~~)

```
mysql> SHOW GRANTS FOR 'casual'@'localhost';
+-----------------------------------------------------+
| Grants for casual@localhost                         |
+-----------------------------------------------------+
| GRANT USAGE ON *.* TO 'casual'@'localhost'          |
| GRANT SELECT ON `yelp_db`.* TO 'casual'@'localhost' |
+-----------------------------------------------------+
```

Figure. 35. The Permission of Casual User

In order to test the permission of the casual user, we create a casual user and check if the user can execute the operations of INSERT, UPDATE, DROP and CREATE. The result is shown following.

```
mysql> select id, stars from business order by stars limit 5;
+------------------------+-------+
| id                     | stars |
+------------------------+-------+
| -2v6N3EZAE4VDkAf7q3CrA |     1 |
| -17Uz92KtnWOpeEGVvg17g |     1 |
| -Ag8JiHlShYxxOKBHZfXVw |     1 |
| -8PUOZ4Q2Nc-sYOeBj4DiQ |     1 |
| -CAH9wLiFuilgPsERKXL5A |     1 |
+------------------------+-------+
5 rows in set (0.08 sec)

mysql> insert into user values('test_insert');
ERROR 1142 (42000): INSERT command denied to user 'casual'@'localhost' for table 'user'
mysql> update user set user_id = 'test_update';
ERROR 1142 (42000): UPDATE command denied to user 'casual'@'localhost' for table 'user'
mysql> drop table user;
ERROR 1142 (42000): DROP command denied to user 'casual'@'localhost' for table 'user'
mysql> create table test_table(ID int(11));
ERROR 1142 (42000): CREATE command denied to user 'casual'@'localhost' for table 'test_table'
```

Figure. 36. The Result of Casual User

A logged in user:

Logged in users have all the privileges that casual users have. They have their own accounts, so they can leave

reviews for places they visit as they want. In addition to the same permission as a casual user, in order to write the review, a logged in user should be provided INSERT, UPDATA and DROP on the review table. However, they are not allowed to alter the structure of tables and create any view.

(Select, Update, Insert, Drop, ~~Create, Create View, Alter~~)

```
mysql> SHOW GRANTS FOR 'logged_in'@'localhost';
+-----------------------------------------------------------------------------------------+
| Grants for logged_in@localhost                                                          |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'logged_in'@'localhost'                                           |
| GRANT SELECT ON `yelp_db`.* TO 'logged_in'@'localhost'                                  |
| GRANT INSERT, UPDATE, DELETE ON `yelp_db`.`review` TO 'logged_in'@'localhost'           |
+-----------------------------------------------------------------------------------------+
```

Figure. 37. The Permission of Logged in User

A analyst user:

Business analysts can use the application to produce sales reports and may want to do special data mining and analysis. They cannot perform IUD (Insert/Update/Delete) operations on the database but should have access to creating extra views on the database schema. A analyst user should be provided SELECT and CREATE VIEW on all of the tables in the database.

(Select~~, Update, Insert, Drop, Create,~~ Create View~~, Alter~~)

```
mysql> SHOW GRANTS FOR 'analyst'@'localhost';
+-----------------------------------------------------------------+
| Grants for analyst@localhost                                    |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'analyst'@'localhost'                     |
| GRANT SELECT, CREATE VIEW ON `yelp_db`.* TO 'analyst'@'localhost' |
+-----------------------------------------------------------------+
```

Figure. 38. The Permission of Analyst User

In order to test the permission of the analyst user, we create a analyst user and check if the user can execute the operations of CREATE VIEW and DROP. The result is shown following.

```
mysql> select business_id, avg(stars) from review group by business_id limit 5;
+------------------------+-------------+
| business_id            | avg(stars)  |
+------------------------+-------------+
| --6MefnULPED_I942VcFNA |      3.2333 |
| --7zmmkVg-IMGaXbuVd0SQ |      3.9048 |
| --8LPVSo5i0Oo61X01sV9A |      4.3333 |
| --9e1ONYQuAa-CB_Rrw7Tw |      4.0871 |
| --9QQLMTbFzLJ_oT-ON3Xw |      3.0000 |
+------------------------+-------------+
5 rows in set (0.01 sec)

mysql> create view Business_Avg as select business_id, avg(stars) from review group by business_id;
Query OK, 0 rows affected (0.01 sec)

mysql> drop table tip;
ERROR 1142 (42000): DROP command denied to user 'analyst'@'localhost' for table 'tip'
```

Figure. 39. The Result of Analyst User

A developer user:

Developers have more privileges than analysts. Developers working with this database are able to create new tables and perform data cleaning and indexing. They are allowed to perform IUD operations on the database. A developer user should be provided SELECT, INSERT, UPDATE, CREATE, CREATE VIEW and DROP on all of the tables. They are allowed to create index, as we merely grant index to them rather than Alter, index could only be added by creating instead of altering.

(Select, Update, Insert, Drop, Create, Create View~~, Alter~~)

```
mysql> SHOW GRANTS FOR 'developer'@'localhost';
+-----------------------------------------------------------------------------------------------------------+
| Grants for developer@localhost                                                                            |
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'developer'@'localhost'                                                             |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, INDEX, CREATE VIEW ON `yelp_db`.* TO 'developer'@'localhost' |
+-----------------------------------------------------------------------------------------------------------+
```

Figure. 40. The Permission to Developer User

In order to test the permission of the analyst user, we create a analyst user and check if the user can execute the

operations of CREATE TABLE, CREATE VIEW, ALTER and CREATE INDEX. The result is shown following.

```
mysql> create table test_create (ID char(5), Name varchar(20));
Query OK, 0 rows affected (0.06 sec)

mysql> create view test_view as select business_id, avg(stars) from review group by business_id;
Query OK, 0 rows affected (0.01 sec)

mysql> alter table test_create add index test_index (ID);
ERROR 1142 (42000): ALTER command denied to user 'developer'@'localhost' for table 'test_create'
mysql> create index test_index on test_create(ID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Figure. 41. The Result of Developer User

database admin:

The database admin who has full access over the database.

```
mysql> SHOW GRANTS FOR 'host'@'localhost';
+-----------------------------------------------------------------------------------+
| Grants for host@localhost                                                         |
+-----------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'host'@'localhost'                                          |
| GRANT ALL PRIVILEGES ON `yelp_db`.* TO 'host'@'localhost' WITH GRANT OPTION       |
+-----------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Figure. 42. the permission to database admin

VI. Conclusion

In this project, we added foreign keys to the original database and then found that table hours did not satisfy 1NF, and thus we split attribute hours into various varchars. Apart from that, we removed the garbage data from the table for speeding up the following work.

Analysis is the main work in our project. Therefore, we analyzed the data in different ways. The first one was determining if a business is declining or improving in its ratings. We also predicted what a user might rate a business, analyzed the relation between operating hours and ratings, and the relation between the length of the review and how people perceive the review.

The last part is imposing permission. We give different users different access to the yelp database.