DSA5104 Project Report

Group member:

A0275684M LI FANGZHENG,
A0275594M YANG ENZE,
A0275712A ZHANG HANTIAN,
A0275731B ZHANG YINGDONG,
A0275755N ZHAO WENKANG

1. Project Goal

Our project goal is to deeply understand and effectively apply the core processes of data management and retrieval. We will encompass key phases such as data collection, cleaning, modeling, storage, access, and web development. The project will be based on the Yelp dataset, augmented with data obtained using web scraping techniques from the Yelp website, and further expanded through data generation methods. We plan to use data visualization tools for an in-depth analysis of the structure and meaning of data, thoroughly examining the scale, distribution, and other statistical characteristics of the dataset. Following this, we will employ efficient data-cleaning techniques to address quality issues, such as null values, encoding problems, and duplicate records. In terms of data storage, we aim to develop and implement suitable relational and non-relational database models, adhering to the complete database design process, including requirement analysis, conceptual design, logical design, and schema optimization. Moreover, we will design a range of queries that are suitable for both routine operations and complex analysis, with a special emphasis on efficiency and response times, highlighting the differences between relational and non-relational databases. We also plan to develop a web application with a custom front-end interface for a more intuitive display of data and analysis results. Overall, through this project, we not only aim to deepen our understanding of data management and retrieval processes but also to achieve significant progress in team collaboration and innovative solutions in database management.

2. Data Introduction and Visualization

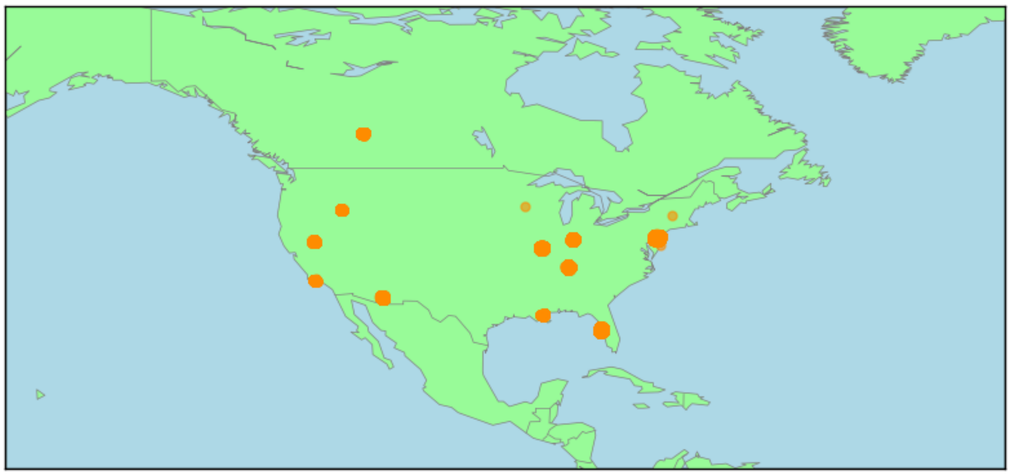
The Yelp dataset is a large, comprehensive dataset used primarily in the fields of data science and machine learning. This dataset contains a large amount of business, user, and interaction data. It was originally in JSON data format and contains 5 forms totaling 8.65G in size.

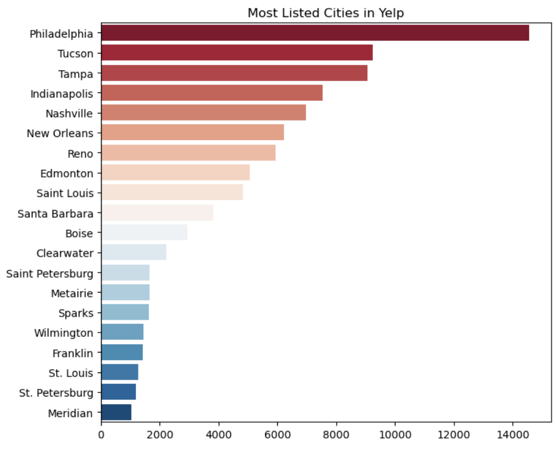
- **business.json**: This file contains detailed information about businesses, such as location, category, rating, opening hours, etc. It might include hundreds of thousands of records, each with multiple fields like address, attributes, city, number of reviews, latitude and longitude, name, postal code, star rating, and more. It consists of 150349 lines and 14 fields, where some of the fields are nested in a dictionary. For example, fields for attributes and categories.

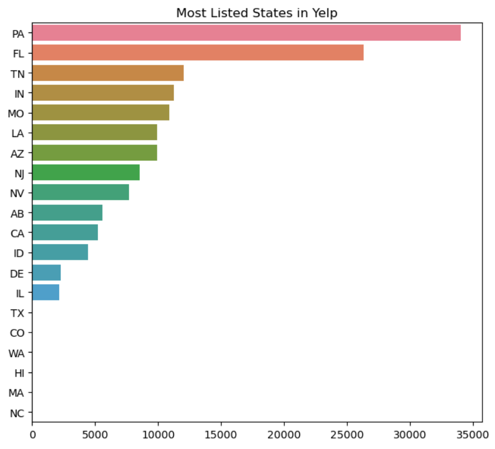
- **review.json**: Contains user reviews of businesses, potentially amounting to several million review records. Each review includes fields such as user ID, business ID, rating, review text, etc. It consists of 6990280 lines and 9 fields, where the text field may be a very long string.
- **user.json**: Includes personal information of users, such as friends, number of reviews, average rating, etc. The user data may contain records of several million users, each including information like the number of friends, number of reviews, average rating, etc. It consists of 19878947 lines and 22 fields, most of which are single attributes describing the user, where the friend field is an indeterminate array.
- **checkin.json**: Records the check-in data of users at specific businesses. This file may contain several million check-in records, each with information like business ID and check-in times. It contains 131930 lines and 2 fields.
- **tip.json**: Contains brief advice or comments from users to businesses. The tips file might include hundreds of thousands of records, each containing user ID, business ID, text content, etc, and contains 908915 lines and 5 fields.

The data visualization focuses on business.json due to the large size of the data and the fact that the key information is concentrated in business.json.

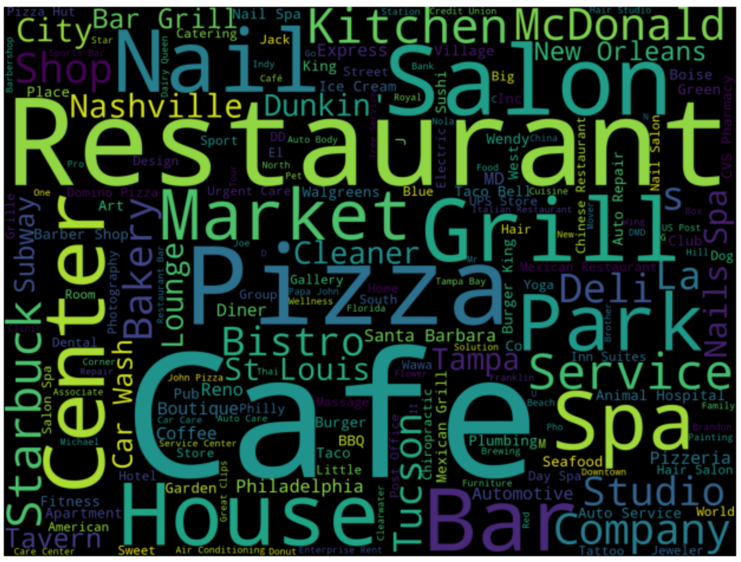
1) First, we analyze the location of those businesses and find the cities and states. We find that most of these businesses are located in some of the cities. We also relied on statistical histograms to count the number of BUSINESSES in each state and city in the Yelp dataset. This was helpful in our subsequent design and evaluation of query statements.



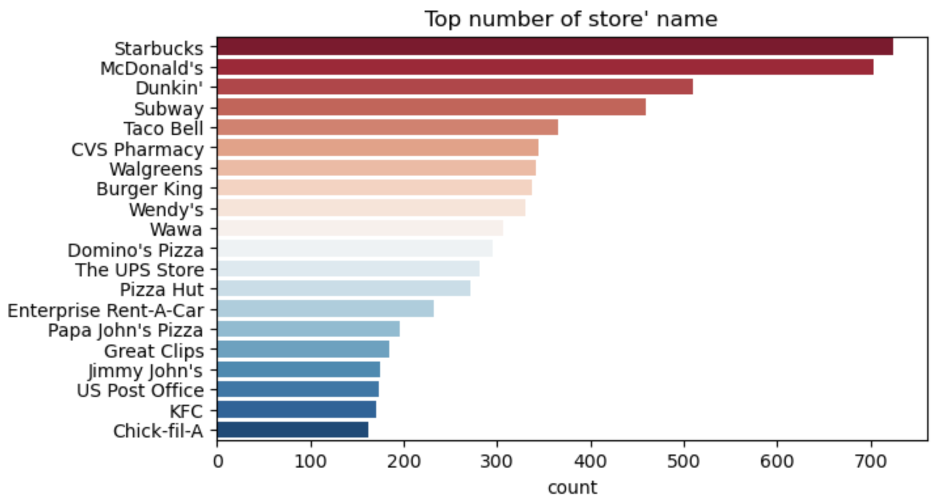




2) WordCloud to find the keywords of the business. We can find keywords like Cafe and Restaurant and more common in the name of those businesses.

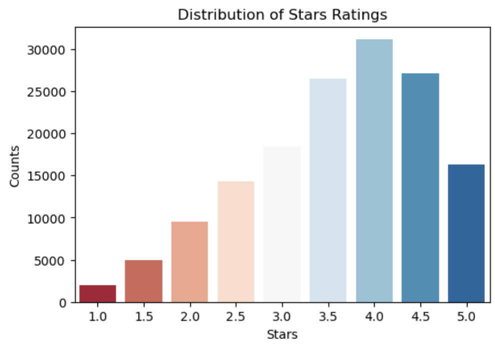


3) Count store name. We counted down the most popular business names. Chain business such as "McDonald's" and "Starbucks" is the most frequent.

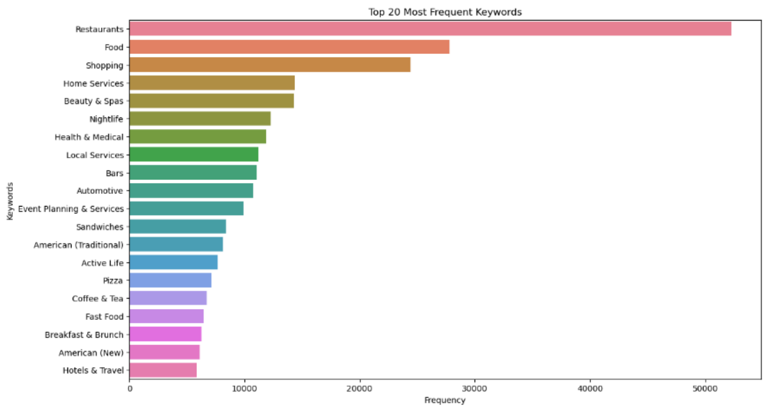


4) We also counted the STAR distribution of all businesses and found that most of the ratings are concentrated in the 3.5 to 4.5 range. We also counted the distribution

of STAR for all businesses and found that most of the ratings were concentrated in the 3.5 to 4.5 range, while the mean and median number of REVIEWS for businesses were 44.86 and 15, respectively.



5) Finally, we also counted the number of times each keyword appeared in the category (since each category had multiple values).



3. Data Preprocessing and Generation

1) Crawling data

We found that the review section of the original open-source Yelp data has the latest section as of 2022-01-19. However, the development of the store may have more changes in the post-epidemic era, so to understand the latest situation of the store, we still need to have the latest store review data which is most affected by time. Meanwhile, the store name and address of BUSINESS in the open-source data are less subject to time changes, so it was decided to get its latest reviews through the BUSINESS data.

Explore(1): Using Yelp official data acquisition API: yelp fusion API

- Advantages: easy to use, get fast, after applying for the official developer ide ntity to get the Yelp developer API_KEY can request JSON format data, eas y to handle.
- Disadvantages:
  (1) If the API is queried too quickly, the JSON body may receive an HTTP 4 29 error:
  (2)Daily Rate Limiting: In addition to QPS rate limiting, there is also a limit to the number of Fusion API requests that can be made per day. By default, a

client is limited to 500 API calls per 24 hours, resetting every midnight UT C.

```
{
    "error": {
        "code": "TOO_MANY_REQUESTS_PER_SECOND",
        "description": "You have exceeded the queries-per-
second limit for this endpoint.
                        Try reducing the rate at which you make queries."
    }
}
```

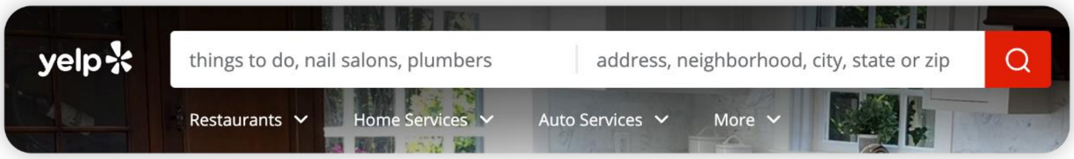
(3) More importantly, there are only three random reviews of the store return ed using the API, which is far from meeting our expectations

Explore (2):

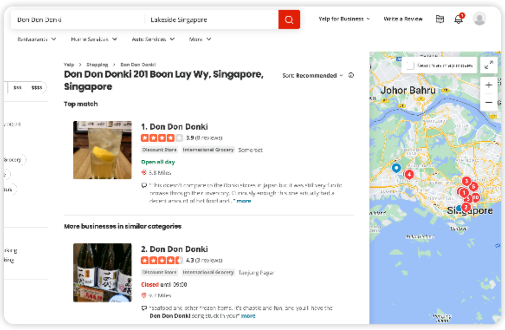
Therefore, to get more and more flexible review data to support our query an d get more conclusions, we use web crawlers.

Problem: [Yelp] is a nonstatic web page rendered through JavaScript, so it is ot possible to obtain data through static means, using [selenium] to achieve a simpl e action of manual operation, and then using XPath tags to locate the data.

Idea: simulate the user's web browsing process


yelp
things to do, nail salons, plumbers
address, neighborhood, city, state or zip
Restaurants
Home Services
Auto Services
More

1. Login to search for the name and location of the store.
2. Jump to the search results page and select the most relevant merchant to enter.
3. See relevant comment information



Yelp > Shopping > Don Don Donki

Don Don Donki 201 Boon Lay Wy, Singapore, Singapore

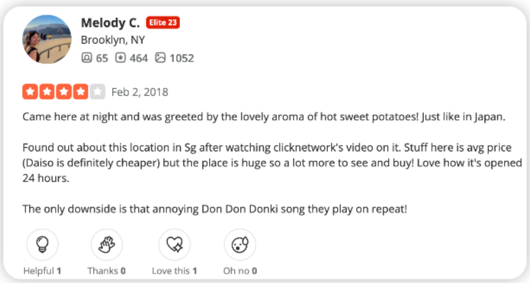
Sort: Recommended

Top match

1. Don Don Donki
3.9 (8 reviews)
Discount Store | International Grocery | Somerset
Open all day
5.8 Miles
"This doesn't compare to the Donki stores in Japan but it was still very fun to browse through their inventory. Curiously enough this one actually had a decent amount of hot food and..." more

More businesses in similar categories

2. Don Don Donki
4.3 (3 reviews)
Discount Store | International Grocery | Tanjong Pagar
Closed until 09:00
9.7 Miles
"seafood and other frozen items. It's chaotic and fun, and you'll have the Don Don Donki song stuck in your" more



Melody C. Elite 23
Brooklyn, NY
65 464 1052

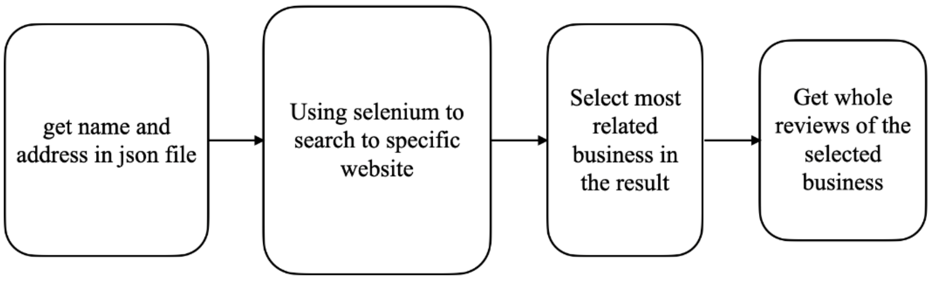
Feb 2, 2018

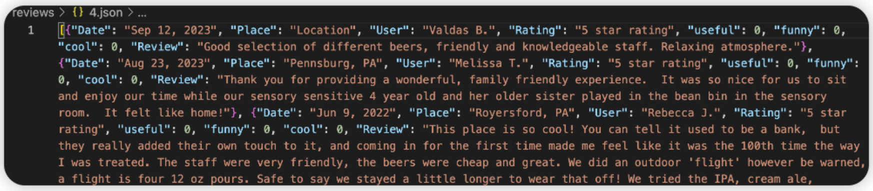
Came here at night and was greeted by the lovely aroma of hot sweet potatoes! Just like in Japan.

Found out about this location in Sg after watching clicknetwork's video on it. Stuff here is avg price (Daiso is definitely cheaper) but the place is huge so a lot more to see and buy! Love how it's opened 24 hours.

The only downside is that annoying Don Don Donki song they play on repeat!

Helpful 1 Thanks 0 Love this 1 Oh no 0

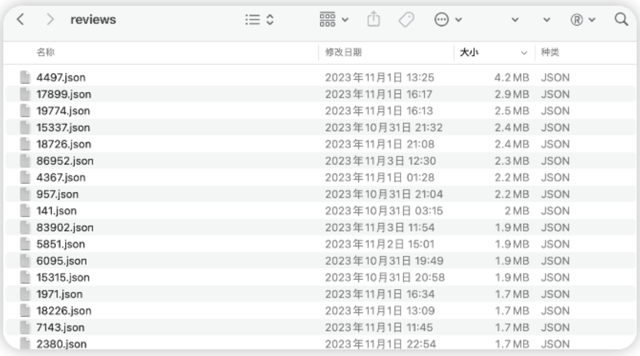

get name and address in json file
Using selenium to search to specific website
Select most related business in the result
Get whole reviews of the selected business


reviews > {} 4.json > ...

```
[{"Date": "Sep 12, 2023", "Place": "Location", "User": "Valdas B.", "Rating": "5 star rating", "useful": 0, "funny": 0,
"cool": 0, "Review": "Good selection of different beers, friendly and knowledgeable staff. Relaxing atmosphere."},
{"Date": "Aug 23, 2023", "Place": "Pennsburg, PA", "User": "Melissa T.", "Rating": "5 star rating", "useful": 0, "funny":
0, "cool": 0, "Review": "Thank you for providing a wonderful, family friendly experience.  It was so nice for us to sit
and enjoy our time while our sensory sensitive 4 year old and her older sister played in the bean bin in the sensory
room.  It felt like home!"}, {"Date": "Jun 9, 2022", "Place": "Royersford, PA", "User": "Rebecca J.", "Rating": "5 star
rating", "useful": 0, "funny": 0, "cool": 0, "Review": "This place is so cool! You can tell it used to be a bank,  but
they really added their own touch to it, and coming in for the first time made me feel like it was the 100th time the way
I was treated. The staff were very friendly, the beers were cheap and great. We did an outdoor 'flight' however be warned,
a flight is four 12 oz pours. Safe to say we stayed a little longer to wear that off! We tried the IPA, cream ale,
```

Similarly, through the open source data business.json a total of 150347 rows, for each line is to use his name and address information to search, and then through the store for each line to get all the reviews and saved in the index named json fileThis includes the review date, the user who made the review, and the rating and review text.

After completing the crawling process, all business reviews will be stored in the form of index.json in the folder of reviews, (due to the limitations of the project cycle) a total of 16905 business reviews are crawled. The data structure is shown above, Date:(Month, Day, Year); Place: User(name), Rating:""; useful, funny, cool, Review(). It can be found that compared with the open source data review is still missing some information, the subsequent crawling data processing will be reflected in the Data processing.


reviews

名称	修改日期	大小	种类
4497.json	2023年11月1日 13:25	4.2 MB	JSON
17899.json	2023年11月1日 16:17	2.9 MB	JSON
19774.json	2023年11月1日 16:13	2.5 MB	JSON
15337.json	2023年10月31日 21:32	2.4 MB	JSON
18726.json	2023年11月1日 21:08	2.4 MB	JSON
86952.json	2023年11月3日 12:30	2.3 MB	JSON
4367.json	2023年11月1日 01:28	2.2 MB	JSON
957.json	2023年10月31日 21:04	2.2 MB	JSON
141.json	2023年10月31日 03:15	2 MB	JSON
83902.json	2023年11月3日 11:54	1.9 MB	JSON
5851.json	2023年11月2日 15:01	1.9 MB	JSON
6095.json	2023年10月31日 19:49	1.9 MB	JSON
15315.json	2023年10月31日 20:58	1.9 MB	JSON
1971.json	2023年11月1日 16:34	1.7 MB	JSON
18226.json	2023年11月1日 13:09	1.7 MB	JSON
7143.json	2023年11月1日 11:45	1.7 MB	JSON
2380.json	2023年11月1日 22:54	1.7 MB	JSON

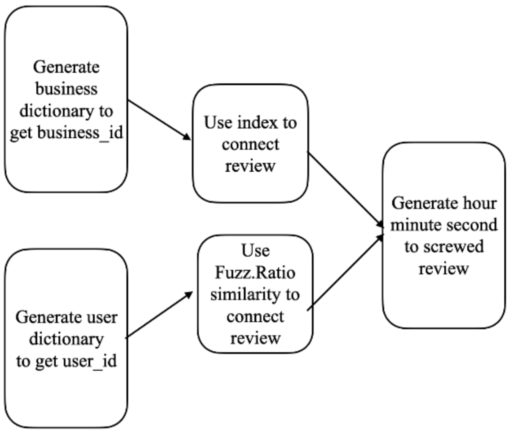
2) Data cleaning and preprocessing

Since the original Yelp dataset is in JSON format, to simplify our data cleaning process and keep the data stored in MySQL and MongoDB consistent, we first adjust the data to the format that can be directly imported into MongoDB, then export the data to the new JSON files, and finally process the data to the same structure as the schema diagram of MySQL.

There are several problems in the original data we need to handle before importing it into MongoDB. First, the values of 'categories' in 'business.json', the values of 'date' in 'checkin.json', and the values of 'elite' and 'friends' in 'user.json' are currently strings separated by commas, which cannot be recognized as arrays in MongoDB. Second, the embedded documents in 'attributes' in 'business.json' are enclosed in quotation marks. Third, the 'checkin.json' is now separated from 'business.json'. We handle these three problems by loading each file in Python and adjust them to the right format. Besides, since we crawled some new review data

from Yelp.com, we dropped the original 'review_id' in 'review' and let MongoDB generate a unique '_id' as the new 'review_id' for us. But for 'business' and 'user', we want MongoDB to use the existing 'business_id' and 'user_id' as '_id' to better link these two entities to the relationships, we rename 'business_id' in business' and 'user_id' in 'user' to '_id'. Then, we import these data into MongoDB. The Python codes used here are attached as 'Data_Cleaning_NonRelational.ipynb'.
In MongoDB, we observed that the data which should have been of datetime type, are now recognized as string. We apply '$toDate' method in MongoDB to solve this problem. We also observed by queries that some 'user_id' in the 'review' collection do not appear in the 'user' collection. We consider them as incorrect data and delete them.

Now, we have completed the data cleaning and preprocessing process for data used in MongoDB. We export some of the processed data to a csv file, including 'review', 'tip', 'visit' and use '_id' generated by MongoDB as their primary key when imported into MySQL. For tables 'business', 'business_category', 'user', and 'user_compliment', we project the data to the corresponding fields and export them to csv file directly from MongoDB. Finally, we generate the rest tables from the json files exported from MongoDB using Python. The Python codes used here are attached as 'Data_Cleaning_Relational.ipynb'.
For the crawler data preprocessing as well as cleaning. Such as:
Date": "Jul 6, 2019", "Place": "Pottstown, PA", "User": "Stephanie S.", "Rating": "5 star rating", "useful": "1", "funny": 0, "cool": 0, “Review”: “xxxxx”


Generate business dictionary to get business_id
Use index to connect review
Generate hour minute second to screwed review
Generate user dictionary to get user_id
Use Fuzz.Ratio similarity to connect review

By comparing the data differences in the open source review, we found that there is a lack of business_id of the review (which can be obtained by going to the open source business through the index of this JSON file to get the business_id of the corresponding line), user_id (but the user data may change after the open source, such as the name and location, and it is not possible to exact match, and the exact match has the suspicion of leaking user information, so the choice of similarity matching, generated for display only and exists in the open source user file user_id), and comments on the date of the lack of the corresponding hours, minutes and seconds.

3) Data Generation Visit Table

To more realistically describe what happened cut without stealing user privacy, we decided to generate a visit form based on a fully processed review data form indicating when a user visited a merchant. The time of the user's review is randomly reduced by 30 minutes-120 minutes as the time of the user's visit to the store.

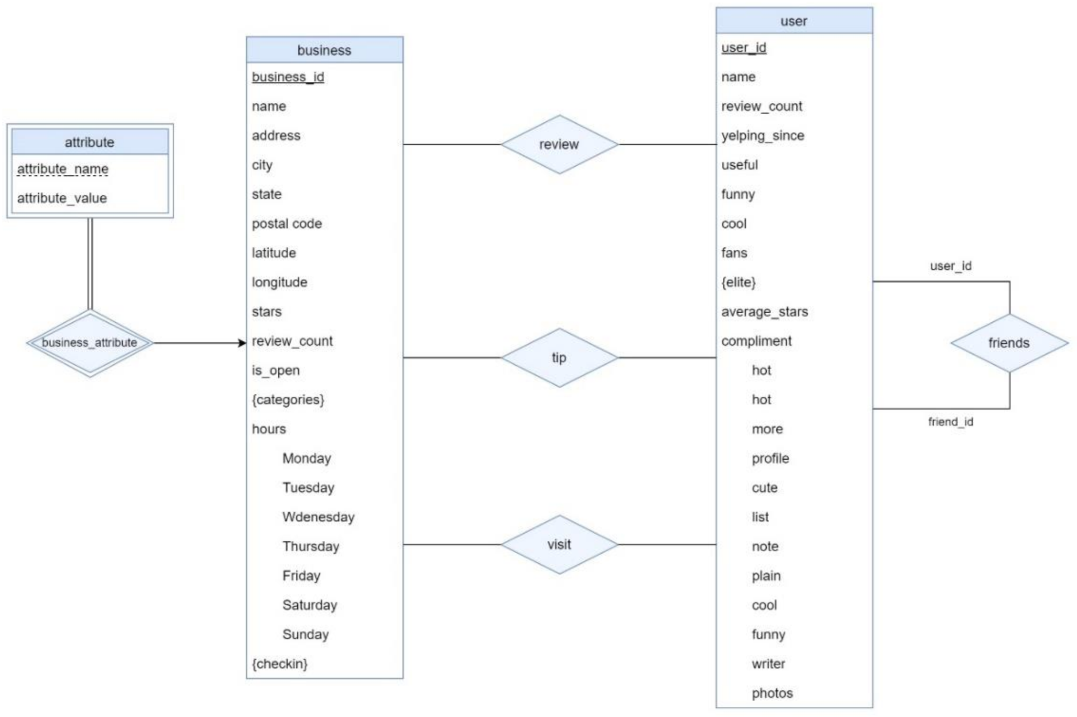
	time_visit	**user_id**	**business_id**
0	2022-05-20 06:22:10	GARCbCdZzIm6aixVEJboTA	UJsufbvfyfONHeWdvAHKjA

4. Data Model

1) Relational Database

a) ER diagram and analysis

The ER diagram is the associated diagrammatic representation of the ER model, which employs three basic concepts: entity sets, relationship sets, and attributes. ER diagram we designed is shown in the figure below.



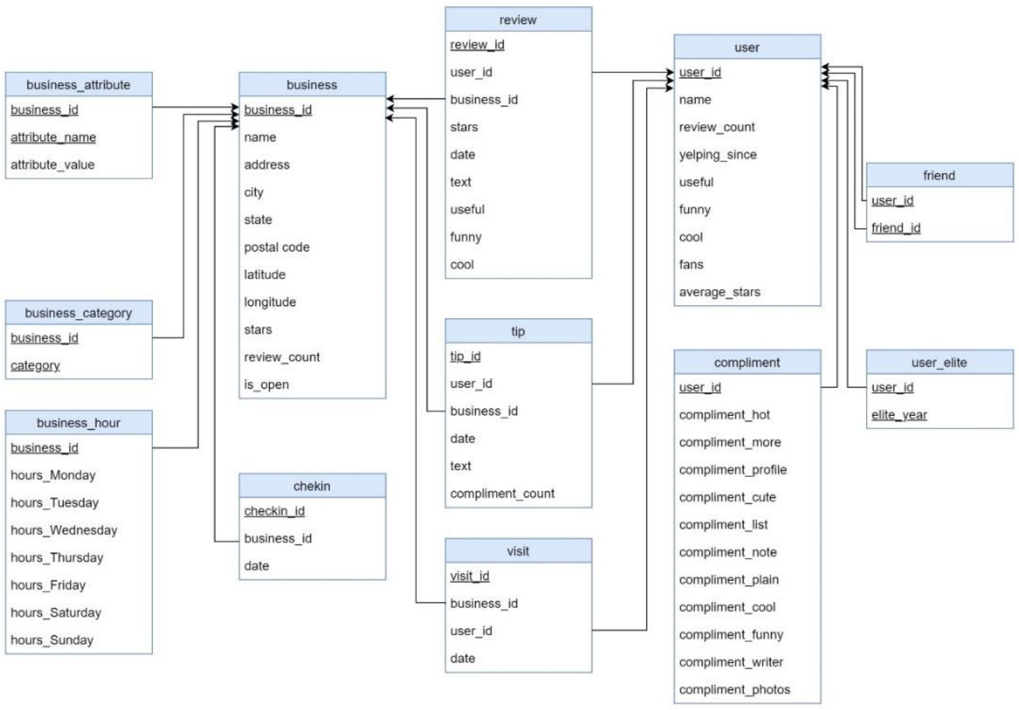
In our ER diagram, there are three entity sets and four relationship sets. The first entity set is 'business', which represents the merchants on the Yelp website. The second entity set is 'user', representing the users of the Yelp platform. These two entity sets are the main entity sets in our model, attached with many attributes to express their characteristics. Among them, 'categories', 'checkin' in 'business', and 'elite' in user are multivalued attributes. 'hours' in 'business' is a composite attribute, which indicates the operating hours of the same store from Monday to Sunday, while 'compliment' in 'user' is also a composite attribute. In addition, there is a weak entity set 'attribute', which is dependent on 'business' through relationship 'business_attribute', indicating the facility availability of entities in 'business'.

There are three relationship sets 'review', 'tip' and 'visit' connecting 'user' and 'business'. They respectively indicate that a user has written a review for a business, given a tip to a business and visited a business. These are the main relationship in our ER model. Besides, the relationship set 'friends' represents the interpersonal relationship between different users.

As for mapping cardinalities, all relationships that appear in the ER diagram are many-to-many relationships, except the identifying relationship between 'attribute' and 'business'.

b) Schema diagram and analysis

The schema we designed The diagram is shown below:



To obtain the schema diagram, we did the following work.

i. The weak entity set 'attribute' becomes table 'business_attribute' that includes the primary key of the identifying strong entity set 'business' and the original attributes of 'attribute'.

ii. For multivalued attribute M of entity E, we construct a separate table EM to represent it. In the ER diagram, 'categories', 'checkin_date' in 'business' and the 'elite' in 'user' become tables 'business_category', 'checkin' and 'user_elite' in the schema diagram.

iii. Composite attributes of entity sets are flattened by creating a separate attribute for each component attribute. Here composite attributes 'hours' in 'business' and 'compliment' in 'user' are flattened. But after that, 'business' table and 'user' table contain too many fields, which may reduce the efficiency of queries that do not query for 'hours' and 'compliment'.

Considering most queries do not query for 'hours' and 'compliment', we separate 'hours' from 'business' and construct a new table 'business_hour', and separate 'compliment' from 'user' and construct a new table 'compliment'.

iv. Convert many-to-many relationship set by adding the primary keys of both sides of the relationship. By doing so, we get four tables 'review', 'tip', 'visit' and 'friend'.

c) Schema Refinement

In the schema refinement process, we hope that all the tables in our shcema diagram satisfy BCNF, which means that for every relation R, the only non-trivial FDs on R are key constraints. After checking the FDs on every relations in our schema diagram, we confirm that every relation in the schema already satisfies BCNF.

One thing that may cause confusion is that among the attributes of the 'business' entity, some attributes actually represent geographical location information, including 'address', 'city', 'state', 'postal_code', 'latitude', and 'longitude'. Some people may think that these attributes violate BCNF. This was something we worried about when designing the database. But actually, they do not violate BCNF. We found that there is no mutual determination relationship between these geographical data. For example, the city cannot determine the state because in the United Sates, different states can have cities of the same name. Also, given the latitude and longitude of a place, there's still some cases that we cannot determine the city and state of the place, when the place is located at the junctions of cities. Considering these cases, we believe that there are no FDs between these geographical attributes in 'business'. Therefore, the relation 'business' is already in BCNF.

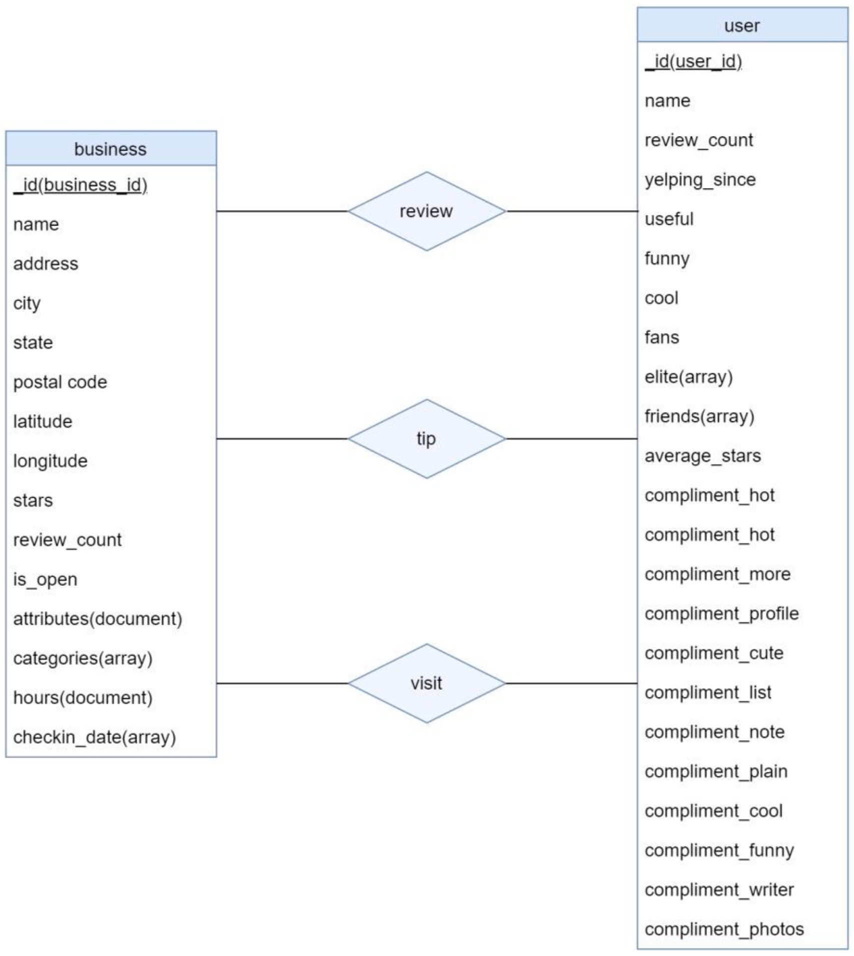
2) **Non-Relational Database**

a) Why MongoDB

MongoDB is a document-based NoSQL database, which means that it does not require a predefined schema like MySQL. Its flexibility allows us to store complex types of data in one collection. Take the Yelp dataset as an example. We can directly embed 'attributes' into 'business' collection, instead of building a new table to store these data. This method makes database design and management much easier in some cases, and also improves the efficiency of queries that require a join operation in MySQL due to separation of data. Besides, MongoDB has built-in support for geospatial indexing and queries, making it a good choice for applications that involve location-based data.

b) Schema design

For the Yelp dataset, we have two main entity sets 'business' and 'user', along with three many-to-many relationship sets 'review', 'tip', 'visit' connecting 'business' and 'user', and a recursive many-to-many relationship 'friend' of 'user'. Since the relationship 'friend' does not contain any extra attributes, we can simply store it as an array in the 'user' collection.



The rest three relationships between 'business' and 'user' all contain extra attributes. We have two ways to store them. One way is embedding the relationship set in one side of the relation. The advantage of this method is higher efficiency for read operations, while embedded array may grow too large and hit the size limit of MongoDB. Additionally, queries accessing the relationship from the other side of the relationship that does not store the embedded relationship data may cost much longer time. The other way is creating a separate collection to store the relationship between entities, which is similar to the representation of many-to-many relationships in relational database. This method prevents the document size from growing above the limit, and is more flexible for data retrieving from both sides of the relationship, but linking collections in MongoDB may involve more complex queries and reduce the efficiency of some queries.

Combining the advantages and disadvantages of both, we decide to use the latter approach to store these three relationship sets. The reason for this choice is that the 'review' and 'visit' relationship sets have large volume of data, and that we may access the three relationships from both sides of the relationship.

c) Analysis

In MongoDB, we end up with five collections 'business', 'user', 'review', 'tip', and 'visit'. We store the check-in and category data of the 'business' entity as arrays 'checkin_date' and 'categories' in 'business' collection, and we keep 'attributes' and 'hours' as the same embedded form in the original Yelp dataset. Also, we store the friend and year of elite data of entity 'user' as arrays 'friends' and 'elite' in the 'user' collection. For the three collections converted from the relationship set, we keep them as the same format in MySQL.

The advantage of our schema design for MongoDB is that for queries acquiring data stored as embedded attributes ('business.attributes', 'business. categories', 'business.checkin_date', 'user.elite', 'user.friends'), the time consumption is significantly reduced.

5. System design

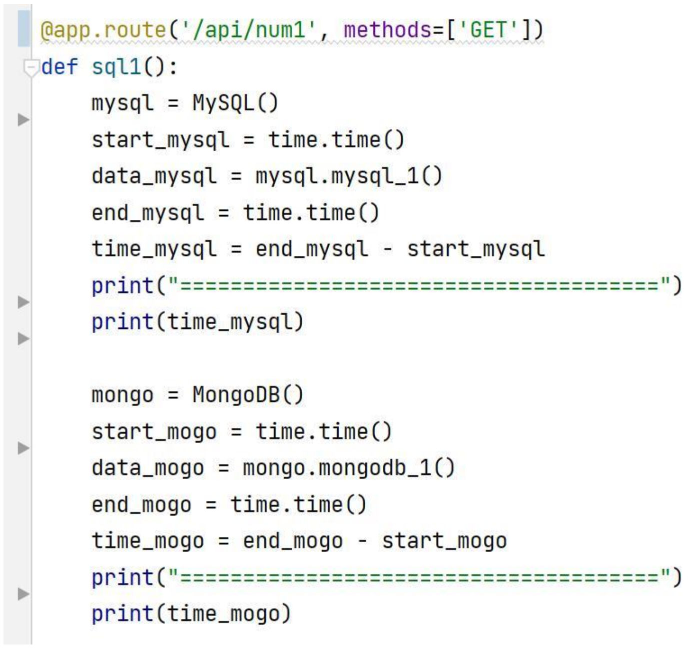
1) Back-end

The backend of this project uses flask framework. Flask is a lightweight web application framework that is more flexible and lightweight to use than other frameworks of its type. This project connects MySQL and MongoDB database through the back-end to query data. And the query data is returned to the front-end through the back-end, so that the front-end can display the data.

The backend is divided into 4 files:

app.py

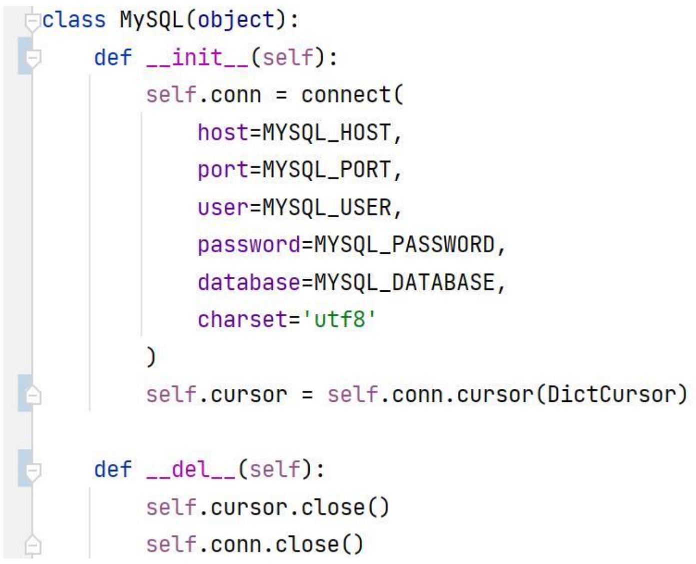
This file is the main file for the backend application. This file is used by the f lask framework to query mysql.py and MongoDB.py and return the results to the front-end.



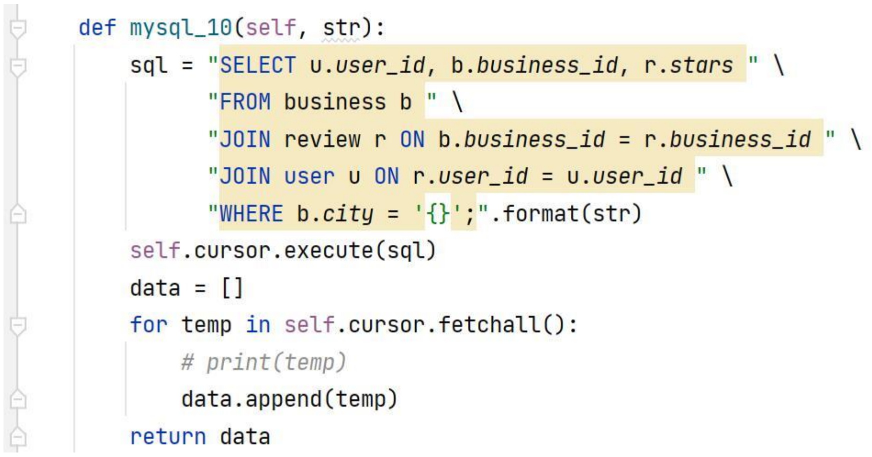
```
@app.route('/api/num1', methods=['GET'])
def sql1():
    mysql = MySQL()
    start_mysql = time.time()
    data_mysql = mysql.mysql_1()
    end_mysql = time.time()
    time_mysql = end_mysql - start_mysql
    print("======================================")
    print(time_mysql)

    mongo = MongoDB()
    start_mogo = time.time()
    data_mogo = mongo.mongodb_1()
    end_mogo = time.time()
    time_mogo = end_mogo - start_mogo
    print("======================================")
    print(time_mogo)
```

mysql.py

This file is used to connect to the MySQL database, and to query the MySQL database by implementing different methods.



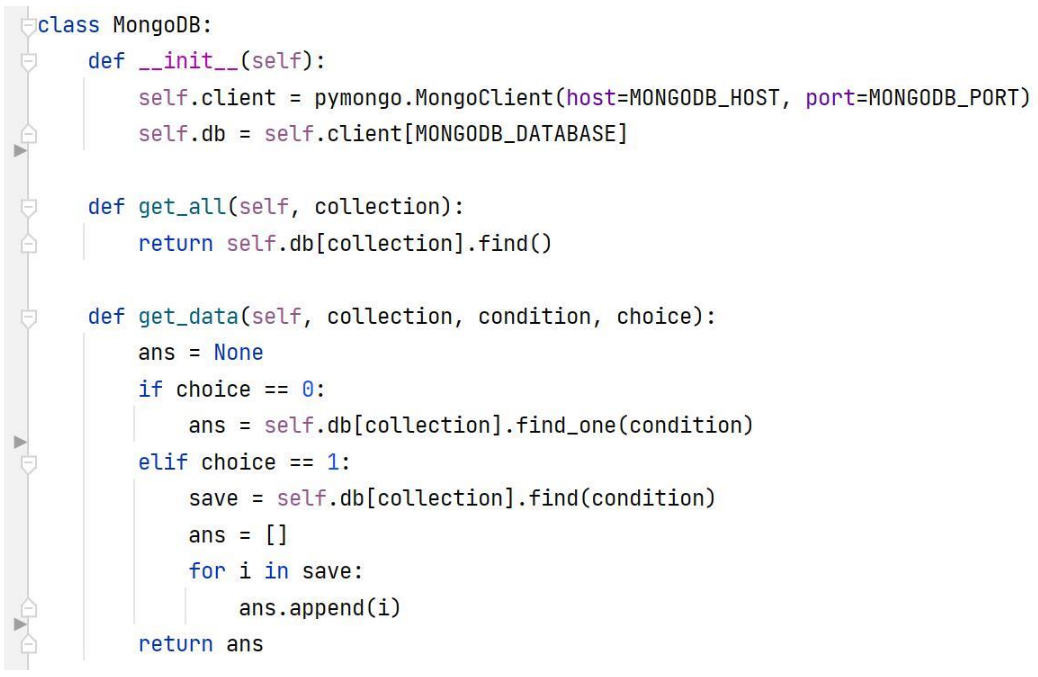
```
class MySQL(object):
    def __init__(self):
        self.conn = connect(
            host=MYSQL_HOST,
            port=MYSQL_PORT,
            user=MYSQL_USER,
            password=MYSQL_PASSWORD,
            database=MYSQL_DATABASE,
            charset='utf8'
        )
        self.cursor = self.conn.cursor(DictCursor)

    def __del__(self):
        self.cursor.close()
        self.conn.close()
```



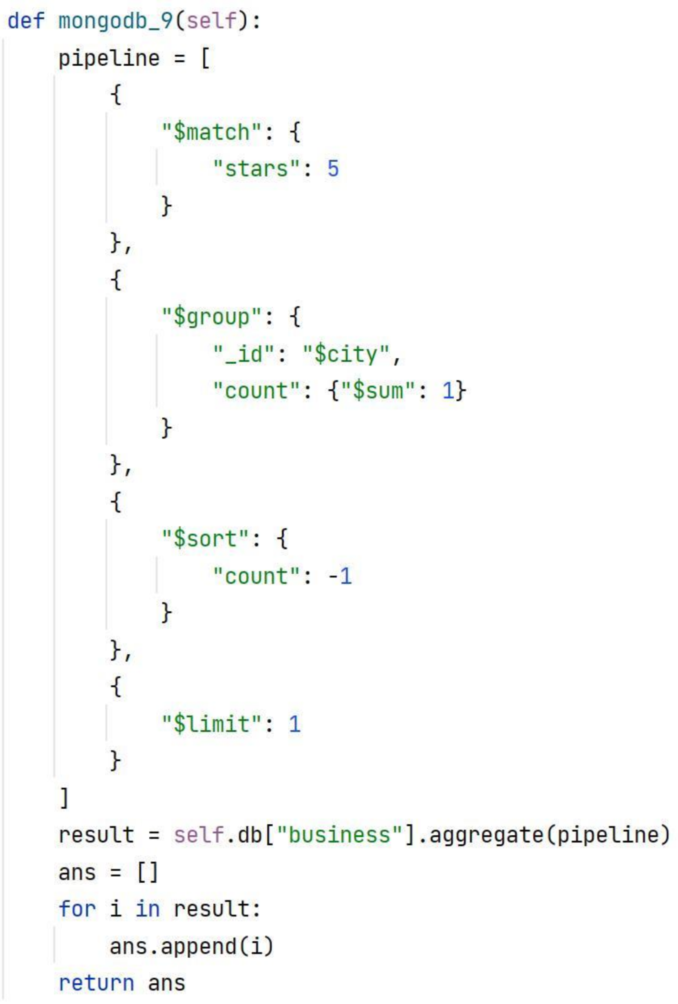
```
def mysql_10(self, str):
    sql = "SELECT u.user_id, b.business_id, r.stars " \
          "FROM business b " \
          "JOIN review r ON b.business_id = r.business_id " \
          "JOIN user u ON r.user_id = u.user_id " \
          "WHERE b.city = '{}';".format(str)
    self.cursor.execute(sql)
    data = []
    for temp in self.cursor.fetchall():
        # print(temp)
        data.append(temp)
    return data
```

MongoDB.py

This file is used to connect to the MongoDB database and implement different methods to query the MongoDB database.



```python
class MongoDB:
    def __init__(self):
        self.client = pymongo.MongoClient(host=MONGODB_HOST, port=MONGODB_PORT)
        self.db = self.client[MONGODB_DATABASE]

    def get_all(self, collection):
        return self.db[collection].find()

    def get_data(self, collection, condition, choice):
        ans = None
        if choice == 0:
            ans = self.db[collection].find_one(condition)
        elif choice == 1:
            save = self.db[collection].find(condition)
            ans = []
            for i in save:
                ans.append(i)
        return ans
```



```python
def mongodb_9(self):
    pipeline = [
        {
            "$match": {
                "stars": 5
            }
        },
        {
            "$group": {
                "_id": "$city",
                "count": {"$sum": 1}
            }
        },
        {
            "$sort": {
                "count": -1
            }
        },
        {
            "$limit": 1
        }
    ]
    result = self.db["business"].aggregate(pipeline)
    ans = []
    for i in result:
        ans.append(i)
    return ans
```

settings.py

This file is used to store the port number, user, password, database name and other information used to connect to the MySQL database and MongoDB databa se.

2) **Front-end**

The front-end portion utilizes the Vue.js framework and aims to create an interactive user interface that allows users to efficiently access, display, and manipulate data stored in two major database systems, MySQL and MongoDB. The front-end implements dynamic display and interactive processing of data, including but not limited to data rendering, viewing of different datasets through paging controls, and user-triggered page routing switches.

For technology selection, the front-end system integrates the following key technologies and frameworks:

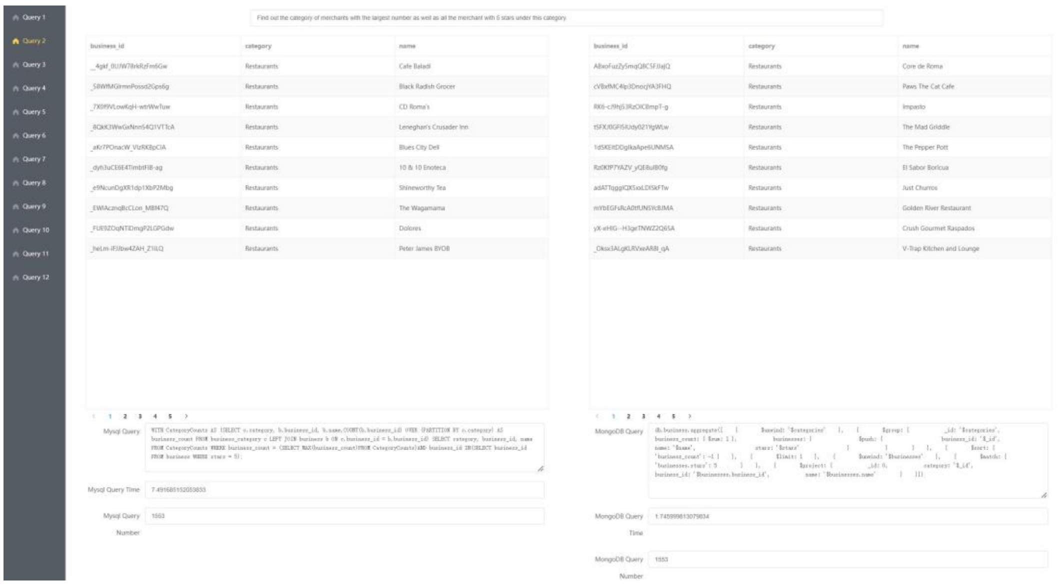
● Vue.js: progressive JavaScript framework for building efficient, dynamic user interfaces.

● Vue Router: the official routing library for managing page jumps in single-page applications.

● Vuex: state management architecture and library for Vue.js applications.

● Element UI: a desktop component library based on Vue 2.0 that provides developers with a full set of designed components and elements including datagrids and pagers.

● Axios: a Promise-based HTTP client for initiating requests and handling responses.



Find out the category of merchants with the largest number as well as all the merchant with 5 stars under this category

business_id	category	name
__4gkf_0UJW78rkRzFm6Gw	Restaurants	Cafe Baladi
_58WfMGlrmnPossd2Gps6g	Restaurants	Black Radish Grocer
_7X0f9VLowKqH-wtrWwTuw	Restaurants	CD Roma's
_8QkK3WwGxNnn54Q1VTTcA	Restaurants	Leneghan's Crusader Inn
_aKr7POnacW_VlzRKBpClA	Restaurants	Blues City Deli
_dyh3uCE6E4TimbtFIB-ag	Restaurants	10 & 10 Enoteca
_e9NcunDgXR1dp1XbP2Mbg	Restaurants	Shineworthy Tea
_EWIAczmqBcCLon_MBI47Q	Restaurants	The Wagamama
_FUE9ZOqNTlDmgP2LGPGdw	Restaurants	Dolores
_heLm-iFJJbw4ZAH_Z1ILQ	Restaurants	Peter James BYOB

Mysql Query
```
WITH CategoryCounts AS (SELECT c.category, b.business_id, b.name,COUNT(b.business_id) OVER (PARTITION BY c.category) AS
business_count FROM business_category c LEFT JOIN business b ON c.business_id = b.business_id) SELECT category, business_id, name
FROM CategoryCounts WHERE business_count = (SELECT MAX(business_count)FROM CategoryCounts)AND business_id IN(SELECT business_id
FROM business WHERE stars = 5);
```

Mysql Query Time: 7.491685152053833

Mysql Query Number: 1553

business_id	category	name
ABxoFuzZy5mqQ8C5FJJajQ	Restaurants	Core de Roma
cVBxfMC4lp3DnocjYA3FHQ	Restaurants	Paws The Cat Cafe
RK6-cJ9hjS3RzOICBmpT-g	Restaurants	Impasto
tSFXJ0GFI5IUdy021YgWLw	Restaurants	The Mad Griddle
1dSKEltDOgIkaApe6UNMSA	Restaurants	The Pepper Pott
Rz0KfP7YAZV_yQE8uIB0fg	Restaurants	El Sabor Boricua
adATTqggIQX5ioLDISkFTw	Restaurants	Just Churros
mYbEGFsRcA0tfUN5YcBJMA	Restaurants	Golden River Restaurant
yX-eHIG--H3geTNWZ2Q65A	Restaurants	Crush Gourmet Raspados
_Oksx3ALgKLRVxeAR8I_qA	Restaurants	V-Trap Kitchen and Lounge

MongoDB Query
```
db.business.aggregate([    {        $unwind: '$categories'    },    {        $group: {            _id: '$categories',
business_count: { $sum: 1 },            businesses: {                $push: {                    business_id: '$_id',
name: '$name',                    stars: '$stars'                }            }        }    },    {        $sort: {
'business_count': -1 }    },    {        $limit: 1    },    {        $unwind: '$businesses'    },    {        $match: {
'businesses.stars': 5        }    },    {        $project: {            _id: 0,            category: '$_id',
business_id: '$businesses.business_id',            name: '$businesses.name'        }    }]);
```

MongoDB Query Time: 1.745999813079834

MongoDB Query Number: 1553

In terms of technical architecture, the front-end deploys a modular and componentized development approach, where each individual component is responsible for a specific function of the interface, such as navigation bar, table display, and data paging. This structure not only promotes code reusability and maintainability, but also simplifies the process of potential feature extensions or updates.The entire front-end system delivers a smooth and engaging user experience through a consistent interface provided by the Element UI component library, combined with the responsive and componentized nature of Vue.js. It also ensures maintainability and extensibility and improves development efficiency.

6. Queries and Analysis

We designed a total of 12 queries. They are categorized into daily queries and analytical queries. These 12 well-designed queries are intended to reflect the differences between MySQL and MongoDB as well as the advantages of our model of data processing.

1) Daily Query

- **Query:** Select those merchants with an average rating higher than 4 star from 2023-01-01 to now in 'Santa Barbara'.

  Explanation: This query will help you find the most popular businesses in the near future. In addition, the query was able to prove that the project crawled the most recent data from 2023. For this query, although both MySQL and MongoDB perform multi-table queries, MongoDB is less efficient for multi-table joins. As a result, MySQL queries are faster.
- **Query:** Count the average star ratings for each category and the number of merchants (still open doors) in each category in each city and select the category with the highest number of merchants and average star ratings in the city.

  Explanation: This query wants to explore the underlying principles of MongoDB and the most basic table joins. After testing, it was found that the MySQL query was one second slower than the MongoDB query. This is because this query involves a complex multi-table query as it needs to use multiple attributes for aggregate sorting and then aggregate sorting. This is clearly detrimental to MySQL, whereas in MongoDB, thanks to our collection design structure, there is no need for multi-table joins because data is usually stored in embedded form. So, for relatively simple aggregated queries, MongoDB's aggregation pipeline is more advantageous. But there is no doubt that MySQL's query structure is more concise and easy to understand.
- **Query:** Use MongoDB's geospatial indexing capabilities to find the highest-rated merchants within 5 kilometers of Tucson's downtown location. (Latitude: 32.25346° N Longitude: 110.91179° W).

  Explanation: This query concerns the geographic coordinate system. On Earth, a coordinate can be accurately identified using longitude and latitude. MongoDB supports geographic coordinates "2dsphere". Therefore, a geographic coordinates index is created on MongoDB. When searching for businesses within 50km of the center of Tucson, the indexed query was several times faster. Compared to the calculation on MySQL, we found that the time difference between the two is not significant. However, since MySQL uses floating point numbers, the number of businesses calculated by MySQL is more accurate than that calculated by MongoDB using $geoWithin and $centerSphere. And MySQL calculation is more flexible, easy to combine with certain work.

2) Analyze Query

- **Query:** Providing User-Item Matrix for Collaborative Filtering Algorithms.
  Explanation: Using the Yelp dataset, simple recommendation algorithms are implemented with query statements. In collaborative filtering algorithms, the User-Item matrix is important, and this query statement is designed to detect the speed of querying the User-Item matrix from the database. Although this query involves joining only three tables, MySQL is significantly faster than MongoDB. Because MongoDB's $lookup operation joins less efficiently than SQL's JOIN on large datasets, and the User-Item matrix does not require the data structure to be scalable, MongoDB has no advantage.
- **Query:** The "Influence Score" is generated from the sum of the 11 attributes of the user's COMPLIMENT plus 100 times the user's ELITE number of years, listing the 20 users with the highest scores.
  Explanation: This query is dedicated to demonstrating the benefits of the MongoDB array structure. Complement-related attributes are listed as a separate table in relational databases, whereas in non-relational databases, a compliment is just an array corresponding to a field in a document. Also, the elite information of users are listed as a separate table in MySQL while an array inside 'user' in MongoDB. Therefore, MongoDB does not need to link the two tables. As a result, there is a significant speedup in operation. Moreover, using aggregate pipelines, especially $addFields and $project, allows for more flexible handling of data.
- **Query:** Find all the businesses that had more check-in numbers in 2020 than in 2019.
  Explanation: This query involves the handling of table links and time attributes. Although the time attribute is handled as a Date format in both MySQL and MongoDB, MongoDB stores it as an array under business, where each element of the array is a signed-in time. In this query, joins and subqueries to multiple tables are required, especially when dealing with time-series data, which can lead to a lot of disk I/O operations and computations. As a non-relational database, MongoDB allows for a more flexible document model. Data can often be stored in a non-normalized form, reducing the number of data join operations that need to be performed. For such queries, MongoDB can utilize its efficient aggregation pipeline, reducing complex join and subquery operations. After observing that check-in times are unevenly

distributed in MySQL, query optimization may be more difficult for such sparse time-series data.

- **Query:** Find the name of merchants in NV state with loyal customers (visited by at least 1 customer 5 times or more).
  Explanation: This query can show the advantage of our generated data, if you want to find out in the original data: 1 customer visits 5 times or more such cases, it is quite difficult. Therefore, we generated the visit table to store the relevant information. As you can see, this table makes its query much faster than MongoDB.
- **Query:** Find out the category of merchants with the largest number as well as all the merchants with 5 stars under this category..
  Explanation: It is designed to handle both nested data and joining different tables as a way to distinguish between the two databases. MySQL's use of CTEs and window functions may make query statements more complex, and “With as” illustrates that SQL is difficult to write under this query. MongoDB, on the other hand, is very easy to write. Although $lookup provides SQL-like join functionality, it may not be as efficient as JOIN in SQL in some complex data association scenarios.
- **Query:** Search review data to find all reviews that mention "excellent service" and display essential details about the merchant.
  Explanation: This query enables the user to query the reviews and quickly find valuable reviews for reference. In this query, MongoDB doesn't involve multiple tables and can get results from a collection, whereas MySQL needs to join two tables, business and review. As a result, MongoDB queries are more efficient. It's worth noting that, in general, MySQL is case-insensitive for string matching, while MongoDB is case-sensitive. Therefore, in this problem, the binary keyword needs to be set in the MySQL database to make the MySQL database sensitive to letter cases.
- **Query:** Analyze which city has the most 5-star merchants and return the number.
  Explanation: This query allows users to quickly know which city has the most five-star businesses, which can be used to judge the service development level of different cities. This query is very fast in both MySQL and MongoDB, taking less than 0.3 seconds. From this, we can see that both relational and non-relational databases perform well when it comes to single-table queries.
- **Query:** Find out which stores in the city: Santa Barbara with a star rating greater than 3.5 have an average star rating greater than the merchant's star rating in reviews after January 1, 2020, at 0:00 pm.
  Explanation: This is a way to get the top-performing merchants in a given location in the recent past, as the merchant's situation may have changed after

the outbreak.

- **Query:** Find out who John's friends are. Find the people who go shopping more often than him.

  Explanation: This can understand someone's social circle, more than he loves to go to a merchant's information,. the recommender system has a certain positive significance, after the user logs in to the system, the system understands his information (that is, here John), and this collaborative filtering, to a certain extent, to facilitate the construction of the user's profile.

7. Summarize

MySQL and MongoDB are popular database management systems, each with unique strengths and application scenarios.MySQL is a relational database that excels at handling multi-table joins and complex transactions. This type of database is particularly efficient in applications that involve complex queries and transactions, such as financial services and enterprise systems.MySQL's data model is relatively fixed, which means that changes to the schema can be more complex, but it excels in handling situations that require strong data integrity and consistency. It provides strong consistency for those scenarios where you need to ensure data accuracy and consistency. In addition, MySQL shows strengths in handling time-series data, especially sparse data, such as being faster when querying user-item matrices in collaborative filtering and recommender systems.

In contrast, MongoDB is a non-relational, NoSQL database that uses JSON-like documents and has a dynamic schema, making it particularly well-suited for unstructured data.MongoDB excels at handling simple aggregated queries and unstructured data and is particularly efficient at queries that do not require multiple table joins. Its flexible document model is easy to adapt to data schemas and supports horizontal scaling, which makes MongoDB particularly well-suited for big data and cloud applications. Its document-oriented storage approach makes it suitable for the rapid development and storage of large amounts of non-relational data for applications such as the Internet of Things, big data analytics, and content management systems.MongoDB provides eventual consistency, making it suitable for applications that don't require much immediate consistency.

Overall, MySQL is more appropriate for traditional applications that require a high degree of consistency and complex relational data, while MongoDB is better suited for modern applications that require rapid iteration and processing of large amounts of unstructured data. These two database systems offer their own advantages and solutions based on different application requirements and scenarios.

8. Member Workload

1) Workload
   - Li Fangzheng:

According to the yelp website and open source existing data characteristics completing the design of the crawler, crawl more time dimension information, and crawl the data for data cleaning, participate in the writing of part of SQL queries. Generate rationalized new data according to the characteristics of existing data and improve the system query.

- Yang Enze:

Mainly involved in the construction of structured databases, including the design of ER diagrams and schema diagrams. Participate in data cleaning and preprocessing for MySQL. Participate in the design of sql and mongodb query statements. Complete the relevant project text.

- Zhang Hantian:

Perform data cleaning and preprocessing for MySQL and MongoDB. Participate in relational database deisgn, including constucting the ER model, converting ER model to schema diagram, and schema refinement. Conduct non-relational database design. Write part of the queries for MySQL and MongoDB. Complete the relevant part in the report.

- Zhang Yingdong:

Complete the back-end code construction, connect the back-end with MySQL database and MongoDB database, complete part of the SQL query, tests and statistics the time of all SQL queries, the number of result items, and other information.

- Zhao Wenkang:

Designed 12 queries for the entire project, completed the transformation of three queries. Participated in data crawling, provided data. Data visualization, visual analysis of data for statistics. Developed the frontend, provided data display, and tested the code. Organized reports and proposal slides.

Teamwork:

1. Ongoing communication
2. Helping each other solve problems
3. Regular discussions
4. clear division of labor but maintaining cooperation

2) Extra credit

Based on the above teamwork, we have also made some additional efforts.

- First, we completed data expansion by applying crawling techniques to crawl new data from Yelp's official website.
- Second, we customized the front end to display the data.

9. Data set and Codes

Github：https://github.com/zyd0131/DSA5104_yelp