

Final Project Data CheckPoint

Project code

<https://github.com/Huadous/final-project>

Data sources

Categories information

1. Origin : [Documentation](#) [Download](#)

```
[  
  {  
    "alias": "3dprinting",  
    "title": "3D Printing",  
    "parents": [  
      "localservices"  
    ]  
  },  
  {  
    "alias": "abruzzese",  
    "title": "Abruzzese",  
    "parents": [  
      "italian"  
    ],  
    "country_whitelist": [  
      "IT"  
    ]  
  },  
  {  
    "alias": "absinthebars",  
    "title": "Absinthe Bars",  
    "parents": [  
      "bars"  
    ],  
    "country_whitelist": [  
      "CZ"  
    ]  
},
```



2. Format : JSON(> 1000 records)

3. Data access and caching : downloaded directly without additional verification methods, I used cache.

4. Summary of data : [≈ 1500 | will use 192]

It contains information about categories and available countries. Therefore, this data can be used as a benchmark for restaurant category search. Because this is an all-category file, which contains not only the categories of restaurants. What I need to do is to filter out the category of the restaurant from all categories.

Important fields:

- "alias" : alias of the child category, offer a different name for title.
- "title" : title of the child category and will be used to find different type of restaurant.
- "parents" : belongs to what parent category. The program use this to find which category belongs to restaurants.
- "country_whitelist" : available countries (without this field means TO ALL THE COUNTRIES)

5. Evidence of caching

```
file_name = 'categories_information'  
file_type = 'categories'  
categories_information = cache.sync_cache(file_name, file_type)  
if categories_information == {} or len(categories_information) == 0:  
    categories_information_url_json = 'https://www.yelp.com/developers/documentation/v3/all_categories.json'  
    r = requests.get(categories_information_url_json)  
    categories_information = json.loads(r.text)  
    cache.save_cache(file_name, file_type, categories_information)
```

```
file_name = 'restaurant_category_information'  
file_type = 'categories'  
restaurant_category_information = cache.sync_cache(file_name, file_type)
```

```
if restaurant_category_information == {} or len(restaurant_category_information) == 0:  
    restaurant_category_information = []
```

```
for ele in categories_information:  
    if 'parents' in ele and len(ele['parents']) > 0 and ele['parents'][0] == 'restaurant':
```

```
        restaurant_category_information.append(ele)
```

```
cache.save_cache(file_name, file_type, restaurant_category_information)
```



名称	修改日期	大小	种类
categories	今天下午 3:31	--	文件夹
categories_information.json	今天下午 5:37	173 KB	JSON Document
restaurant_category_information.json	今天下午 6:36	21 KB	JSON Document
covid_services	今天下午 3:31	--	文件夹
Locations	今天下午 3:31	--	文件夹
restaurant_search	今天下午 7:19	--	文件夹
American (New)_New_York_0.json	今天下午 7:05	48 KB	JSON Document
American (New)_Orlando_0.json	今天下午 7:18	51 KB	JSON Document
American (New)_Rockford_0.json	今天下午 7:14	49 KB	JSON Document
American (Traditional)_New_York_0.json	今天下午 7:06	48 KB	JSON Document
American (Traditional)_Orlando_0.json	今天下午 7:19	51 KB	JSON Document
American (Traditional)_Rockford_0.json	今天下午 7:15	49 KB	JSON Document
Andalusian_New_York_0.json	今天下午 6:49	48 KB	JSON Document
Andalusian_Orlando_0.json	今天下午 6:55	48 KB	JSON Document
Andalusian_Rockford_0.json	今天下午 7:16	51 KB	JSON Document
Andalusian_Rockford_0.json	今天下午 7:13	49 KB	JSON Document

Macintosh HD > 用户 > huayu > 文稿 > canvas > finalproj > cache > categories > categories_information.json

ISO 3166-1 alpha-2 code

1. Origin : [Documentation](#) [Download](#)

```
[{"Code": "AF", "Name": "Afghanistan"}, {"Code": "AX", "Name": "Åland Islands"}, {"Code": "AZ", "Name": "Albania"}, {"Code": "DZ", "Name": "Algeria"}, {"Code": "AS", "Name": "American Samoa"}, {"Code": "AD", "Name": "Andorra"}, {"Code": "AO", "Name": "Angola"}, {"Code": "AI", "Name": "Anguilla"}, {"Code": "AQ", "Name": "Antarctica"}, {"Code": "AG", "Name": "Antigua and Barbuda"}, {"Code": "AR", "Name": "Argentina"}, {"Code": "AM", "Name": "Armenia"}, {"Code": "AW", "Name": "Aruba"}, {"Code": "AU", "Name": "Australia"}, {"Code": "AT", "Name": "Austria"}, {"Code": "AZ", "Name": "Azerbaijan"}, {"Code": "BS", "Name": "Bahamas"}, {"Code": "BH", "Name": "Bahrain"}, {"Code": "BD", "Name": "Bangladesh"}, {"Code": "BB", "Name": "Belarus"}, {"Code": "BE", "Name": "Belgium"}, {"Code": "BZ", "Name": "Belize"}, {"Code": "BJ", "Name": "Benin"}, {"Code": "BM", "Name": "Bermuda"}, {"Code": "BT", "Name": "Bhutan"}, {"Code": "BO", "Name": "Bolivia, Plurinational State of"}, {"Code": "BW", "Name": "Botswana"}, {"Code": "BV", "Name": "Bouvet Island"}, {"Code": "BR", "Name": "Brazil"}, {"Code": "IO", "Name": "British Indian Ocean Territory"}, {"Code": "BN", "Name": "Brunei Darussalam"}, {"Code": "BG", "Name": "Bulgaria"}, {"Code": "BP", "Name": "Burkina Faso"}, {"Code": "BI", "Name": "Burundi"}, {"Code": "KH", "Name": "Cambodia"}, {"Code": "CM", "Name": "Cameroon"}, {"Code": "CA", "Name": "Canada"}, {"Code": "CV", "Name": "Cape Verde"}, {"Code": "KY", "Name": "Cayman Islands"}, {"Code": "CF", "Name": "Central African Republic"}, {"Code": "TD", "Name": "Chad"}, {"Code": "CL", "Name": "Chile"}, {"Code": "CN", "Name": "China"}, {"Code": "CI", "Name": "Christmas Island"}, {"Code": "CC", "Name": "Cocos (Keeling) Islands"}, {"Code": "CO", "Name": "Colombia"}, {"Code": "KM", "Name": "Comoros"}, {"Code": "CG", "Name": "Congo"}, {"Code": "CD", "Name": "Democratic Republic of the Congo"}, {"Code": "CR", "Name": "Costa Rica"}, {"Code": "CI", "Name": "Côte d'Ivoire"}, {"Code": "HR", "Name": "Croatia"}, {"Code": "CU", "Name": "Cuba"}, {"Code": "CW", "Name": "Curaçao"}, {"Code": "CY", "Name": "Cyprus"}, {"Code": "CZ", "Name": "Czech Republic"}, {"Code": "DK", "Name": "Denmark"}, {"Code": "DJ", "Name": "Djibouti"}, {"Code": "DM", "Name": "Dominica"}, {"Code": "DO", "Name": "Dominican Republic"}, {"Code": "EC", "Name": "Ecuador"}, {"Code": "EG", "Name": "Egypt"}, {"Code": "SV", "Name": "El Salvador"}, {"Code": "GQ", "Name": "Equatorial Guinea"}, {"Code": "ER", "Name": "Eritrea"}, {"Code": "EE", "Name": "Estonia"}, {"Code": "ET", "Name": "Ethiopia"}, {"Code": "FK", "Name": "Falkland Islands (Malvinas)"}, {"Code": "FO", "Name": "Faroe Islands"}, {"Code": "FJ", "Name": "Fiji"}, {"Code": "FI", "Name": "Finland"}, {"Code": "FR", "Name": "France"}, {"Code": "GF", "Name": "French Guiana"}, {"Code": "PF", "Name": "French Polynesia"}, {"Code": "TF", "Name": "French Southern Territories"}, {"Code": "GA", "Name": "Gabon"}, {"Code": "GM", "Name": "Gambia"}, {"Code": "GE", "Name": "Georgia"}, {"Code": "DE", "Name": "Germany"}, {"Code": "GH", "Name": "Ghana"}, {"Code": "GI", "Name": "Gibraltar"}, {"Code": "GR", "Name": "Greece"}, {"Code": "GL", "Name": "Greenland"}, {"Code": "GD", "Name": "Grenada"}, {"Code": "GP", "Name": "Guadeloupe"}, {"Code": "GU", "Name": "Guam"}, {"Code": "GT", "Name": "Guatemala"}, {"Code": "GG", "Name": "Guernsey"}, {"Code": "GG", "Name": "Guernsey"}, {"Code": "GT", "Name": "Guatemala"}, {"Code": "GG", "Name": "Guernsey"}, {"Code": "GT", "Name": "Guatemala"}, {"Code": "GG", "Name": "Guernsey"}]
```

2. Format : JSON(< 1000 records)

3. Data access and caching : downloaded directly without additional verification methods, I used cache.

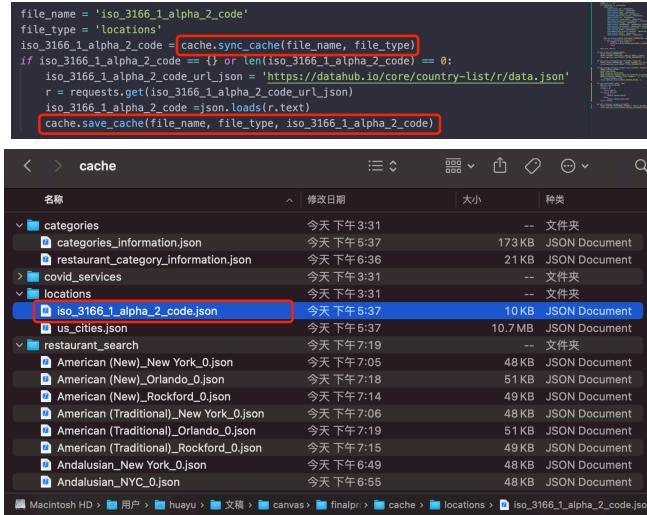
4. Summary of data : [≈ 250 I will use all of them]

Because in the previous category file, there will be 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 the restaurant in this category can be searched in the United States

Important fields :

- "Code" : code of the country, this field is the same as the "country_whitelist" and improve its readability by providing the full name of the country.
- "Name" : name of the country, which is better for human reading.

5. Evidence of caching



United States Cities Database

1. Origin: [Documentation](#) [Download](#)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1	city	city_ascii	state_id	state_name	county_fips	county_name	lat	lng	population	density	source	military	incorporated	timezone	ranking	zips	id
2	New York	New York	NY	New York	36061	New York	40.6943	-73.9249	18713220	10715	polygon	FALSE	TRUE	America/Ne	1	11229 11226 1840034016	
3	Los Angeles	Los Angeles	CA	California	6037	Los Angeles	34.1139	-118.4068	12750807	3276	polygon	FALSE	TRUE	America/Los	1	80291 90293 1840020491	
4	Chicago	Chicago	IL	Illinois	17031	Cook	41.8373	-87.6862	8604203	4574	polygon	FALSE	TRUE	America/Ch	1	80018 60649 184000494	
5	Miami	Miami	FL	Florida	12068	Miami-Dade	25.7636	-80.1704	6404545	5019	polygon	FALSE	TRUE	America/Ne	1	33129 125 1840015149	
6	Dallas	Dallas	TX	Texas	48113	Dallas	32.7036	-96.7662	5743938	1554	polygon	FALSE	TRUE	America/	1	75295 75295 1840004440	
7	Philadelphia	Philadelphia	PA	Pennsylvania	42101	Philadelphia	40.0377	-75.1339	5649200	4654	polygon	FALSE	TRUE	America/Ne	1	19154 19151 184000073	
8	Houston	Houston	TX	Texas	48201	Harris	29.7663	-95.3689	5464251	1599	polygon	FALSE	TRUE	America/Ch	1	77069 77069 184002025	
9	Atlanta	Atlanta	GA	Georgia	13121	Fulton	33.7627	-84.4224	5449898	1441	polygon	FALSE	TRUE	America/Ne	1	30334 30331 1840013636	
10	Washington	Washington	DC	District of C	11001	District of C	38.9047	-77.0163	5379184	4457	polygon	FALSE	TRUE	America/Ne	1	20010 20011 1840006060	
11	Boston	Boston	MA	Massachusetts	26026	Suffolk	42.3188	-71.0946	4682346	5532	polygon	FALSE	TRUE	America/Ne	1	02120 02121 184000465	
12	Phoenix	Phoenix	AZ	Arizona	4013	Mariopa	33.5722	-112.0891	4219697	1253	polygon	FALSE	TRUE	America/Ph	1	85008 85005 1840020568	
13	Seattle	Seattle	WA	Washington	53033	King	47.6211	-122.3244	3789215	3469	polygon	FALSE	TRUE	America/Los	1	98109 98108 1840021117	
14	San Francisco	San Francisco	CA	California	6075	San Francisc	37.7562	-122.443	3592294	7256	polygon	FALSE	TRUE	America/Los	1	94130 94131 1840021543	

2. Format : CSV (>1000 records [≈ 28000])

3. Data access and caching: downloaded directly without additional verification methods. I used cache.

4. Summary of data : [28399 I will use all of them]

The main usage of this data source is to provide an effective direct 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. This is more general, and I will use it as each city's name.
- "state_id" : abbreviation for state. This is more convenient than the "state_name" and can be used 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.

5. Evidence of caching

```
source_path = './src/location/uscities.csv'
file_name = 'us_cities'
file_type = 'locations'
us_cities = cache.sync_cache(file_name, file_type)
if us_cities == {} or len(us_cities) == 0:
    us_cities = csv_helper.csv_to_dict(source_path)
cache.save_cache(file_name, file_type, us_cities)
```

名称	修改日期	大小	种类
categories	今天下午9:23	--	文件夹
categories_information.json	今天下午9:23	173 KB	JSON Document
restaurant_category_information.json	今天下午9:23	21 KB	JSON Document
covid_services	今天下午9:23	--	文件夹
locations	今天下午9:23	--	文件夹
iso_3166_1_alpha_2_code.json	今天下午9:23	10 KB	JSON Document
us_cities.json	今天下午9:23	10.7 MB	JSON Document
restaurant_search	今天下午9:28	--	文件夹
American (New)_Los Angeles_0.json	今天下午9:26	51 KB	JSON Document
American (Traditional)_Los Angeles_0.json	今天下午9:27	51 KB	JSON Document
Andalusian_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document
Armenian_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document
Asian Fusion_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document
Asturian_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document
Australian_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document
Baguettes_Los Angeles_0.json	今天下午9:24	51 KB	JSON Document

Using API key to get base information and do analysis

1. Origin : [Documentation](#) Request: GET <https://api.yelp.com/v3/businesses/search>

```
{"time": "2021-04-19 12:05:44", "cache": {"businesses": [{"id": "H4jJ7XB3CeIfrpg56CczQ", "alias": "levain-bakery-new-york", "name": "Levain Bakery", "image_url": "https://s3-media2.fl.yelpcdn.com/bphoto/jCdXah-NjPa0Lb-30Buw/o.jpg", "is_closed": false, "url": "https://www.yelp.com/biz/levain-bakery-new-york?adjust_creative=BEX42Nx4TWW@0caxy9tW&utm_campaign=yelp_api_v3&utm_medium=api_v3_business_search&utm_source=BEX42Nx4TWW@0caxy9tW", "review_count": 8340, "categories": [{"alias": "bakeries", "title": "Bakeries"}]}, {"rating": 4.5, "coordinates": {"latitude": 40.779961, "longitude": -73.980299}, "transactions": [], "price": "$$", "location": {"address1": "167 W 74th St", "address2": "", "address3": "", "city": "New York", "zip_code": "10023", "country": "US", "state": "NY", "display_address": ["167 W 74th St", "New York, NY 10023"]}, "phone": "+19174643769", "display_phone": "(917) 464-3769", "distance": 8369.262424680568}, {"id": "V7LXZKB0zSc0eG8BjmnzSA", "alias": "Katz's-delicatessen-new-york", "name": "Katz's Delicatessen", "image_url": "https://s3-media4.fl.yelpcdn.com/bphoto qr7eSU6CfWkGz7Rc-QEoTQ/
```

2. Format : JSON (Each request can only get up to 50 results. you still can only get up to 1000 results using multiple queries and combinations of the "limit" and "offset" parameters)

3. Data access and caching : The Yelp Fusion API uses private key authentication to authenticate all endpoints. I used cache.

4. Summary of data : [I will use [50, 1000] for each category of restaurant]

What I'm trying to get from this API is the data of different categories of restaurants. Each request can get up to 50 results. 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 :

- "total" : Total number of business Yelp finds based on the search criteria.
- "businesses" : List of business Yelp finds based on the search criteria.
 - "categories" : List of category title and alias pairs associated with this business.
 - "coordinates" : Coordinates of this business.
 - "id" : Unique Yelp ID of this business. Example: '4kMBvIEWPxWkWKFN__8SxQ'
 - "name" : Name of this business.
 - "rating" : Rating for this business (value ranges from 1, 1.5, ... 4.5, 5).

5. Evidence of caching

```

yelp_fusion_business_search_url = 'https://api.yelp.com/v3/businesses/search'
headers = {"Authorization": "Bearer {}".format(API_KEY)}

def api_search(category, location, offset=0):
    file_name = category + '_' + location + '_' + str(offset)
    file_type = 'restaurant_search'
    api_search = cache.sync_cache(file_name, file_type)
    if api_search == {}:
        params = {
            'categories' : category,
            'location' : location,
            'offset' : offset,
            'limit' : 50
        }
        r = requests.get(yelp_fusion_business_search_url, headers=headers, params=params)
        api_search = json.loads(r.text)
        cache.save_cache(file_name, file_type, api_search)
    return api_search

```



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

1. Origin : website : <https://www.yelp.com/>

COVID-19 Updates

Updated Services

- ✓ Delivery
- ✓ Takeout
- ✓ Sit-down dining
- ✗ Outdoor seating
- ✓ Indoor dining

Health & Safety Measures Based on info from the business or our users ⓘ

- ✓ Limited capacity
- ✓ Masks required

Make a Reservation

Mon, Apr 19

7:00 pm

2 people

Find a Table

Waitlist closed

Waitlist usually opens at 5:00 pm

The restaurant is not taking waitlist parties right

2. Format : Html

3. Data access and caching : By crawling and scraping. I used cache.

4. Summary of data : [I will use [50, 1000] for each category of restaurant]

This part of the data is not fixed, each restaurant has its own services dealing with covid-19. There are some basic services provided by yelp. However, the user and also the owner of the restaurant can change the information on the website. I'm trying to get all of the services 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 & safety measures the restaurant has implemented.

5. Evidence of caching

```

try:
    with open("./cache/covid_services/" + url.replace('.', '_').replace('/', '&') + '.json', 'r') as f:
        text = f.read()
        print("Using Cache : covid_services")
        data = json.loads(text)
        if not ("time" in data and "html" in data):
            raise()
        d = now - datetime.datetime.strptime(data['time'], '%Y-%m-%d %H:%M:%S')
        if now - d > 3600 > 2.0:
            raise()
        self.curbside_pickup = data['element']['curbside_pickup']
        self.sit_down_dining = data['element']['sit_down_dining']
        self.delivery = data['element']['delivery']
        self.takeout = data['element']['takeout']
        self.outdoor_seating = data['element']['outdoor_seating']
        # Health & Safety Measures
        self.staff_wears_masks = data['element']['staff_wears_masks']
        self.social_distancing_enforced = data['element']['social_distancing_enforced']
        self.masks_required = data['element']['masks_required']
        self.limited_capacity = data['element']['limited_capacity']
        self.staff_wears_gloves = data['element']['staff_wears_gloves']
        self.sanitizing_between_customers = data['element']['sanitizing_between_customers']
        self.temperature_checks = data['element']['temperature_checks']
        self.hand_sanitizer_provided = data['element']['hand_sanitizer_provided']
        self.contactless_payments = data['element']['contactless_payments']
        self.other_service = data['element']['other_service']
except:
    try:
        with open("./cache/yelp_pages/" + url.replace('.', '_').replace('/', '&') + '.json', 'r') as f:
            text = f.read()
            print("Using Cache : yelp_pages")
            data = json.loads(text)
            if not ("time" in data and "html" in data):
                raise()
    except:
        # make yelp pages as cache
        data_yelp_pages_json = {"time": str(now)[0:19], "html": text}
        with open("./cache/yelp_pages/" + url.replace('.', '_').replace('/', '&') + '.json', 'w', encoding='utf-8') as f:
            f.write(json.dumps(data_yelp_pages_json, indent=4, ensure_ascii=False))

# make covid services as cache
element = {
    'curbside_pickup': self.curbside_pickup,
    'sit_down_dining': self.sit_down_dining,
    'delivery': self.delivery,
    'takeout': self.takeout,
    'outdoor_seating': self.outdoor_seating,
    'staff_wears_masks': self.staff_wears_masks,
    'social_distancing_enforced': self.social_distancing_enforced,
    'masks_required': self.masks_required,
    'limited_capacity': self.limited_capacity,
    'staff_wears_gloves': self.staff_wears_gloves,
    'sanitizing_between_customers': self.sanitizing_between_customers,
    'temperature_checks': self.temperature_checks,
    'hand_sanitizer_provided': self.hand_sanitizer_provided,
    'contactless_payments': self.contactless_payments,
    'other_service': self.other_service
}
data_covid_services_json = {"time": str(now)[0:19], "element": element}
with open("./cache/covid_services/" + url.replace('.', '_').replace('/', '&') + '.json', 'w', encoding='utf-8') as f:
    f.write(json.dumps(data_covid_services_json, indent=4, ensure_ascii=False))

```

Database

Database schema

```

1 | CREATE TABLE IF NOT EXISTS restaurant_category_information(
2 |     "title" TEXT NOT NULL, # name of the category
3 |     "alias" TEXT NOT NULL, # alias of the name
4 |     "country_whitelist" TEXT # Which countries offer searches in this category);

```

It contains all the restaurant categories (not all the categories provided by yelp fusion)

```

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

```

ISO 3166-1 alpha-2 codes are two-letter country codes defined in ISO 3166-1, part of the ISO 3166 standard published by the International Organization for Standardization (ISO), to represent countries, dependent territories, and special areas of geographical interest. This form is suitable for filtering which categories are available in which countries (although my program is only used in the United States).

```

1 | CREATE TABLE IF NOT EXISTS us_states(
2 |     "city" TEXT, # name of the city
3 |     "city_ascii" TEXT, # ascii of the name of the city
4 |     "state_id" TEXT, # alpha2 of the state (^'NY` for New York)
5 |     "state_name" TEXT, # name of the state (New York)
6 |     "county_fips" TEXT,
7 |     "county_name" TEXT,
8 |     "lat" REAL,
9 |     "lng" REAL,
10 |     "population" INTEGER,
11 |     "density" INTEGER,
12 |     "source" TEXT,

```

```

13 "military" INTEGER,
14 "incorporated" INTEGER,
15 "timezone" TEXT,
16 "ranking" INTEGER,
17 "zips" TEXT,
18 "id" TEXT, # unique id for each city
19 PRIMARY KEY("id"));

```

The purpose of this form is to help users select a certain city in a certain state, and then facilitate the flask application to continue to deal with subsequent statistical problems of different categories of restaurant ratings. You can see it from the pic in the *Interaction and Presentation Plans* part. Because location is needed when searching, all the location is associated with the search record table (restaurant_category_fetch).

```

1 CREATE TABLE IF NOT EXISTS restaurant_information(
2   "id" TEXT, # id of each restaurant
3   "alias" TEXT, # alias of the restaurant
4   "name" TEXT, # name of the restaurant
5   "image_url" TEXT, # url of the image of the restaurant
6   "is_closed" INTEGER, # whether is closed
7   "url" TEXT, # url in yelp
8   "review_count" INTEGER,
9   "categories" TEXT,
10  "rating" REAL,
11  "coordinates_latitude" REAL,
12  "coordinates_longitude" REAL,
13  "transactions" TEXT,
14  "price" TEXT,
15  "location" TEXT,
16  "phone" TEXT,
17  "display_phone" TEXT,
18 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.

```

1 CREATE TABLE IF NOT EXISTS restaurant_category_fetch (
2   "id" TEXT NOT NULL, # id of restaurant(the same as restaurant_information.id)
3   "category" TEXT NOT NULL, # searched category(the same as us_states.city_ascii)
4   "city" TEXT NOT NULL # searched city(the same as restaurant_category_information.title));

```

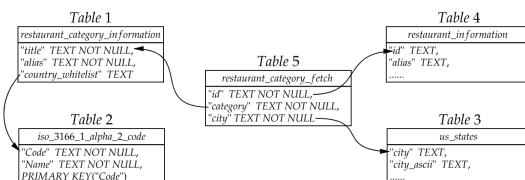
This is a table that records each search, which contains necessary information including location, category, and returned shop id. They are respectively associated with the three tables, please see the follow-up for details. This table may change in the future, or there is another table to filter the information of the categories of valid stores.

Foreign key-primary key relations

```

1 1. restaurant_category_information.country_whitelist = iso_3166_1_alpha_2_code.Code
2 2. restaurant_category_fetch.id = restaurant_information.id
3 3. restaurant_category_fetch.city = us_states.city_ascii
4 4. restaurant_category_fetch.category = restaurant_category_information.title

```



Screenshots of the data

- Table 1 : restaurant_category_information

	title	alias	country_whitelist
1	Afghan	afghani	TB
2	Afghan	afghani	MX
3	African	african	TB
4	Andalaskan	andalasian	US
5	Arabian	arabian	DK
6	Argentine	argentine	FI
7	Armenian	armesian	US
8	Asian Fusion	asianfusion	US
9	Australian	australian	US
10	Australian	australian	US
11	Austrian	austrian	ES
12	Austrian	austrian	DK
13	Baguettes	baguettes	US
14	Bangladeshi	bangladeshi	DK

- Table 2 : iso_3166_1_alpha_2_code

Code	Name
AF	Afghanistan
AX	Åland Islands
AL	Albania
DZ	Algeria
AS	American Samoa
AD	Andorra
AO	Angola
AI	Anguilla
AQ	Antarctica
AG	Antigua and Barbuda
AR	Argentina
AM	Armenia
AW	Aruba

- Table 3 : us_states

city	city_ascii	lat	state_name	unity_id	county_name	lat	lon	population	density	source	incorporated	timezone	ranking	zips	id
New York	NY	40.6945	New York	36061	New York	40.6945	-73.9249	16713220	1075	polygon	0	1	America/New_York	11229	11228 11225 11224 11222 11221 11220 11385 10169...
Los Angeles	Los_Angeles	34.1159	CA	60637	Los Angeles	34.1159	-116.4068	12790807	3276	polygon	0	1	America/Los_Angeles	80259	80283 90292 91316 91307 90031 90008...
Chicago	Chicago	41.8737	IL	70301	Cook	41.8737	-87.6882	8604203	4574	polygon	0	1	America/Chicago	60051	60049 60641 60640 60643 60642 60645 60644...
Miami	Miami	25.7839	FL	12086	Miami-Dade	25.7839	-80.2102	644584	5019	polygon	0	1	America/New_York	33129	33125 32928 33127 33128 33149 33144 33145...
Dallas	Dallas	32.7935	TX	48119	Dallas	32.7935	-96.7662	5763938	1526	polygon	0	1	America/Chicago	7529	76298 75232 76254 75252 75253 75023...
Philadelphia	Philadelphia	40.0077	PA	42101	Philadelphia	40.0077	-76.1339	5649300	4554	polygon	0	1	America/New_York	19514	19151 19160 19153 19152 19152 19102 1910...
Houston	Houston	29.7863	TX	48201	Harris	29.7863	-95.3889	5464251	1399	polygon	0	1	America/Chicago	7065	77068 77065 77062 77062 77065 77064...
Atlanta	Atlanta	33.7627	GA	13021	Fulton	33.7627	-84.4224	5443998	1441	polygon	0	1	America/New_York	30338	30331 30332 30330 30339 30337 30337 3033...
Washington	Washington	38.9047	DC	13001	District of ..	38.9047	-77.0161	5379894	4457	polygon	0	1	America/New_York	20010	20011 20012 20013 20232 20232 20307 20419...
Boston	Boston	42.3188	MA	25025	Suffolk	71.0848	-71.0848	468534	5532	polygon	0	1	America/New_York	21210	21211 21212 21214 21216 21216 21217 21218...
Phoenix	Phoenix	33.5722	AZ	40113	Metroplex	33.5722	-112.0891	4210897	1253	polygon	0	1	America/Phoenix	85009	85009 85007 85007 85004 85003 85008 85007...
Seattle	Seattle	47.6211	WA	53035	King	47.6211	-122.3244	3789215	3449	polygon	0	1	America/Los_Angeles	88108	88108 88108 88107 88107 88101 88101 88102...
San Francisco	San Francisco	37.7562	CA	60755	San Francisco	37.7562	-122.4433	3592294	7256	polygon	0	1	America/Los_Angeles	94159	94131 94131 94133 94133 94109 94109 94103...
Detroit	Detroit	42.3854	MI	26163	Wayne	-83.1024	3505126	1864	polygon	0	1	America/Detroit	48209	48208 48207 48207 48205 48205 48203 48202...	
San Diego	San Diego	32.8312	CA	60735	San Diego	32.8312	-117.1225	3220118	1686	polygon	0	1	America/Los_Angeles	9710	97108 97108 97101 97115 97115 97120 97120...
Minneapolis	Minneapolis	44.9635	MN	27053	Hennepin	44.9635	-93.2679	2977172	3071	polygon	0	1	America/Chicago	15450	15449 15449 15447 15446 15446 15446 15442...
Tampa	Tampa	27.8982	FL	12087	Hillsborough	27.8982	-82.4481	2808063	1351	polygon	0	1	America/New_York	33637	33629 33629 33621 33620 33619 33619 33613 33610...
Denver	Denver	39.7621	CO	60831	Peterson	39.7621	-104.6759	2876620	1831	polygon	0	1	America/Denver	80224	80230 80230 80211 80223 80237 80235 80238 80239...

- Table 4 : restaurant_information

id	alias	name	image_url	do	url	ew_co	categories	ating	latitudes	latitudes_low	latitudes_high	transactions	price	location	phone	isiply_phon
1	xlnEXM1DF6z4...	The halal-guys-new-york-2	The Halal Guys	https://s...	0	https://www.yelp.co...	["halal","fastfood"]	"title": "Food..."	4.0	40.7618...	-73.87...	"pickup": ..., "address": "W 53..."	"address": "W 53...	"address": "W 53...	"address": "W 53...	"address": "W 53...
2	StreBQhWbOQX...	the-cowfish-sushi-bar-bar...	The Cowfish Sushi Bar...	https://s...	0	https://www.yelp.co...	["halibut","sushi","title": "Sushi Bar"]	"title": "Sushi Bar..."	4.8	28.473...	-81.46...	"pickup": ..., "address": "E 42...	"address": "E 42...	"address": "E 42...	"address": "E 42...	"address": "E 42...
3	vSHHm...	the-toothsome-chocolate...	The Toothsome Chocolate...	https://s...	0	https://www.yelp.co...	["chocolate","dessert"]	"title": "Chocolate Dessert"	4.8	28.473...	-81.46...	"pickup": ..., "address": "E 42...	"address": "E 42...	"address": "E 42...	"address": "E 42...	"address": "E 42...
4	FEVQzobPwO...	shake-shack-new-york-2	Shake Shack	https://s...	0	https://www.yelp.co...	["shakes","burgers"]	"title": "Burgers" ...	4.0	40.7474...	-73.98...	"delivery": ..., "address": "E 23...	"address": "E 23...	"address": "E 23...	"address": "E 23...	"address": "E 23...
5	SB00502QDkWcL...	the-diner-2lando...	@ The Diner	https://s...	0	https://www.yelp.co...	["diner","breakfast"]	"title": "Breakfast Brunch" ...	4.5	28.429...	-81.44...	"delivery": ..., "address": "W 9...	"address": "W 9...	"address": "W 9...	"address": "W 9...	"address": "W 9...
6	4x5AtP...	mediterranean-del-mando...	Mediterranean Dell	https://s...	0	https://www.yelp.co...	["greek","italian"]	"title": "Greek" ...	4.5	28.598...	-81.39...	"delivery": ..., "address": "W 91...	"address": "W 91...	"address": "W 91...	"address": "W 91...	"address": "W 91...
7	wLYn7zQXWtN6...	anise-polish-del-round-lake...	Anise's Polish Deli	https://s...	0	https://www.yelp.co...	["polish","title": "Deli"]	"title": "Deli" ...	4.5	42.381...	-88.0...	"delivery": ..., "address": "W 92...	"address": "W 92...	"address": "W 92...	"address": "W 92...	"address": "W 92...
8	6xExIHoGqj...	kalbiyan-round-the-beach...	Kalbiyan	https://s...	0	https://www.yelp.co...	["kalbi","kebab"]	"title": "Kalbiyan" ...	4.5	42.381...	-88.0...	"pickup": ..., "address": "W 93...	"address": "W 93...	"address": "W 93...	"address": "W 93...	"address": "W 93...
9	enhu3SgjTCQm...	upstate-craft-beer-and-ster...	Upstate Craft Beer &...	https://s...	0	https://www.yelp.co...	["beer","brewery"]	"title": "Upstate Craft Beer" ...	4.8	42.384...	-73.38...	"delivery": ..., "address": "W 94...	"address": "W 94...	"address": "W 94...	"address": "W 94...	"address": "W 94...
10	HzRZ...	takeo-oreando	Tako Oreano	https://s...	0	https://www.yelp.co...	["japanese"]	"title": "Japanese" ...	1634	40.1634...	-123.443...	"pickup": ..., "address": "W 95...	"address": "W 95...	"address": "W 95...	"address": "W 95...	"address": "W 95...
11	gxwvWbU-CHETK...	faces-el-norte-round-land-e...	Tacos El Norte	https://s...	0	https://www.yelp.co...	["mexican"]	"title": "Mexican" ...	4.0	42.382...	-88.0...	"delivery": ..., "address": "W 96...	"address": "W 96...	"address": "W 96...	"address": "W 96...	"address": "W 96...
12	TJLJ88Y81YKXM...	oasis-micro-pub-rockford	Oasis Micro Pub	https://s...	0	https://www.yelp.co...	["pub"]	"title": "Pub" ...	4.8	42.266...	-89.0...	"pickup": ..., "address": "W 97...	"address": "W 97...	"address": "W 97...	"address": "W 97...	"address": "W 97...
13	SNYYD7W...	bento-be-halibet...	Bento Cafe	https://s...	0	https://www.yelp.co...	["japanese"]	"title": "Japanese" ...	4.0	42.3445...	-88.0...	"delivery": ..., "address": "W 98...	"address": "W 98...	"address": "W 98...	"address": "W 98...	"address": "W 98...
14	hov72mzuFZ9...	3-amigos-lake-villa-3	3 Amigo's	https://s...	0	https://www.yelp.co...	["mexican"]	"title": "Mexican" ...	96	42.414...	-88.0...	"delivery": ..., "address": "W 99...	"address": "W 99...	"address": "W 99...	"address": "W 99...	"address": "W 99...
15	BHB3JfJ...	grand-central-terminal-new-y...	Grand Central Terminal	https://s...	0	https://www.yelp.co...	["american"]	"title": "American" ...	4.5	40.792...	-73.97...	"address": "N...	"address": "N...	"address": "N...	"address": "N...	"address": "N...
16	ShfT4Akz2u...	cafe-tu-tango-tango-lando...	Cafe Tu Tu Tango	https://s...	0	https://www.yelp.co...	["mexican"]	"title": "Mexican" ...	3303	48.2440...	-81.47...	"delivery": ..., "address": "W 100...	"address": "W 100...	"address": "W 100...	"address": "W 100...	"address": "W 100...
17	JtDRNctY91Y9...	mrs-vs-restaurant-round-land-e...	Mrs V's Restaurant	https://s...	0	https://www.yelp.co...	["italian"]	"title": "Italian" ...	68	42.380...	-88.0...	"delivery": ..., "address": "W 101...	"address": "W 101...	"address": "W 101...	"address": "W 101...	"address": "W 101...
18	N...	standers-pizza-round-land-r...	Olando's Pizza Round Lake	https://s...	0	https://www.yelp.co...	["italian"]	"title": "Italian" ...	164	42.378...	-88.10...	"delivery": ..., "address": "W 102...	"address": "W 102...	"address": "W 102...	"address": "W 102...	"address": "W 102...

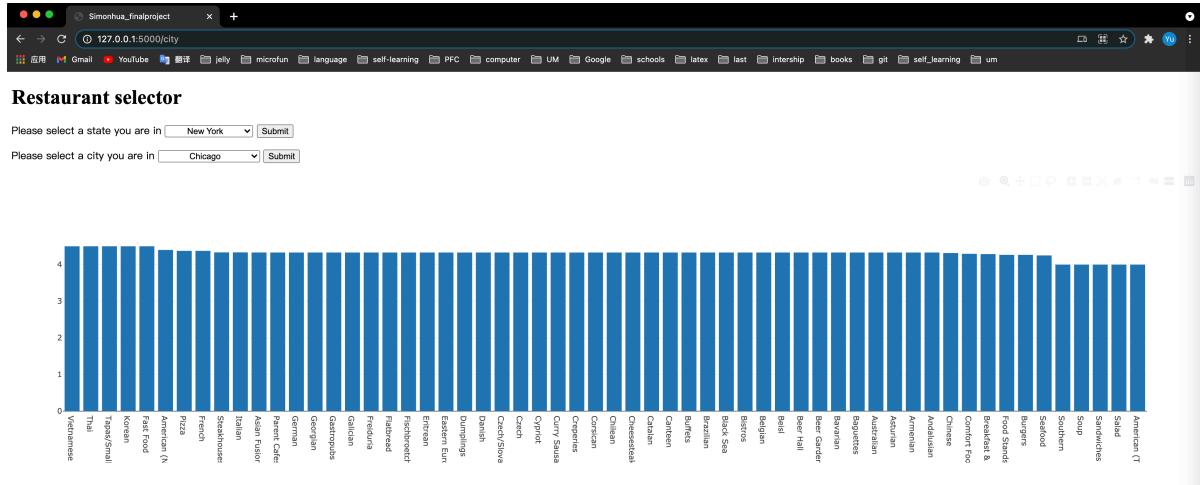
- Table 5 : restaurant_category_fetch

id	category	city
1	sd6p400K97wDxDjg3QOA	Andaluzian
2	vouduj3t1Hd4Q56TE0	Andaluzian
3	4kX70w1C1NpGmWAn4dzQ	Andaluzian
4	g2vawBwUCh6tKtwuHw0	Andaluzian
5	80ggyTufwdr4Yg3k3tQG	Andaluzian
6	3p9Dp2DxpDr-MANQFU	Andaluzian
7	7nAglp-Tflq5n6ZPZfW	Andaluzian
8	wOLzTwC1zBwBMyJtH4	Andaluzian
9	N-jmpzz2zW5DcoQd9u8	Andaluzian
10	szw_LwkvD0INW-2Ma3sq3	Andaluzian
11	JVUokW4Xp0gjB2ymergg	Andaluzian
12	JtDRNctY91Y9...	Andaluzian
13	6ye8tkZgv4uNdc2PwzQ	Andaluzian
14	wlyDrnZpBXNzfH-Me-SQ	Andaluzian
15	J0BNYzv2L2qDOW4p3yA	Andaluzian
16	3vRhiRfRppze-iQ2g	Andaluzian
17	G7z6uAfc4Dn7Dv15sWeEc	Andaluzian
18	3y8e8tkZgv4uNdc2PwzQ	Andaluzian

Interaction and Presentation Plans

Description

A program that lets a user choose a specific state and a city and see the average ratings for different restaurant types from Yelp as **plotly bar**. You can see it from the pic below. However, This is only a draft version and not the final version. You can select a state at first and submit, then a city and submit. The program will generate the bar plot by the state and the city you provided.



Then, you can choose a specific type of food, the program will present a useful statistical information about the covid-19 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.

The program will also provide a **plot of map** to facilitate you to choose a location closer to you. I intend to generate a map information from the address of each restaurant, so that users can roughly get the location of each restaurant. Specific information can be viewed by entering the detailed information of the restaurant.

Finally, you can choose one restaurant specifically, the program will offer you more details in **text**. This part will contain as much information as I can provide.

Technologies

1. **Flask** : The whole program is running within a Flask App.
2. **Plotly** : I used plotly to draw bar plot. There are some statistic data for covid 19 I can use.
3. **Command line** : Only for logging, you can see what the flask app is doing from the command line.