Final Project

Project code

- **Github repo link:** https://github.com/Huadous/final-project
- **README:** This program uses the API provided by Yelp Fusion. The **API verification key** is very easy to apply, please apply through this link: https://www.yelp.com/fusion. Sign up for an account and manage an app to gain the API Key. Then, you need to create a file named `secrets.py` and copy this variable `API_KEY = 'your_api_key'` into the file to support the program.

  The entire program is built within the **Flask app**, and the interaction is relatively simple. The main thing is to use the **three drop-down menus** to select the state, the city in state, and the available restaurant types. After selecting the state and city, statistical information about different categories will be displayed. Then you can choose your favorite category based on them. At this time, the information of the restaurants in that category will be displayed in a table, or you can click the **Map button** to display their locations. In the table, you can click the "**Click here" buttons** to enter the detailed information page of the restaurant.
- **Required Python packages:** requests, plotyly, folium, pandas, flask and bs4.

Data sources

Data source 1:(4 points)Multiple related data files(one>1000 records).(source1,2,3) ***Data source 2:***(4 points)Web API requires API key✣.(source 4) ***Data source 3:***(8 points)Crawling and scraping multiple pages✣.(source 5)

Categories information

- **Origin:** Documentation Download **Format:** JSON$(> 1000 \text{ records})[\approx 1500]\{192 \text{ used}\}$
- **Data access and caching:** downloaded directly without additional verification methods, I used cache.
- **Summary of data:** It contains information about categories and available countries. Then, this data can be used as a benchmark for restaurant category search. Because this file contains all the categories. What I need to do is to filter out the category of restaurants from all categories.
- **Important fields:** ***"alias":*** alias of the child category, offer a different name for title. ***"title":*** title of the child category. Alias will be used to find different type of restaurant. ***"parents":*** belongs to what parent category. ***"country_whitelist":*** available countries (without this field means **TO ALL THE COUNTRIES**).

ISO 3166-1 alpha-2 code

- **Origin:** Documentation Download **Format:** JSON$(< 1000 \text{ records})[\approx 250]\{250 \text{ used}\}$
- **Data access and caching:** Downloaded directly without additional verification methods, I used cache.
- **Summary of data:** Because in the previous category file, there are information about different restaurant categories in which countries provide search services. Therefore, it is necessary to use the abbreviations of the names of each country in this file to determine whether this category can be searched in the US.
- **Important fields:** ***"Code":*** code of the country and improve its readability by providing the full name of the country. ***"Name":*** name of the country, which is better for human reading.

United States Cities Database

- **Origin:** Documentation Download **Format:** CSV$(> 1000\ \text{records})[\approx 28000]\{28399\ \text{used}\}$
- **Data access and caching:** downloaded directly without additional verification methods. I used cache.
- **Summary of data:** The main usage of this data source is to provide an effective state-city relationship for the flask app. What's more, this data source have a very useful definition for city, city id, state and state id. I can use it as a mark for each city and state.
- **Important fields:** ***"city":*** name of the city. ***"city_ascii":*** ascii version. I will use it as each city's name. ***"state_id":*** abbreviation for state. It is more convenient as a mark of the state. ***"state_name":*** full state name. ***"id":*** unique id for each city, which can be the primary key for each city in the database.

Using API key to get base information and do analysis

- **Origin:** Documentation **Format:** JSON$(> 1000\ \text{records})[\approx \infty]\{[50, 1000]\ \text{for each type will be used}\}$

  `GET https://api.yelp.com/v3/businesses/search` (Each request can only get up to 50 results and get up to 1000 results using multiple queries and combinations of the "limit" and "offset")

- **Data access and caching** : The Yelp Fusion API uses private key to authenticate. I used cache.

- **Summary of data** : What I'm trying to get from this API is the data of different categories of restaurants. In order to make the flask app faster, I decided to let each type of category of the restaurant only gets 50 records at most to draw the average rating bar plot. There are approximately 200 types of restaurants available in yelp in the US. Then, each plot needs nearly 10000 records of restaurants(The restaurant may not be completely unique, because the restaurant may have more than one category)

- ***Important fields in "businesses": "categories:"*** List of category title and alias pairs associated with this business. ***"id":*** Unique Yelp ID of this business. Example: `'4kMBvIEWPxWkWKFN__8SxQ'` . ***"name":*** Name of this business. ***"rating":*** Rating for this business. ***"coordinates":*** Coordinates of this business.

Crawling and scraping multiple pages in Yelp to gain information related covid-19

- **Origin:** https://www.yelp.com/ **Format:** HTML$[\approx \infty]\{[50, 1000]\ \text{for each type will be used}\}$
- **Data access and caching:** By crawling and scraping. I used cache.
- **Summary of data:** This part is not fixed, each restaurant has its own services dealing with covid-19. There are some basic services provided by yelp. But, the user and the owner of the restaurant can change the information on the webpage. I will get all of them from the site and provide it in my flask app to the users.
- **Important attributes:** ***"Updated Services":*** some basic services the restaurant can provide to the customer. ***"Health & Safety Measures:"*** what the health & satety measures the restaurant has implemented.

Database

Database schema

It contains all the restaurant categories (not all the categories provided by yelp fusion). The *"alias"* of the category is what the API accepts. We will use the *"alias"* in search actions and use the *"title"* for web pages.

```sql
CREATE TABLE IF NOT EXISTS restaurant_category_information(
"title" TEXT NOT NULL, # name of the category
"alias" TEXT NOT NULL, # alias of the name
"country_whitelist" TEXT # Which countries offer searches in this category);
```

ISO 3166-1 alpha-2 codes are two-letter country codes. This form is suitable for filtering which categories are available in which countries (Although my program is only used in the United States).

```sql
CREATE TABLE IF NOT EXISTS iso_3166_1_alpha_2_code(
"Code"  TEXT NOT NULL, # ISO 3166-1 alpha-2 code
"Name"  TEXT NOT NULL, # English short name officially used by the ISO 3166
PRIMARY KEY("Code"));
```

The purpose of this form is to help users select a certain city in a certain state, and then facilitate the flask application to deal with subsequent statistical problems. Location is needed when searching and this field associate with the search record table(*"restaurant_category_fetch"*).

```sql
CREATE TABLE IF NOT EXISTS us_states(
"city"  TEXT, # name of the city
"city_ascii"  TEXT, # ascii of the name of the city
"state_id"  TEXT, # alpha2 of the state (`'NY'` for New York)
"state_name"  TEXT, # name of the state (New York)
"county_fips" TEXT,
"county_name" TEXT,
"lat" REAL,
"lng" REAL,
"population"  INTEGER,
"density" INTEGER,
"source"  TEXT,
"military"  INTEGER,
"incorporated"  INTEGER,
"timezone"  TEXT,
"ranking" INTEGER,
"zips"  TEXT,
"id"  TEXT, # unique id for each city
PRIMARY KEY("id"));
```

This table is used to record restaurant information. Through the search API provided by yelp fusion, information related to the search results can be collected. I will store the restaurant information obtained by each search in the database because restaurant information is not frequently updated data. The information of category will be stored in another table named "restaurant_category" separately and this is for searching more conveniently.

```
CREATE TABLE IF NOT EXISTS restaurant_information(
"id"  TEXT, # id of each restaurant
"alias" TEXT, # alias of the restaurant
"name"  TEXT, # name of the restaurant
"image_url" TEXT, # url of the image of the restaurant
"is_closed" INTEGER, # whether is closed
"url" TEXT, # url in yelp
"review_count"  INTEGER,
"rating"  REAL,
"coordinates_latitude"  REAL,
"coordinates_longitude" REAL,
"transactions"  TEXT,
"price" TEXT,
"location"  TEXT,
"phone" TEXT,
"display_phone" TEXT,
PRIMARY KEY("id"));
```

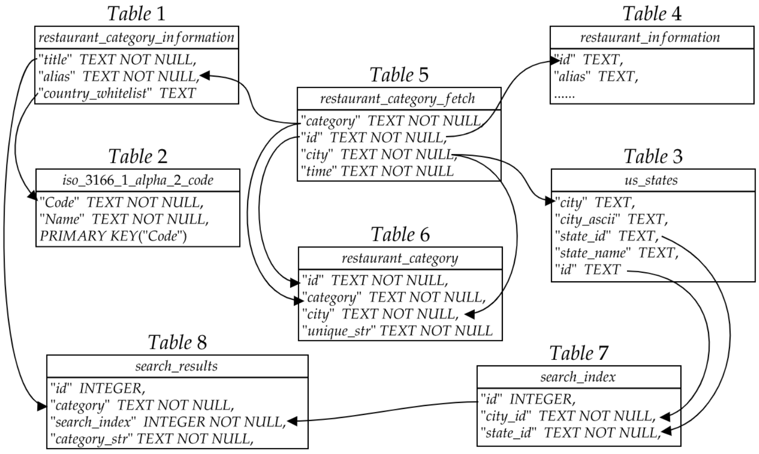
This two tables contain information for each search action. **They have different focuses, although the information recorded is roughly the same** (they can't become one table, because you can search one restaurant serval times but it should only insert into Table 2 once). Table 1 contains **necessary information** including location, category, **time** and shop id **for each search**. Table 2 helps to store each shop's **category** in a specific city by each search (restaurants have more than one type in general).**[These two are state-city level]**

```
CREATE TABLE IF NOT EXISTS restaurant_category_fetch (
"id"  TEXT NOT NULL, # id of restaurant(the same as restaurant_information.id)
"category"  TEXT NOT NULL, # searched category(the same as us_states.city_ascii)
"city" TEXT NOT NULL, # searched city(the same as restaurant_category_information.alias)
"time" TEXT NOT NULL # a time stamp);

CREATE TABLE IF NOT EXISTS restaurant_category (
"id"  TEXT NOT NULL, # id of restaurant(the same as restaurant_information.id)
"category"  TEXT NOT NULL, # searched category(the same as us_states.city_ascii)
"city" TEXT NOT NULL, # searched city(the same as restaurant_category_information.alias)
"unique_str" TEXT NOT NULL, # this is the unique key for id-category-city specificly
PRIMARY KEY("unique_str"));
```

These two tables are for each search's **results** (you select $\text{state} \rightarrow \text{city} \rightarrow \text{category}$), the first one generate and store a search index for the search, and the second one stores the category and display information(for webpages showing) for each search_index. **[These two are category level]**

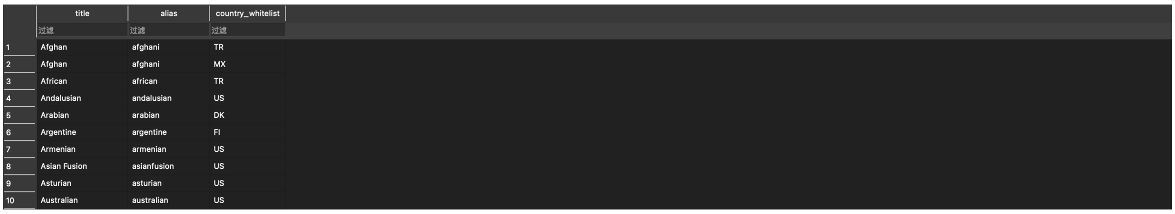
```
CREATE TABLE IF NOT EXISTS search_index (
"id"  INTEGER, # search index(the same as search_results.search_index)
"city_id" TEXT NOT NULL, # id of the city(the same as us_states.id)
"state_id"  TEXT NOT NULL, # id of the state(the same as us_states.state_id)
PRIMARY KEY("id" AUTOINCREMENT));

CREATE TABLE IF NOT EXISTS search_results (
"id"  INTEGER,  # id of each record, key of the table
"search_index"  INTEGER NOT NULL, # search index(the same as search_index.id)
"category"  TEXT NOT NULL, # category for each search(after you select state and city)
"category_str" TEXT NOT NULL, # this is a display version for webpages
PRIMARY KEY("id" AUTOINCREMENT));
```

Foreign key-primary key relation



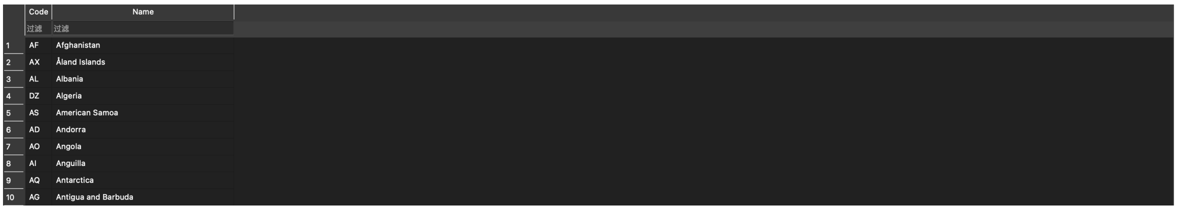
Screenshots of the data

- **Table 1:** restaurant_category_information



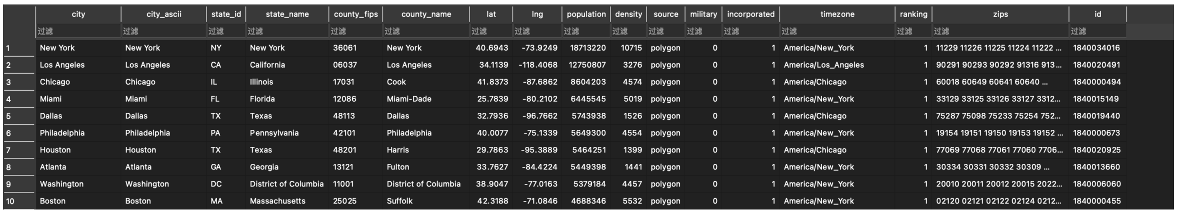
	title	alias	country_whitelist
	过滤	过滤	过滤
1	Afghan	afghani	TR
2	Afghan	afghani	MX
3	African	african	TR
4	Andalusian	andalusian	US
5	Arabian	arabian	DK
6	Argentine	argentine	FI
7	Armenian	armenian	US
8	Asian Fusion	asianfusion	US
9	Asturian	asturian	US
10	Australian	australian	US

- **Table 2:** iso_3166_1_alpha_2_code



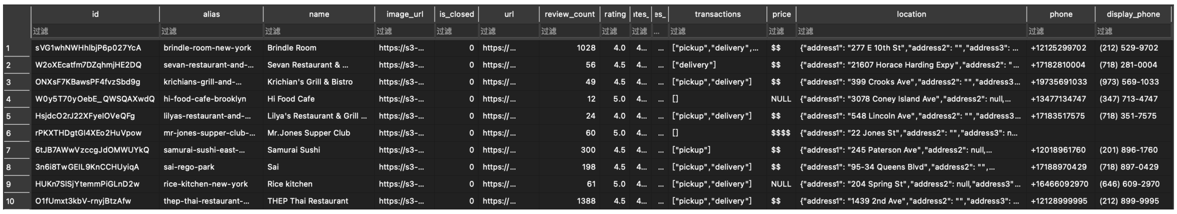
	Code	Name
	过滤	过滤
1	AF	Afghanistan
2	AX	Åland Islands
3	AL	Albania
4	DZ	Algeria
5	AS	American Samoa
6	AD	Andorra
7	AO	Angola
8	AI	Anguilla
9	AQ	Antarctica
10	AG	Antigua and Barbuda

- **Table 3:** us_states



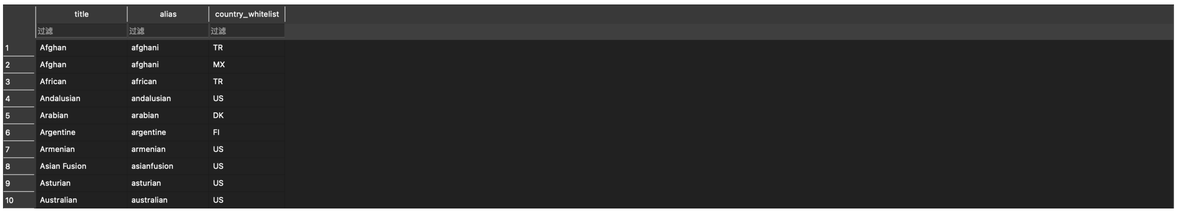
	city	city_ascii	state_id	state_name	county_fips	county_name	lat	lng	population	density	source	military	incorporated	timezone	ranking	zips	id
	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤
1	New York	New York	NY	New York	36061	New York	40.6943	-73.9249	18713220	10715	polygon	0	1	America/New_York	1	11229 11226 11225 11224 11222 ...	1840034016
2	Los Angeles	Los Angeles	CA	California	06037	Los Angeles	34.1139	-118.4068	12750807	3276	polygon	0	1	America/Los_Angeles	1	90291 90293 90292 91316 913...	1840020491
3	Chicago	Chicago	IL	Illinois	17031	Cook	41.8373	-87.6862	8604203	4574	polygon	0	1	America/Chicago	1	60018 60649 60641 60640 ...	1840000494
4	Miami	Miami	FL	Florida	12086	Miami-Dade	25.7839	-80.2102	6445545	5019	polygon	0	1	America/New_York	1	33129 33125 33126 33127 3312...	1840015149
5	Dallas	Dallas	TX	Texas	48113	Dallas	32.7936	-96.7662	5743938	1526	polygon	0	1	America/Chicago	1	75287 75098 75233 75254 752...	1840019440
6	Philadelphia	Philadelphia	PA	Pennsylvania	42101	Philadelphia	40.0077	-75.1339	5649300	4554	polygon	0	1	America/New_York	1	19154 19151 19150 19153 19152 ...	1840000673
7	Houston	Houston	TX	Texas	48201	Harris	29.7863	-95.3889	5464251	1399	polygon	0	1	America/Chicago	1	77069 77068 77061 77060 7706...	1840020925
8	Atlanta	Atlanta	GA	Georgia	13121	Fulton	33.7627	-84.4224	5449398	1441	polygon	0	1	America/New_York	1	30334 30331 30332 30309 ...	1840013660
9	Washington	Washington	DC	District of Columbia	11001	District of Columbia	38.9047	-77.0163	5379184	4457	polygon	0	1	America/New_York	1	20010 20011 20012 20015 2022...	1840006060
10	Boston	Boston	MA	Massachusetts	25025	Suffolk	42.3188	-71.0846	4688346	5532	polygon	0	1	America/New_York	1	02120 02121 02122 02124 0212...	1840000455

- **Table 4:** restaurant_information



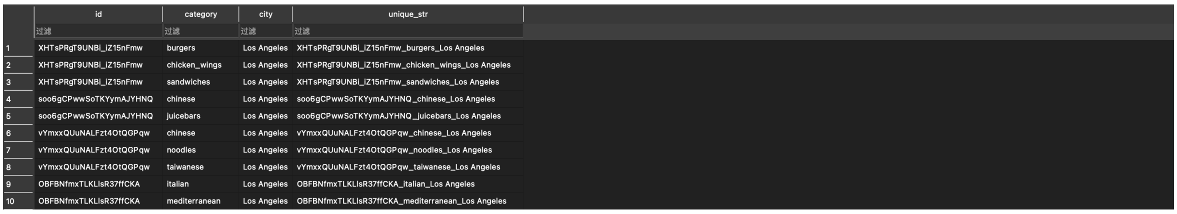
	id	alias	name	image_url	is_closed	url	review_count	rating	ıtes_	es_	transactions	price	location	phone	display_phone
	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤	...	过滤	过滤	过滤	过滤	过滤
1	sVG1whNWHhlbjP6p027YcA	brindle-room-new-york	Brindle Room	https://s3-...	0	https://...	1028	4.0	4...	...	["pickup","delivery",...	$$	{"address1": "277 E 10th St","address2": "","address3": ...	+12125299702	(212) 529-9702
2	W2oXEcatfm7DZqhmjHE2DQ	sevan-restaurant-and-...	Sevan Restaurant & ...	https://s3-...	0	https://...	56	4.5	4...	...	["delivery"]	$$	{"address1": "21607 Horace Harding Expy","address2": "...	+17182810004	(718) 281-0004
3	ONXsF7KBawsPF4fvzSbd9g	krichians-grill-and-...	Krichian's Grill & Bistro	https://s3-...	0	https://...	49	4.5	4...	...	["pickup","delivery"]	$$	{"address1": "399 Crooks Ave","address2": "","address3...	+19735691033	(973) 569-1033
4	W0y5T70yOebE_QWSQAXwdQ	hi-food-cafe-brooklyn	Hi Food Cafe	https://s3-...	0	https://...	12	5.0	4...	...	[]	NULL	{"address1": "3078 Coney Island Ave","address2": null,...	+13477134747	(347) 713-4747
5	HsjdcO2rJ22XFyelOVeQFg	lilyas-restaurant-and-...	Lilya's Restaurant & Grill ...	https://s3-...	0	https://...	24	4.0	4...	...	["pickup","delivery"]	$$	{"address1": "548 Lincoln Ave","address2": "","address3...	+17183517575	(718) 351-7575
6	rPKXTHDgtGI4XEo2HuVpow	mr-jones-supper-club-...	Mr.Jones Supper Club	https://s3-...	0	https://...	60	5.0	4...	...	[]	$$$$	{"address1": "22 Jones St","address2": "","address3": n...		
7	6tJB7AWwVzccgJdOMWUYkQ	samurai-sushi-east-...	Samurai Sushi	https://s3-...	0	https://...	300	4.5	4...	...	["pickup"]	$$	{"address1": "245 Paterson Ave","address2": null,...	+12018961760	(201) 896-1760
8	3n6i8TwGEIL9KnCCHUyiqA	sai-rego-park	Sai	https://s3-...	0	https://...	198	4.5	4...	...	["pickup","delivery"]	$$	{"address1": "95-34 Queens Blvd","address2": "",...	+17188970429	(718) 897-0429
9	HUKn7SlSjYtemmPiGLnD2w	rice-kitchen-new-york	Rice kitchen	https://s3-...	0	https://...	61	5.0	4...	...	["pickup","delivery"]	NULL	{"address1": "204 Spring St","address2": null,"address3"...	+16466092970	(646) 609-2970
10	O1fUmxt3kbV-rnyjBtzAfw	thep-thai-restaurant-...	THEP Thai Restaurant	https://s3-...	0	https://...	1388	4.5	4...	...	["pickup","delivery"]	$$	{"address1": "1439 2nd Ave","address2": "","address3": ...	+12128999995	(212) 899-9995

- **Table 5:** restaurant_category_fetch



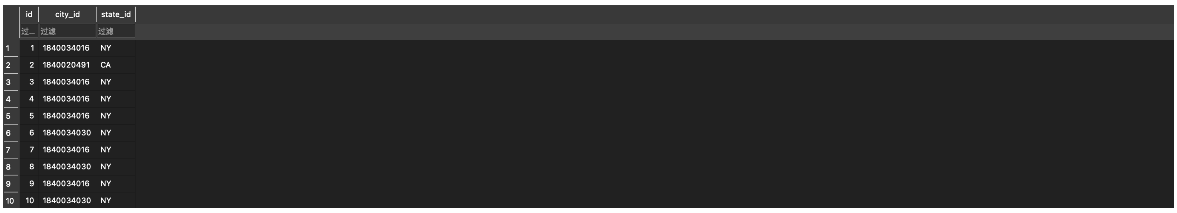
	title	alias	country_whitelist
	过滤	过滤	过滤
1	Afghan	afghani	TR
2	Afghan	afghani	MX
3	African	african	TR
4	Andalusian	andalusian	US
5	Arabian	arabian	DK
6	Argentine	argentine	FI
7	Armenian	armenian	US
8	Asian Fusion	asianfusion	US
9	Asturian	asturian	US
10	Australian	australian	US

- **Table 6:** restaurant_category



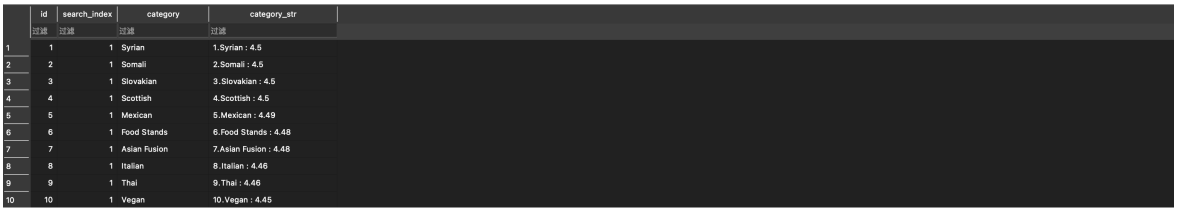
	id	category	city	unique_str
	过滤	过滤	过滤	过滤
1	XHTsPRgT9UNBi_iZ15nFmw	burgers	Los Angeles	XHTsPRgT9UNBi_iZ15nFmw_burgers_Los Angeles
2	XHTsPRgT9UNBi_iZ15nFmw	chicken_wings	Los Angeles	XHTsPRgT9UNBi_iZ15nFmw_chicken_wings_Los Angeles
3	XHTsPRgT9UNBi_iZ15nFmw	sandwiches	Los Angeles	XHTsPRgT9UNBi_iZ15nFmw_sandwiches_Los Angeles
4	soo6gCPwwSoTKYymAJYHNQ	chinese	Los Angeles	soo6gCPwwSoTKYymAJYHNQ_chinese_Los Angeles
5	soo6gCPwwSoTKYymAJYHNQ	juicebars	Los Angeles	soo6gCPwwSoTKYymAJYHNQ_juicebars_Los Angeles
6	vYmxxQUuNALFzt4OtQGPqw	chinese	Los Angeles	vYmxxQUuNALFzt4OtQGPqw_chinese_Los Angeles
7	vYmxxQUuNALFzt4OtQGPqw	noodles	Los Angeles	vYmxxQUuNALFzt4OtQGPqw_noodles_Los Angeles
8	vYmxxQUuNALFzt4OtQGPqw	taiwanese	Los Angeles	vYmxxQUuNALFzt4OtQGPqw_taiwanese_Los Angeles
9	OBFBNfmxTLKLIsR37ffCKA	italian	Los Angeles	OBFBNfmxTLKLIsR37ffCKA_italian_Los Angeles
10	OBFBNfmxTLKLIsR37ffCKA	mediterranean	Los Angeles	OBFBNfmxTLKLIsR37ffCKA_mediterranean_Los Angeles

- **Table 7:** search_index



	id	city_id	state_id
	过...	过滤	过滤
1	1	1840034016	NY
2	2	1840020491	CA
3	3	1840034016	NY
4	4	1840034016	NY
5	5	1840034016	NY
6	6	1840034030	NY
7	7	1840034016	NY
8	8	1840034030	NY
9	9	1840034016	NY
10	10	1840034030	NY

- **Table 8:** search_results



	id	search_index	category	category_str
	过滤	过滤	过滤	过滤
1	1	1	Syrian	1.Syrian : 4.5
2	2	1	Somali	2.Somali : 4.5
3	3	1	Slovakian	3.Slovakian : 4.5
4	4	1	Scottish	4.Scottish : 4.5
5	5	1	Mexican	5.Mexican : 4.49
6	6	1	Food Stands	6.Food Stands : 4.48
7	7	1	Asian Fusion	7.Asian Fusion : 4.48
8	8	1	Italian	8.Italian : 4.46
9	9	1	Thai	9.Thai : 4.46
10	10	1	Vegan	10.Vegan : 4.45

Interaction and Presentation Options

Description

The program allows users to select specific states and cities, and view the average ratings of different types of restaurants from the bar graph. You can first select a state and submit, and then select a city and submit. The program will generate bar graphs based on the state and city you provide. It contains two **bar graphs**, the first is the average rating, and the second is the statistics of each type of restaurantThen, you can choose a specific type of food, the program will present useful information about the covid-19 (including updated services and health & safety measures)and specific restaurant information presented in a **table** to help you decide where to go. I will only show some useful information in the table(including name, image, url, and rating). The program will also provide a **plot of map** to facilitate you to choose a location closer to you. You can click the Map button to switch the table to the map.Finally, you can choose one restaurant specifically, the program will offer you more details in **text**. This part will contain many details for each store. The URL is also provided if you want to make an order or see more details on the yelp webpage.

Besides the flask app, I also provide the **logging information** in the command line. You can know what exactly the flask app is doing from the command line. This including several types: yelp_covid, yelp_fusion, CSV, flask, database, and cache. Each of them represents a specific python module and you can figure out the relationship easily.

Technologies

- **Flask:** The whole program is running within a Flask App.
- **Plotly:** I use plotly to draw bar plot. There are some statistic data for covid 19 I can use.
- **Folium:** This is a map package, I use it to draw a map to show the locations of the restaurants.
- **Command line:** Only for logging, you can see what the flask app is doing from the command line.

Brief instructions

The entire program is built within the **Flask app**, and the interaction is relatively simple. The main thing is to use the **three drop-down menus** to select the state, the city in state, and the available restaurant types. After selecting the state and city, statistical information about different categories will be displayed. Then you can choose your favorite category based on them. At this time, the information of the restaurants in that category will be displayed in a table, or you can click the **Map button** to display their locations. In the table, you can click the **"Click here" buttons** to enter the detailed information page of the restaurant.

Demo Link

https://youtu.be/IKSoSzoPjVI