HW8: Database Joins & Matplotlib

In this homework, you will select data from a database, process it, and create a visualization using Matplotlib.

We have provided a data source and a starter code file:

- ***restaurants_south_u.db*** - a database with local restaurant data.
- **HW8_start.py** - starter code for the functions below.

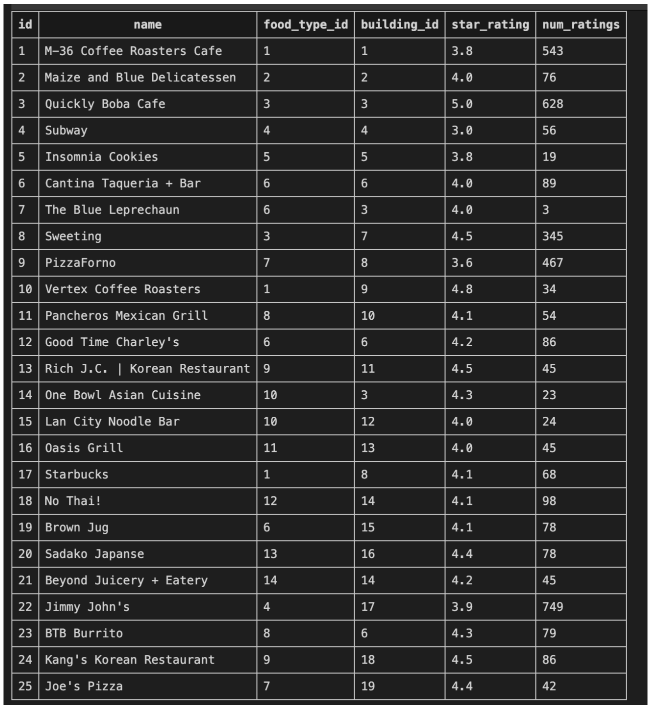
Make sure you are using Anaconda ("base":conda) python for this assignment (preferred), which already has `matplotlib`. But if you have to install `matplotlib` on your own, you can use `pip3 install matplotlib` or `pip install matplotlib` or another installation method. We have also provided test cases that will pass if the functions are written correctly. You should not edit these test cases.

Note: It is okay for the extra credit test case to fail if you do not attempt the extra credit; you can also comment out those specific test cases.

Before you start: Look at the database

Check out *restaurants_south_u.db* in your DB Browser for SQLite program.

1. Open DB Browser for SQLite
2. Click on "Open Database" and choose restaurants_south_u*.db*.
3. Click on Browse Data
4. Take some time to familiarize yourself with the table and column names.



id	name	food_type_id	building_id	star_rating	num_ratings
1	M-36 Coffee Roasters Cafe	1	1	3.8	543
2	Maize and Blue Delicatessen	2	2	4.0	76
3	Quickly Boba Cafe	3	3	5.0	628
4	Subway	4	4	3.0	56
5	Insomnia Cookies	5	5	3.8	19
6	Cantina Taqueria + Bar	6	6	4.0	89
7	The Blue Leprechaun	6	3	4.0	3
8	Sweeting	3	7	4.5	345
9	PizzaForno	7	8	3.6	467
10	Vertex Coffee Roasters	1	9	4.8	34
11	Pancheros Mexican Grill	8	10	4.1	54
12	Good Time Charley's	6	6	4.2	86
13	Rich J.C. | Korean Restaurant	9	11	4.5	45
14	One Bowl Asian Cuisine	10	3	4.3	23
15	Lan City Noodle Bar	10	12	4.0	24
16	Oasis Grill	11	13	4.0	45
17	Starbucks	1	8	4.1	68
18	No Thai!	12	14	4.1	98
19	Brown Jug	6	15	4.1	78
20	Sadako Japanse	13	16	4.4	78
21	Beyond Juicery + Eatery	14	14	4.2	45
22	Jimmy John's	4	17	3.9	749
23	BTB Burrito	8	6	4.3	79
24	Kang's Korean Restaurant	9	18	4.5	86
25	Joe's Pizza	7	19	4.4	42

Part 1: Process the data

Complete the ***restaurant_data_loader(db)*** function that accepts the filename of the database as a parameter, and returns a nested dictionary.
Each outer key of the dictionary is the name of each restaurant in the database, and the value is a dictionary where the inner keys are *food_type*, *building_number*, *star_rating*, and *num_ratings* for the restaurant, the inner values are the corresponding information extracted from *restaurants_south_u.db*. Your function must pass all the unit tests to get full credit.
Note: Because all the restaurants are on the same street (in this case, South University Ave), the addresses only contain the *building_numbers* as an integer.

Expected output:

```
{'M-36 Coffee Roasters Cafe':
        {'food_type: 'Cafe',
        'building_number': 1101,
        'star_rating': 3.8,
        'num_ratings': 543},
         . . . }
```

Part 2: Visualize data: Find the restaurants with low *num_ratings* by *food_type*

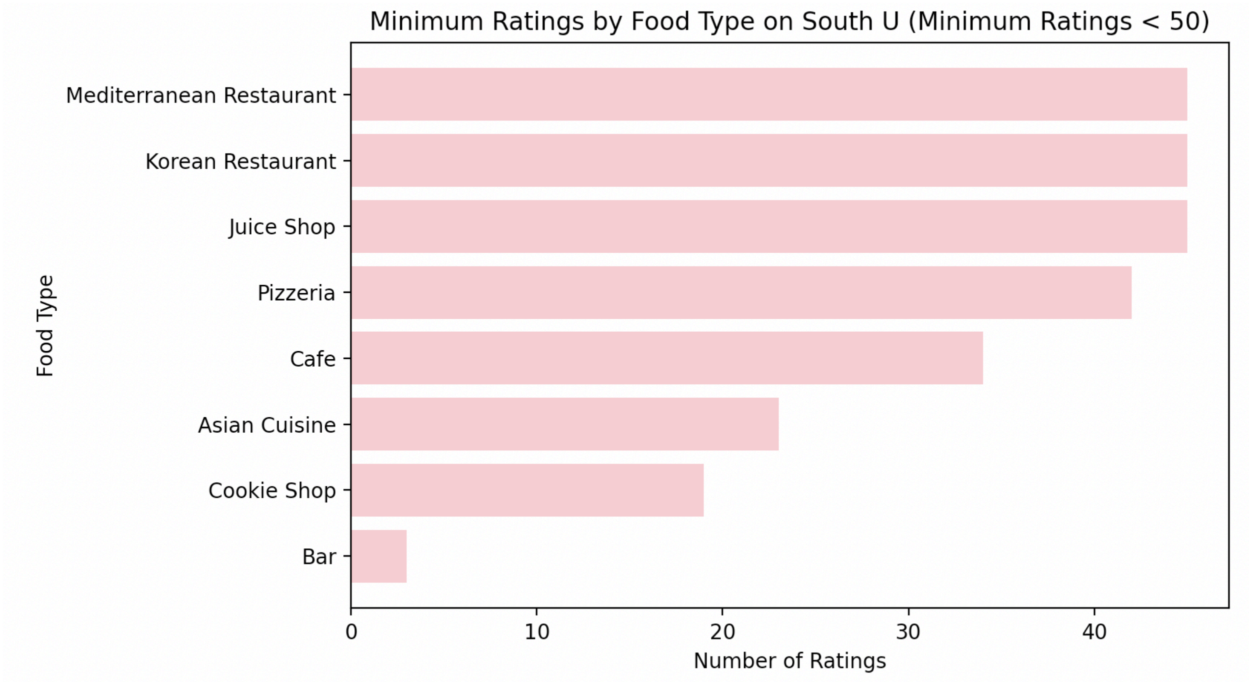
Complete the function ***plot_low*****_num_rating_*****by_type(db)***, which accepts the filename of the database as a parameter and returns a dictionary. Only the restaurants with fewer than 50 ratings are included when finding the minimum *num_ratings* for each food_type. It returns a dictionary where the keys are food types, and the values are the minimum number of ratings among the restaurants in that food type. (**hint:** use the SQL MIN keyword).

Expected return value:

{'Bar': 3, 'Cookie Shop': 19, 'Asian Cuisine': 23, 'Cafe': 34, 'Pizzeria': 42, 'Juice Shop': 45, 'Korean Restaurant': 45, 'Mediterranean Restaurant': 45}

The function should also create a bar chart (horizontal or vertical – figure out which one gives a better visualization) with restaurant *food_types* along one axis and the corresponding minimum *num_ratings* along the other axis. In the chart, the minimum *num_ratings* should be in ***descending*** order.

Example chart:



Submit an image file of your bar chart to Canvas, along with your repository link.

Part 3: Find restaurants with a specified star rating

Complete the function ***find_restaurant_with_star***(***star_rating, db)***, which accepts the *star_rating* and the filename of the database as parameters and returns a list with the restaurants of the same *star_rating in db*. The restaurants should be sorted by their *building_number* from largest to smallest (**hint:** Use the SQL WHERE keyword).

For example, for star_rating 3.8, the expected return value is:

['Insomnia Cookies', 'M-36 Coffee Roasters Cafe']

The *building_number* for 'Insomnia Cookies' is **1229**, and the *building_number* for 'M-36 Coffee Roasters Cafe' is **1101**

Extra Credit: Visualize more data

Finish the ***get_highest_weighted_average_ratings(db)*** function to determine which *food_type* and *building_number* have the highest weighted average *star_rating* for restaurants. For our calculations, the weight (w_i) of each restaurant's rating will be the number of ratings (*num_ratings*) that the restaurant received. **Hint: the weighted average can be calculated**

entirely in your SQL query, although this is not required. Some SQL functions you may find useful are SUM() and ROUND().

Formula

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}$$

W = weighted average

n = number of terms to be averaged

w_i = weights applied to x values

X_i = data values to be averaged

For an example of the weighted average, take the *food_type* “Bubble Tea Shop”. There are two restaurants of this *food_type*:

- Quickly Boba Cafe with a *star_rating* of 5.0 and a *num_ratings* of 628
- Sweeting with a *star_rating* of 4.5 and a *num_ratings* of 345

This means that the weighted average rating for the *food_type* “Bubble Tea Shop” is:

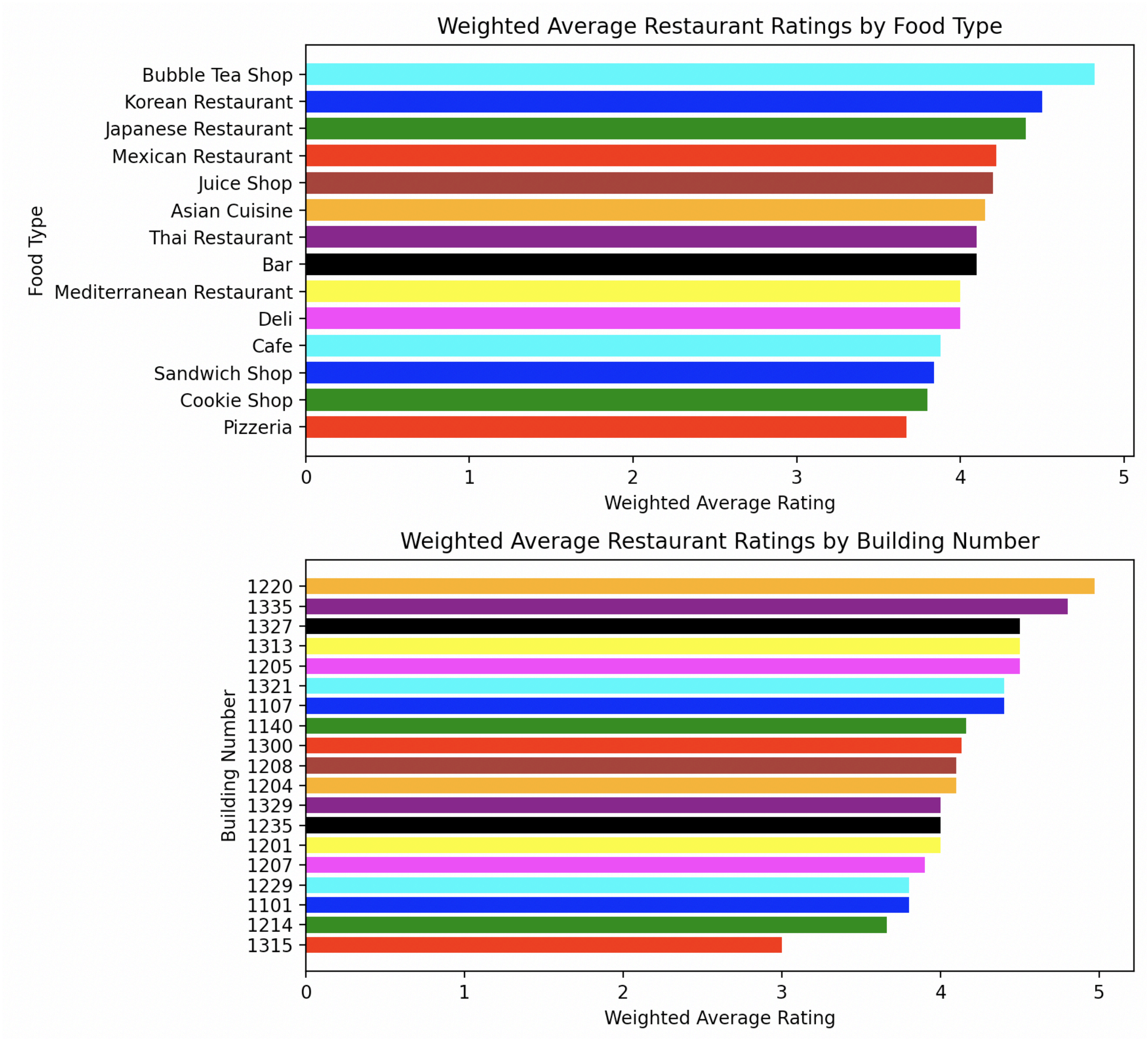
$$\frac{(5.0\times628)+(4.5\times345)}{(628)+(345)} = 4.82$$

Complete function ***get_highest_weighted_average_ratings(db)*** to plot two bar charts in one figure using `plt.subplot()`.

For the first bar chart, the y-axis will be the *food_type* of each restaurant. The x-axis will be the weighted average *star_rating* for the restaurants of each *food_type*. The average values should be rounded to two decimal places. Sort the y-axis in **descending order** from top-to-bottom by rating.

For the second bar chart, the y-axis will be different *building_numbers*. The x-axis will be the weighted average *star_rating* for the restaurants in each building. The average values should also be rounded to two decimal places, and the y-axis should be sorted in **descending order** by rating.

The chart must have appropriate axis labels and a title. The limit of the x-axis should be **0 - 5** for both charts. You can use `plt.figure(figsize=(8,8))` to adjust the size of the figure. Your chart should look like this:



Finally, this function should return a dictionary with two key-value pairs. The first key is the name of the highest-rated restaurant *food_type*, the value is its corresponding weighted average star rating; the second key is the highest-rated *building_number*, and the value is its corresponding weighted average star rating.

Expected Output:
{'Bubble Tea Shop': 4.82, 1220: 4.97}

Grading

load_restaurant_data(db)	10 pts

plot_best_star_ratings_by_food_type(db)	10 pts
find_restaurant_with_star(building_number, db)	15 pts
Submitted the bar chart image file	5 pts
Bars are ordered in descending order	5 pts
Title on the bar chart	5 pts
Informative X-axis label on bar chart	5 pts
Informative Y-axis label on bar chart	5 pts
get_highest_weighted_average_ratings(db) *Correct code and image file for extra credit*	*6 pts extra credit*
Total	**60 pts + 6 pts extra credit**