ETL Project – San Francisco Restaurants and Parking Meters

Junlin Chen/Jinghua Yao

Background

San Francisco, the second-most densely populated major American city, covers an area of about 46.89 square miles, with more than 358,000 household, 70% of which owns vehicle (about 1 car per household).
There are about 4,400 restaurants in San Francisco and about 275,450 street parking spaces citywide. Finding parking spots is always a challenge during dinning hours in San Francisco.

Business Need

We perform the collection, integration and transformation of large volumes of parking meters data from San Francisco Open Data to and 1,000 restaurants' data by calling yelp API. Our goal is to locate parking spots of each restaurant base on the restaurant's longitude and latitude, select the 10 closest ones to each specific coordinate set and store the data into a new table.

Steps.

- Parking Meter Data
  1. Data Extraction: We download 35,000 parking meters data set (CSV), randomly selected from total available 275,450 parking spaces from San Francisco Open Data.
  2. Data Cleaning: We get rid of all rows with invalid data, including the ones with blank cells, error, or the ones with different format, and duplicate ones. We select columns that are relevant to our project only.

	objectid	street_num	street_name	longitude	latitude	meter_type	point
0	197016	133	STEUART ST	-122.392658	37.792929	SS	(37.79292919936527, -122.39265829868762)
1	197018	331	EMBARCADERO NORTH	-122.396884	37.798065	SS	(37.79806519436348, -122.39688371089093)
2	197026	511	COMMERCIAL ST	-122.401983	37.794322	SS	(37.79432199832723, -122.4019826007808)
3	197034	225	SANSOME ST	-122.401163	37.792819	SS	(37.79281930205363, -122.40116260011312)
4	197043	403	SANSOME ST	-122.401437	37.794149	SS	(37.794148599417014, -122.40143680050025)

3. Data Transformation: After the data cleaning process, there are 32,865 rows and 7 columns left which later will be saved in a new parking meter data file.

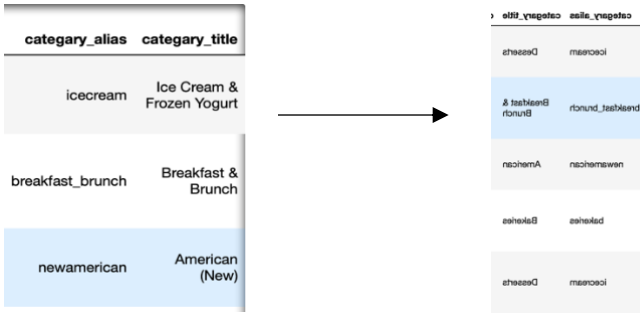
- Restaurant Data
  1. Data Extraction: We connect to Yelp API to obtain San Francisco restaurant data (JSON). By reading the API documentation, we understand that we are only allowed to query 50 business data per request with 20 requests limit. We extract 1,000 total restaurant data from Yelp API and then transform the date to pd.DataFrame.

	alias	categories	coordinates.latitude	coordinates.longitude	display_phone	distance
0	bi-rite-creamery-san-francisco	[{'alias': 'icecream', 'title': 'Ice Cream & F...	37.761591	-122.425717	(415) 626-5600	946.386739
1	brendas-french-soul-food-san-francisco	[{'alias': 'breakfast_brunch', 'title': 'Break...	37.782902	-122.419043	(415) 345-8100	2885.389131

  2. Data Cleaning

     (1) Extract the information in “categories’ column

categary_alias	categary_title
icecream	Ice Cream & Frozen Yogurt
breakfast_brunch	Breakfast & Brunch
newamerican	American (New)



     (2) Rename the categories.title to clean up repeated category

     Reduce 131 categories to 86

```
In [44]: type=df['category_title'].unique()
         type

Out[44]: array(['Ice Cream & Frozen Yogurt', 'Breakfast & Brunch',
                'American (New)', 'Bakeries', 'Seafood', 'American (Traditional)',
                'Burmese', 'Greek', 'Asian Fusion', 'Chinese', 'Mexican',
                'Vietnamese', 'Pizza', 'Bars', 'French', 'British',
                'Landmarks & Historical Buildings', 'Ramen', 'Parks',
                'Tapas/Small Plates', 'Steakhouses', 'Coffee & Tea', 'German',
                'Italian', 'Museums', 'Korean', 'Peruvian', 'Cocktail Bars',
                'Sushi Bars', 'Hiking', 'Grocery', 'Burgers', 'Desserts',
                'Hot Dogs', 'Thai', 'Sandwiches', 'Japanese', 'Caribbean',
                'Local Flavor', 'Mini Golf', 'Delis', 'Food Trucks', 'Vegan',
                'Southern', 'Filipino', 'Chicken Wings', 'Indian', 'Soul Food',
                'Dim Sum', 'Tea Rooms', 'Latin American', 'Spanish', 'Lounges',
                'Venues & Event Spaces', 'Sports Bars', 'Chicken Shop',
                'Mediterranean', 'Ethiopian', 'Patisserie/Cake Shop',
                'Middle Eastern', 'Gastropubs', 'Chocolatiers & Shops',
                'Candy Stores', 'Diners', 'Wine Bars', 'Himalayan/Nepalese',
                'Japanese Curry', 'Art Museums', 'Singaporean', 'Moroccan',
                'Botanical Gardens', 'Dive Bars', 'Falafel', 'Hot Pot',
                'Comfort Food', 'Beaches', 'Creperies', 'Bubble Tea', 'Gelato',
                'Food Stands', 'Themed Cafes', 'Vegetarian', 'Gluten-Free',
                'Cafes', 'Brasseries', 'Fast Food', 'Scandinavian', 'Pakistani',
                'Breweries', 'Dog Parks', 'Afghan', 'Music Venues', 'Tacos',
                'Izakaya', 'Laotian', 'Argentine', 'Barbeque', 'Irish',
                'Tapas Bars', 'Cheese Shops', 'Cupcakes', 'New Mexican Cuisine',
                'Czech', 'Belgian', 'Turkish', 'Salvadoran', 'Tiki Bars', 'Pubs',
                'Whiskey Bars', 'Basque', 'Donuts', 'Noodles', 'Street Vendors',
                'Cambodian', 'Dance Clubs', 'Fondue', 'Yoga', 'Bowling', 'Salad',
                'Jazz & Blues', 'Fruits & Veggies', 'Hawaiian', 'Cantonese',
                'Banks & Credit Unions', 'Juice Bars & Smoothies', 'Senegalese',
                'Boating', 'Zoos', 'Mongolian', 'Conveyor Belt Sushi', 'Beer Bar'],
               dtype=object)
```

```
df['category_title'].unique()

array(['Desserts', 'Breakfast & Brunch', 'American', 'Bakeries',
       'Seafood', 'Burmese', 'Greek', 'Asian Fusion', 'Chinese',
       'Mexican', 'Vietnamese', 'Pizza', 'Bars', 'French', 'British',
       'Landmarks & Historical Buildings', 'Japanese', 'Parks', 'Spanish',
       'Steakhouses', 'Drinks', 'German', 'Italian', 'Museums', 'Korean',
       'Peruvian', 'Hiking', 'Grocery', 'Fast Food', 'Thai', 'Caribbean',
       'Others', 'Mini Golf', 'Indian', 'Food Trucks', 'Vegan',
       'Southern', 'Filipino', 'Latin American', 'Venues & Event Spaces',
       'Sports Bars', 'Mediterranean', 'Ethiopian', 'Middle Eastern',
       'Gastropubs', 'Diners', 'Himalayan/Nepalese', 'Art Museums',
       'Singaporean', 'Moroccan', 'Botanical Gardens', 'Beaches',
       'Food Stands', 'Cafes', 'Vegetarian', 'Scandinavian', 'Pakistani',
       'Dog Parks', 'Afghan', 'Music Venues', 'Laotian', 'Argentine',
       'Barbeque', 'Irish', 'Czech', 'Belgian', 'Turkish', 'Salvadoran',
       'Basque', 'Street Vendors', 'Cambodian', 'Dance Clubs', 'Fondue',
       'Yoga', 'Bowling', 'Jazz & Blues', 'Hawaiian',
       'Banks & Credit Unions', 'Senegalese', 'Boating', 'Zoos',
       'Mongolian'], dtype=object)
```

(3) Drop the non-restaurant business information

3. transformation: We save the final dataframe into CSV. The main columns in this df are names of the restaurants, food category, location coordinates ,reviews, price, and image and url, and yelp ID.

- Pair-Data
  1. Data Collection: In order to collect the pairing data of the parking meters and restaurants, we import two python modules: math and heapq.
  2. We use the following formula to calculate between each two coordinates

```
  dlon = lon2 - lon1
dlat = lat2 - lat1
  a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
  c = 2 * atan2(sqrt(a), sqrt(1 - a))
distance = R * c
```

  3. We use heapq to push closest distance value in sorted order among 32,865 meters and 1,000 restaurants.
     Heap is a module for implementing heaps on ordinary lists, allowing us to extract and append the 10 smallest distance values into the new list. Each restaurant has a pairing result in the format of list of ten list of tuples which contain the distance between that particular restaurant to its 10 closest parking meters, the coordinates of these 10 meters, and the meter IDs.

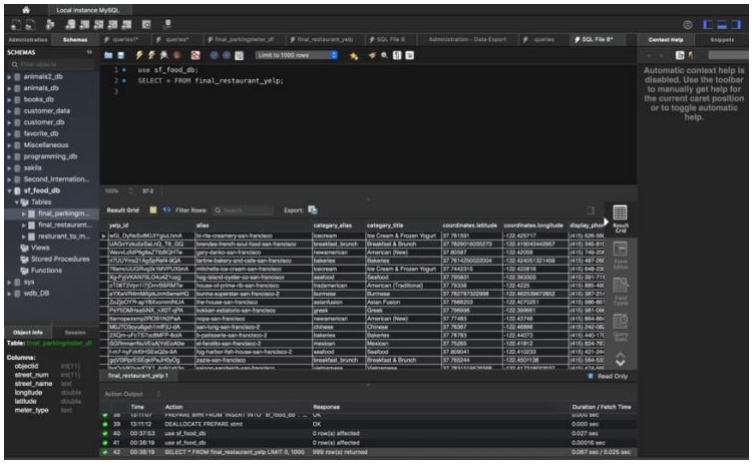
```
[[(-6493.637310698818, (37.7210142024, -122.4805171003), 229415),
  (-6493.636455707242, (37.7219431017, -122.4810152983), 229613),
  (-6493.631206743086, (37.7219452028, -122.48108100120001), 229154),
  (-6493.610427602818, (37.7219538976, -122.4813412984), 229775),
  (-6493.6363200939495, (37.721014599200004, -122.4805294999), 229652),
  (-6493.558514182305, (37.7219454989, -122.4819757973), 229745),
  (-6493.605170468464, (37.7219559987, -122.481407102), 229128),
  (-6493.564071292379, (37.7219632015, -122.4819166993), 229199),
  (-6493.59968673584, (37.7219583009, -122.4814758), 229313),
  (-6493.569977019215, (37.7219705999, -122.4818479007), 229227)],
```

  4. Data Transformation:
     What we need in the new pairing dataset is the restaurant's names and their yelp IDs along with their closest meter IDs. We export the data into

CSV file.

A	B	C	D	E	F	G	H	I	J	K	L	M
yelp_id	meter1	meter2	meter3	meter4	meter5	meter6	meter7	meter8	meter9	meter10	resturant_name	
wGl_DyNxSv	229415	229613	229154	229775	229652	229745	229128	229199	229313	229227	Bi-Rite Creamery	
lJAGnYzku5z	229415	229652	229613	229128	229154	229745	229227	229313	229199	229775	Brenda's French Soul Food	
WavvLdfdP6	229415	229652	229613	229128	229154	229775	229745	229313	229199	229227	Gary Danko	
ri7UUYmx21	229415	229613	229652	229128	229154	229745	229227	229199	229313	229775	Tartine Bakery & Cafe	
76smcUUGR	229568	229415	229128	229154	229652	229227	229745	229775	229199	229313	Mitchell's Ice Cream	

5. Data Loading
   The last step is to load the restaurant data, parking meter data, and pairing data into SQL database for data query applications. We create a sf_food db and then use table Wizard to upload csv file to the db



6. Try to find the closest 3 meter location to the restaurant “Gary Danko”