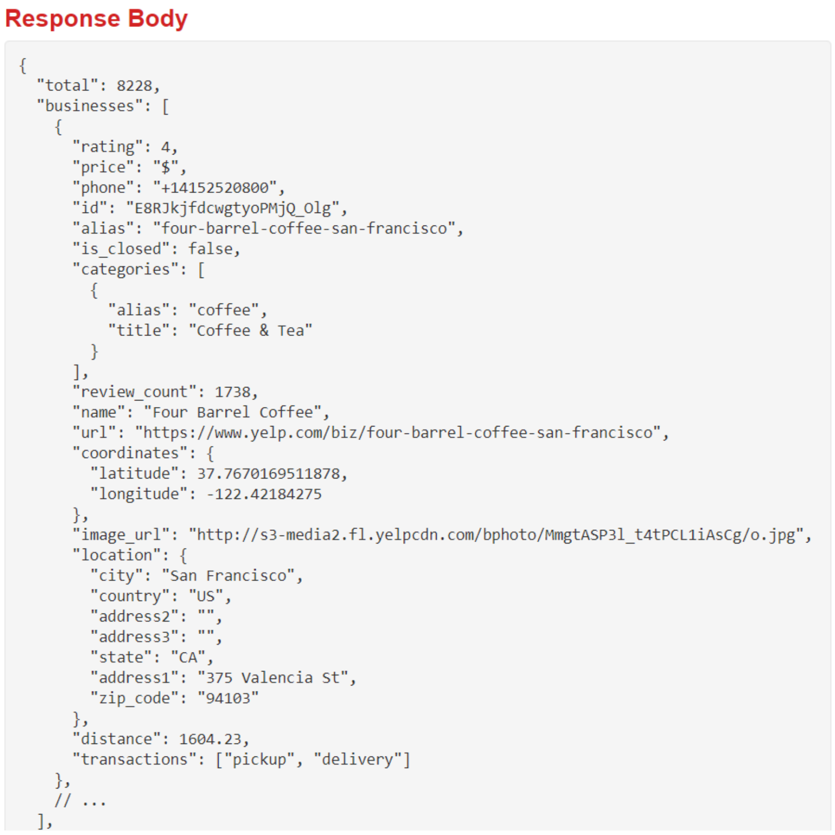
Project code

1. Link to github repo: https://github.com/zsx102/si507-final
2. Required Python packages: requests, flask, decimal, sqlite3, time, bs4

Data sources

1. The URL of the web page I scraped: https://www.latlong.net/category/colleges-236-35.html
2. The API I used: https://www.yelp.com/developers/documentation/v3/business_search

   The response is like below:



Response Body

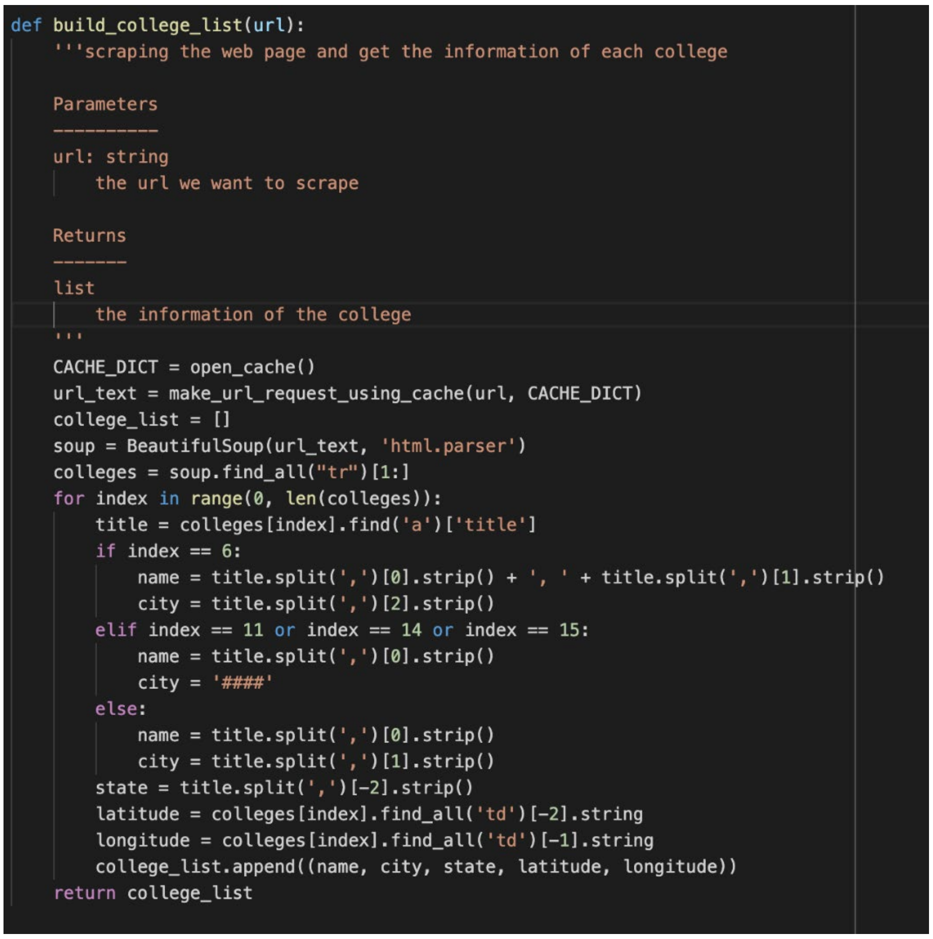
```
{
  "total": 8228,
  "businesses": [
    {
      "rating": 4,
      "price": "$",
      "phone": "+14152520800",
      "id": "E8RJkjfdcwgtyoPMjQ_Olg",
      "alias": "four-barrel-coffee-san-francisco",
      "is_closed": false,
      "categories": [
        {
          "alias": "coffee",
          "title": "Coffee & Tea"
        }
      ],
      "review_count": 1738,
      "name": "Four Barrel Coffee",
      "url": "https://www.yelp.com/biz/four-barrel-coffee-san-francisco",
      "coordinates": {
        "latitude": 37.7670169511878,
        "longitude": -122.42184275
      },
      "image_url": "http://s3-media2.fl.yelpcdn.com/bphoto/MmgtASP3l_t4tPCL1iAsCg/o.jpg",
      "location": {
        "city": "San Francisco",
        "country": "US",
        "address2": "",
        "address3": "",
        "state": "CA",
        "address1": "375 Valencia St",
        "zip_code": "94103"
      },
      "distance": 1604.23,
      "transactions": ["pickup", "delivery"]
    },
    // ...
  ],
```

   The ‘businesses’ part is what we wanted.

3. Brief description:

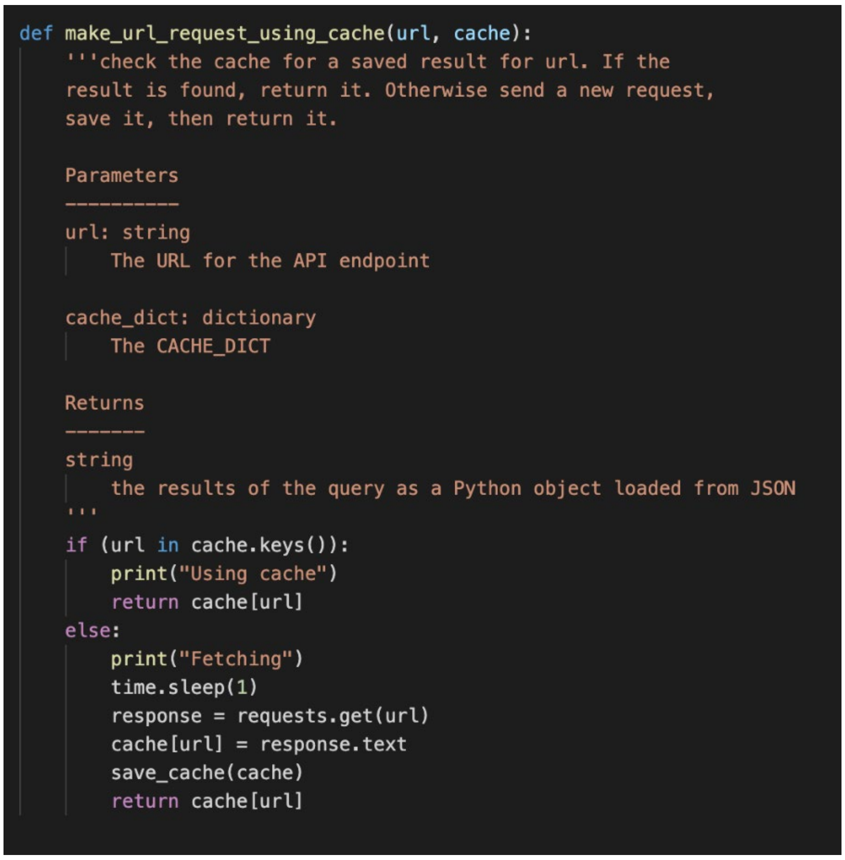
   I use the BeautifulSoup aligned with cache to scrape the data on the web page.

   a) Scrape part:

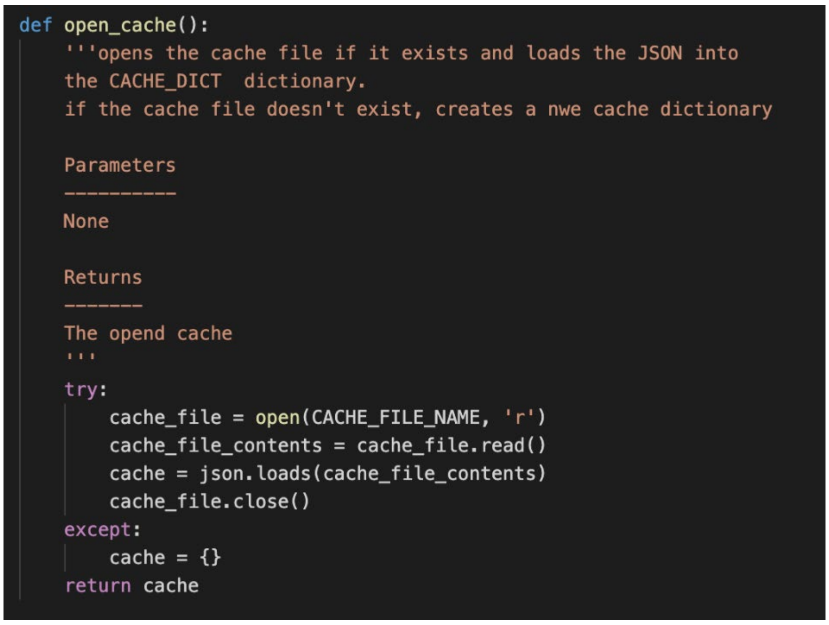


```
def build_college_list(url):
    '''scraping the web page and get the information of each college

    Parameters
    ----------
    url: string
        the url we want to scrape

    Returns
    -------
    list
        the information of the college
    '''
    CACHE_DICT = open_cache()
    url_text = make_url_request_using_cache(url, CACHE_DICT)
    college_list = []
    soup = BeautifulSoup(url_text, 'html.parser')
    colleges = soup.find_all("tr")[1:]
    for index in range(0, len(colleges)):
        title = colleges[index].find('a')['title']
        if index == 6:
            name = title.split(',')[0].strip() + ', ' + title.split(',')[1].strip()
            city = title.split(',')[2].strip()
        elif index == 11 or index == 14 or index == 15:
            name = title.split(',')[0].strip()
            city = '####'
        else:
            name = title.split(',')[0].strip()
            city = title.split(',')[1].strip()
        state = title.split(',')[-2].strip()
        latitude = colleges[index].find_all('td')[-2].string
        longitude = colleges[index].find_all('td')[-1].string
        college_list.append((name, city, state, latitude, longitude))
    return college_list
```

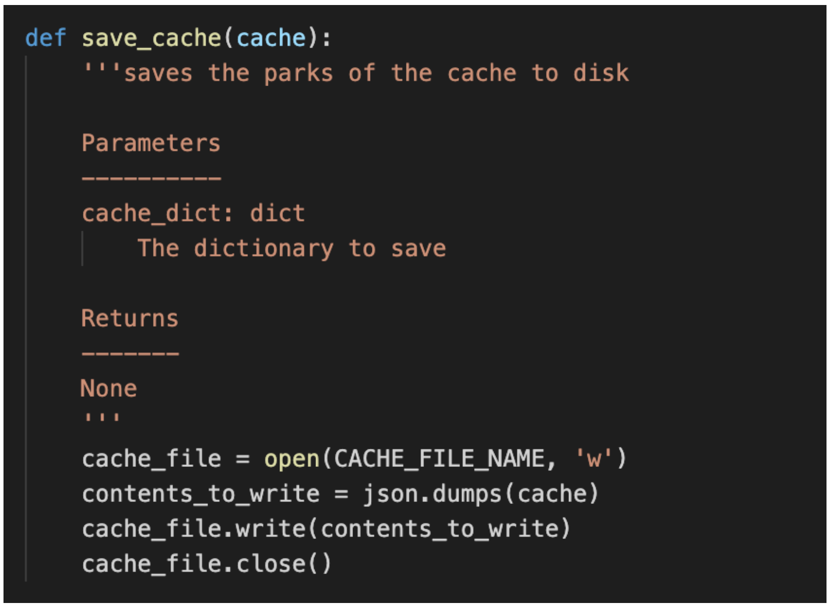
b) Cache part:



```
def make_url_request_using_cache(url, cache):
    '''check the cache for a saved result for url. If the
    result is found, return it. Otherwise send a new request,
    save it, then return it.

    Parameters
    ----------
    url: string
        The URL for the API endpoint

    cache_dict: dictionary
        The CACHE_DICT

    Returns
    -------
    string
        the results of the query as a Python object loaded from JSON
    '''
    if (url in cache.keys()):
        print("Using cache")
        return cache[url]
    else:
        print("Fetching")
        time.sleep(1)
        response = requests.get(url)
        cache[url] = response.text
        save_cache(cache)
        return cache[url]
```

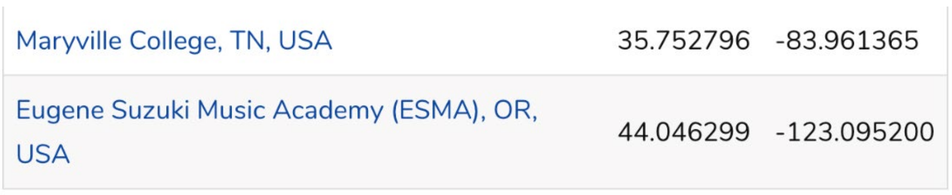


```python
def open_cache():
    '''opens the cache file if it exists and loads the JSON into
    the CACHE_DICT  dictionary.
    if the cache file doesn't exist, creates a nwe cache dictionary

    Parameters
    ----------
    None

    Returns
    -------
    The opend cache
    '''
    try:
        cache_file = open(CACHE_FILE_NAME, 'r')
        cache_file_contents = cache_file.read()
        cache = json.loads(cache_file_contents)
        cache_file.close()
    except:
        cache = {}
    return cache
```



```python
def save_cache(cache):
    '''saves the parks of the cache to disk

    Parameters
    ----------
    cache_dict: dict
        The dictionary to save

    Returns
    -------
    None
    '''
    cache_file = open(CACHE_FILE_NAME, 'w')
    contents_to_write = json.dumps(cache)
    cache_file.write(contents_to_write)
    cache_file.close()
```

Notice that some colleges miss some information like below:



Maryville College, TN, USA	35.752796	-83.961365
Eugene Suzuki Music Academy (ESMA), OR, USA	44.046299	-123.095200

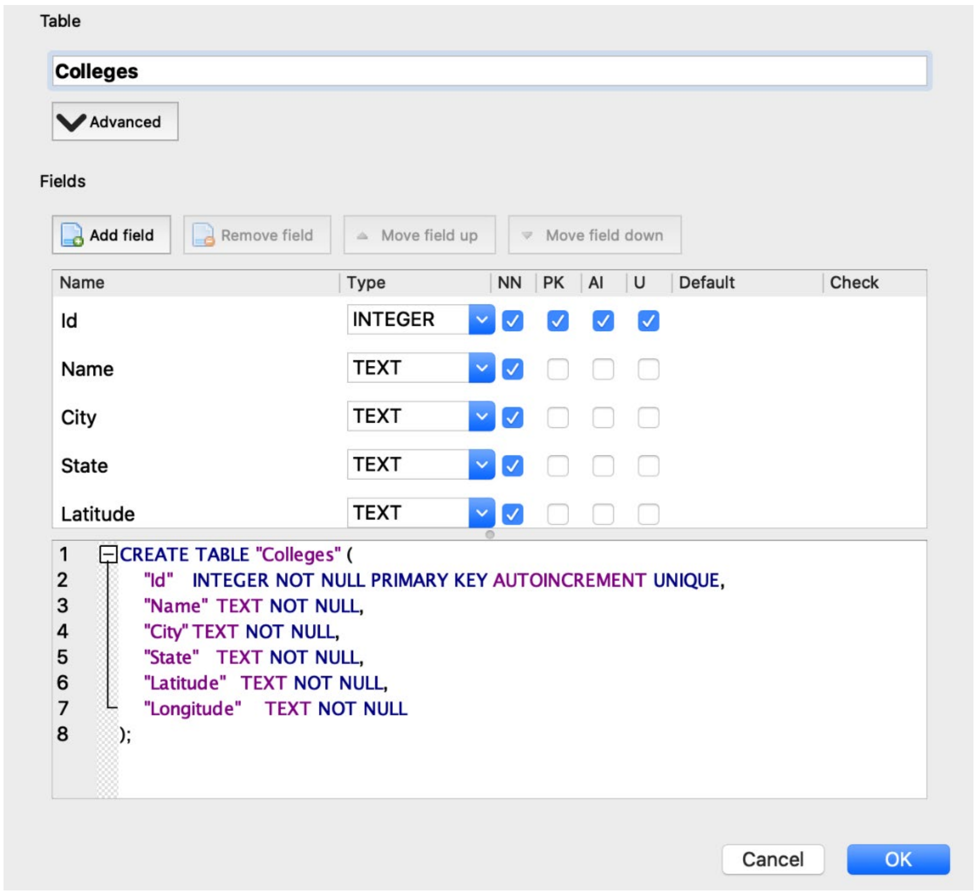
Those two miss the city field, all the missing part I use '####' to represent.

4. Summary of data:
   The data sources have five important fields: college name, college city, college state, latitude and longitude.

Database

1. Database schema:

a) College Table:



Table

Colleges

Advanced

Fields

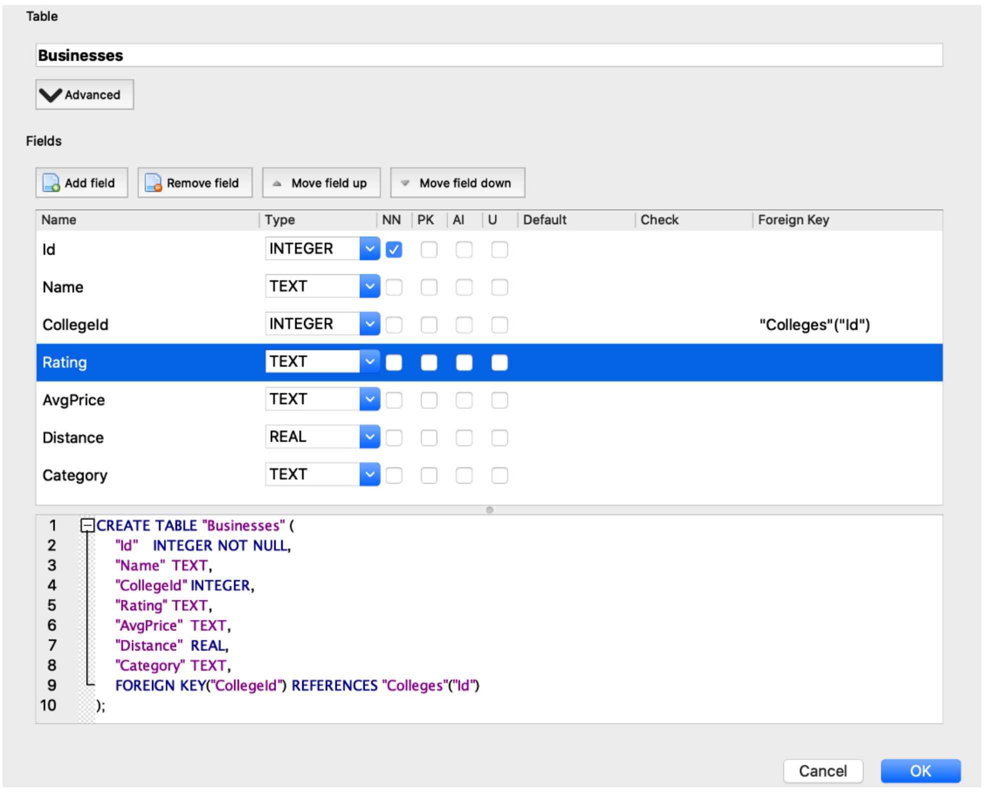
Add field | Remove field | Move field up | Move field down

Name	Type	NN	PK	AI	U	Default	Check
Id	INTEGER	✓	✓	✓	✓		
Name	TEXT	✓					
City	TEXT	✓					
State	TEXT	✓					
Latitude	TEXT	✓					

```
CREATE TABLE "Colleges" (
    "Id"   INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "Name"  TEXT NOT NULL,
    "City" TEXT NOT NULL,
    "State"   TEXT NOT NULL,
    "Latitude"   TEXT NOT NULL,
    "Longitude"    TEXT NOT NULL
);
```

Cancel OK

b) Business Table:



Table

Businesses

Advanced

Fields

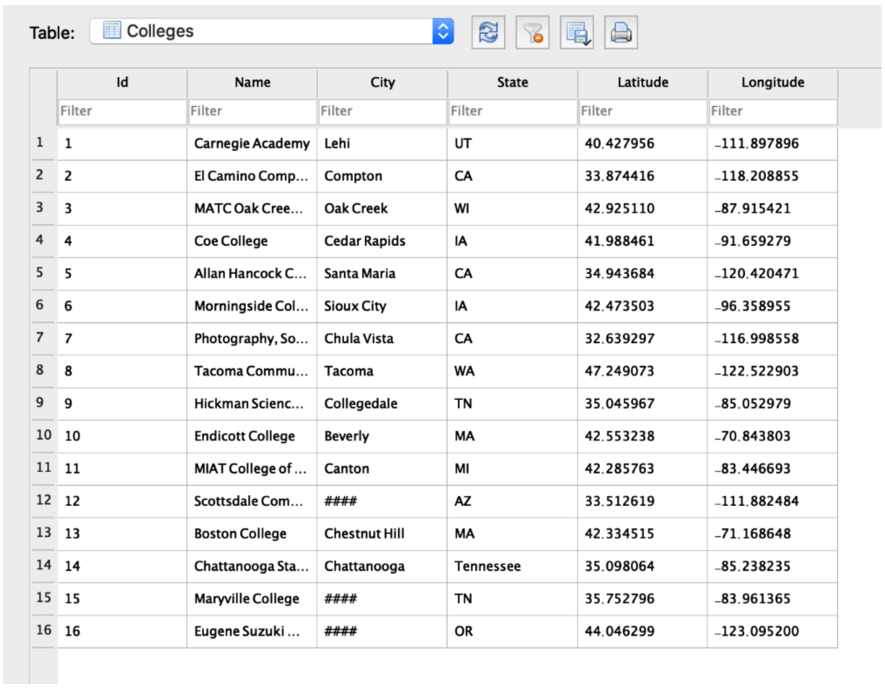
Add field | Remove field | Move field up | Move field down

Name	Type	NN	PK	AI	U	Default	Check	Foreign Key
Id	INTEGER	✓						
Name	TEXT							
CollegeId	INTEGER							"Colleges"("Id")
Rating	TEXT							
AvgPrice	TEXT							
Distance	REAL							
Category	TEXT							

```
CREATE TABLE "Businesses" (
    "Id"   INTEGER NOT NULL,
    "Name"  TEXT,
    "CollegeId" INTEGER,
    "Rating" TEXT,
    "AvgPrice"  TEXT,
    "Distance"  REAL,
    "Category" TEXT,
    FOREIGN KEY("CollegeId") REFERENCES "Colleges"("Id")
);
```

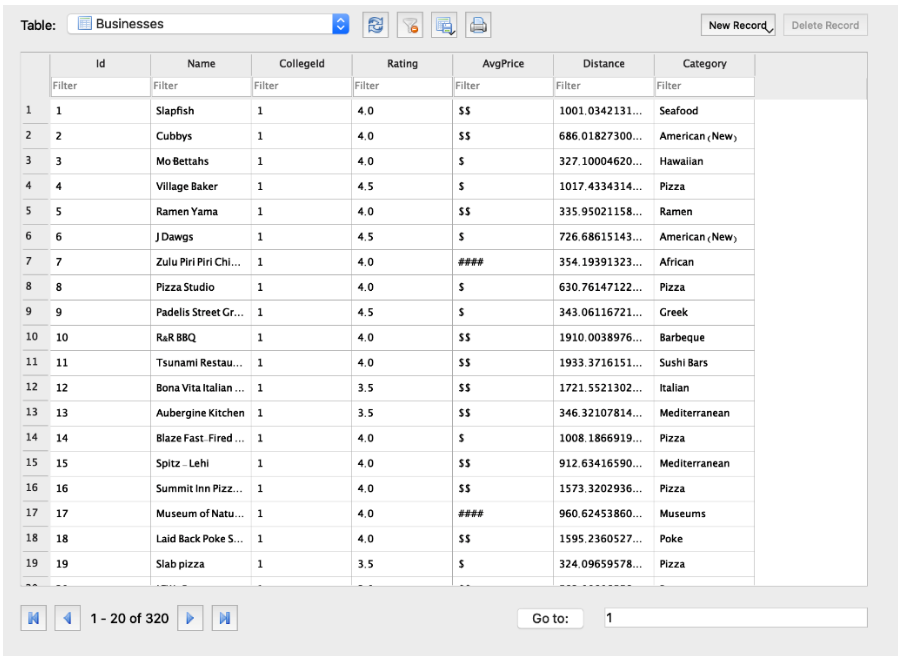
Cancel OK

2. Foreign key – Primary key relation:
   The primary key in college table is the foreign key in businesses table (CollegeId).
3. Screenshots:
   a) College Table:



	Id	Name	City	State	Latitude	Longitude
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Carnegie Academy	Lehi	UT	40.427956	-111.897896
2	2	El Camino Comp...	Compton	CA	33.874416	-118.208855
3	3	MATC Oak Cree...	Oak Creek	WI	42.925110	-87.915421
4	4	Coe College	Cedar Rapids	IA	41.988461	-91.659279
5	5	Allan Hancock C...	Santa Maria	CA	34.943684	-120.420471
6	6	Morningside Col...	Sioux City	IA	42.473503	-96.358955
7	7	Photography, So...	Chula Vista	CA	32.639297	-116.998558
8	8	Tacoma Commu...	Tacoma	WA	47.249073	-122.522903
9	9	Hickman Scienc...	Collegedale	TN	35.045967	-85.052979
10	10	Endicott College	Beverly	MA	42.553238	-70.843803
11	11	MIAT College of ...	Canton	MI	42.285763	-83.446693
12	12	Scottsdale Com...	####	AZ	33.512619	-111.882484
13	13	Boston College	Chestnut Hill	MA	42.334515	-71.168648
14	14	Chattanooga Sta...	Chattanooga	Tennessee	35.098064	-85.238235
15	15	Maryville College	####	TN	35.752796	-83.961365
16	16	Eugene Suzuki ...	####	OR	44.046299	-123.095200

   b) Business Table:

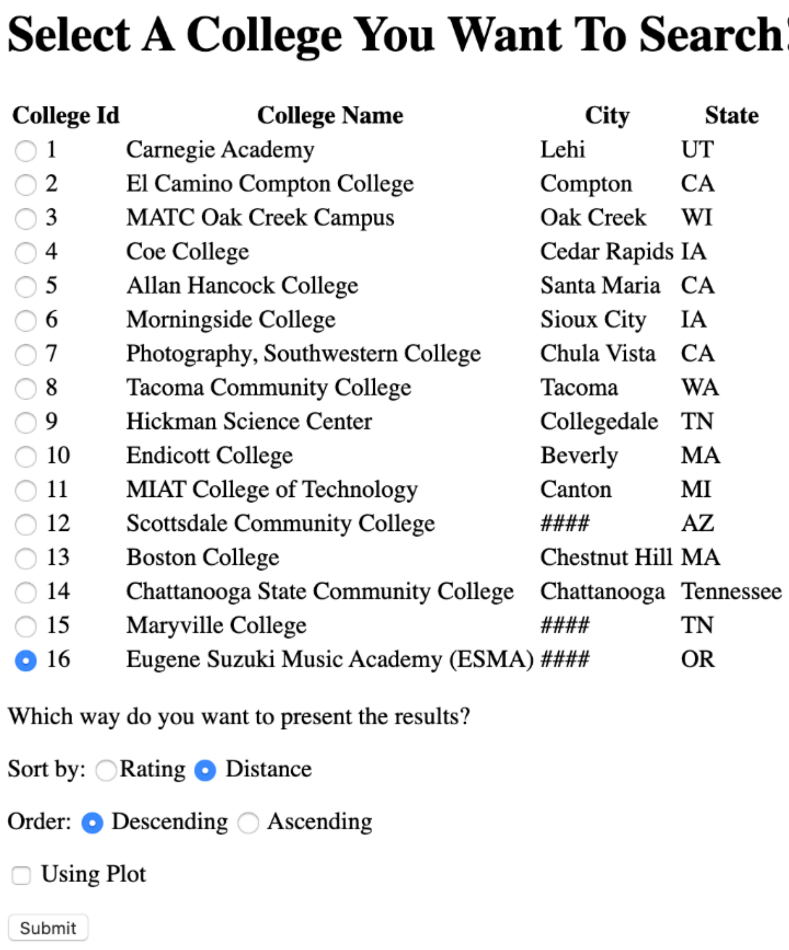


	Id	Name	CollegeId	Rating	AvgPrice	Distance	Category
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Slapfish	1	4.0	$$	1001.0342131...	Seafood
2	2	Cubbys	1	4.0	$$	686.01827300...	American (New)
3	3	Mo Bettahs	1	4.0	$	327.10004620...	Hawaiian
4	4	Village Baker	1	4.5	$	1017.4334314...	Pizza
5	5	Ramen Yama	1	4.0	$$	335.95021158...	Ramen
6	6	J Dawgs	1	4.5	$	726.68615143...	American (New)
7	7	Zulu Piri Piri Chi...	1	4.0	####	354.19391323...	African
8	8	Pizza Studio	1	4.0	$	630.76147122...	Pizza
9	9	Padelis Street Gr...	1	4.5	$	343.06116721...	Greek
10	10	R&R BBQ	1	4.0	$$	1910.0038976...	Barbeque
11	11	Tsunami Restau...	1	4.0	$$	1933.3716151...	Sushi Bars
12	12	Bona Vita Italian ...	1	3.5	$$	1721.5521302...	Italian
13	13	Aubergine Kitchen	1	3.5	$$	346.32107814...	Mediterranean
14	14	Blaze Fast-Fired ...	1	4.0	$	1008.1866919...	Pizza
15	15	Spitz - Lehi	1	4.0	$$	912.63416590...	Mediterranean
16	16	Summit Inn Pizz...	1	4.0	$$	1573.3202936...	Pizza
17	17	Museum of Natu...	1	4.0	####	960.62453860...	Museums
18	18	Laid Back Poke S...	1	4.0	$$	1595.2360527...	Poke
19	19	Slab pizza	1	3.5	$	324.09659578...	Pizza

Interaction and Presentation Options

1. Description:
   This project scrapes the information of colleges on a web page and use the yelp api attached with latitude and longitude to search the nearby businesses.

2. Technologies:
   a) Using flask to create the front-end web pages and the router between each page.
   b) If user chooses to use plot, the data will be presented in a graph, plot is used.
3. Brief instruction:
   a) Select the radio buttons like below:



Select A College You Want To Search!

College Id	College Name	City	State
○ 1	Carnegie Academy	Lehi	UT
○ 2	El Camino Compton College	Compton	CA
○ 3	MATC Oak Creek Campus	Oak Creek	WI
○ 4	Coe College	Cedar Rapids	IA
○ 5	Allan Hancock College	Santa Maria	CA
○ 6	Morningside College	Sioux City	IA
○ 7	Photography, Southwestern College	Chula Vista	CA
○ 8	Tacoma Community College	Tacoma	WA
○ 9	Hickman Science Center	Collegedale	TN
○ 10	Endicott College	Beverly	MA
○ 11	MIAT College of Technology	Canton	MI
○ 12	Scottsdale Community College	####	AZ
○ 13	Boston College	Chestnut Hill	MA
○ 14	Chattanooga State Community College	Chattanooga	Tennessee
○ 15	Maryville College	####	TN
[✓ 16]	Eugene Suzuki Music Academy (ESMA)	####	OR

Which way do you want to present the results?

Sort by: Rating [✓ Distance]

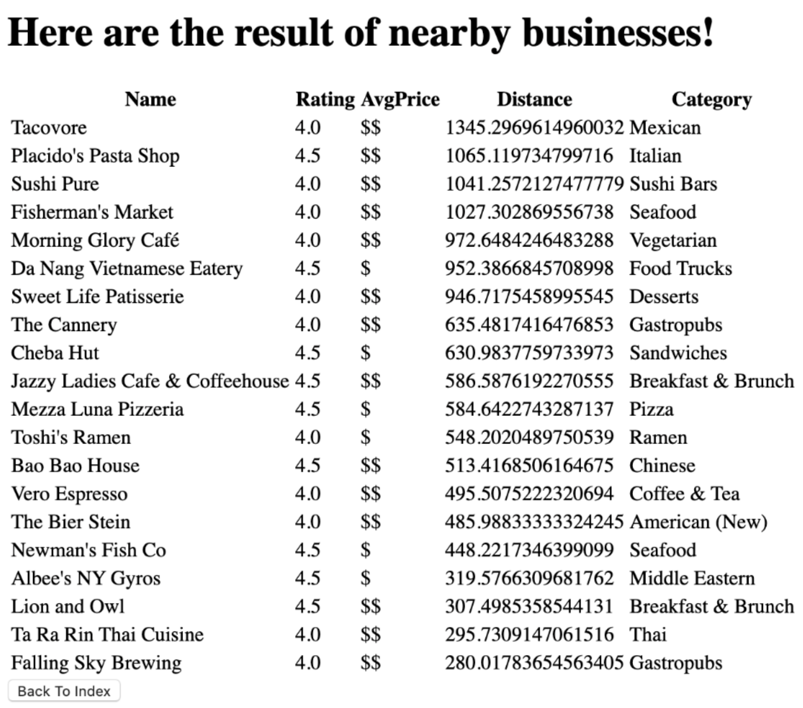
Order: [✓ Descending] Ascending

☐ Using Plot

Submit

   Then press the submit button.

   b) If you don’t use the plot, the result will be presented as below:



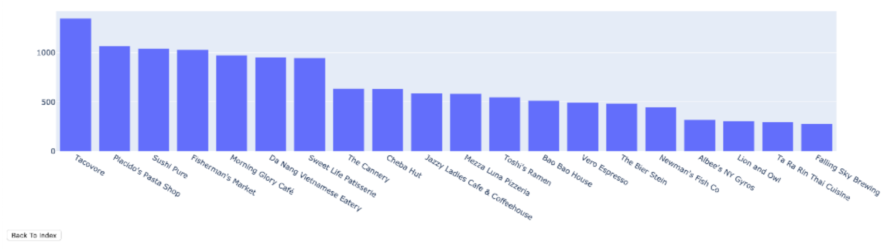
Here are the result of nearby businesses!

Name	Rating	AvgPrice	Distance	Category
Tacovore	4.0	$$	1345.2969614960032	Mexican
Placido's Pasta Shop	4.5	$$	1065.119734799716	Italian
Sushi Pure	4.0	$$	1041.2572127477779	Sushi Bars
Fisherman's Market	4.0	$$	1027.302869556738	Seafood
Morning Glory Café	4.0	$$	972.6484246483288	Vegetarian
Da Nang Vietnamese Eatery	4.5	$	952.3866845708998	Food Trucks
Sweet Life Patisserie	4.0	$$	946.7175458995545	Desserts
The Cannery	4.0	$$	635.4817416476853	Gastropubs
Cheba Hut	4.5	$	630.9837759733973	Sandwiches
Jazzy Ladies Cafe & Coffeehouse	4.5	$$	586.5876192270555	Breakfast & Brunch
Mezza Luna Pizzeria	4.5	$	584.6422743287137	Pizza
Toshi's Ramen	4.0	$	548.2020489750539	Ramen
Bao Bao House	4.5	$$	513.4168506164675	Chinese
Vero Espresso	4.0	$$	495.5075222320694	Coffee & Tea
The Bier Stein	4.0	$$	485.98833333324245	American (New)
Newman's Fish Co	4.5	$	448.2217346399099	Seafood
Albee's NY Gyros	4.5	$	319.5766309681762	Middle Eastern
Lion and Owl	4.5	$$	307.4985358544131	Breakfast & Brunch
Ta Ra Rin Thai Cuisine	4.0	$$	295.7309147061516	Thai
Falling Sky Brewing	4.0	$$	280.01783654563405	Gastropubs

Back To Index

c) If you use the plot, the result will be presented as below:

Here is the plot!



d) Press 'Back To Index' button to return to the index.

Demo Link

https://www.loom.com/share/55760b1b77044f5ea22d95891d431202

PS: It seems my microphone was broken, so I just attached the statement here to explain what I am doing in the video.

First, I show my resource page, and list the fields I record, college name, city, state, latitude and longitude. Notice that some miss the city field, and I use '####' to represent the missing information. After that I use the yelp api to search the nearby businesses of the colleges. The process to get the api key is the same as iTunes api we learned at class. And the params are latitude and longitude which presented in the table. (location is not provided) And check the reponse body we find that "businesses" part is what we wanted and I choose some information to record, rating, price, title, name and distance. So they are the fields in Businesses Table. Since

there are 16 colleges, each college has 20 nearby businesses, so the number of records at Businesses Table is 320.
Then, the user can choose one college to search and the user can also decide the way the result presented. (sorted by rating or distance, descending or ascending, using plot or not).
That’s all I’ve done in the vedio. Thank you.