COMS 4111 project 1 part2

> Note: this notebook can be opened at this Google Colab (https://colab.research.google.com/drive/1UHwwJFimwo2k8wVBSu5kVon2yh7YctRC?usp=sharing)

Teammates:

- Jace Yang (uni: jy3174)
- Binghong Yu (uni: by2325)

The UNI used to create the schema on the course database server: by2325

0. Environment Setup

```
!pip3 install sqlalchemy # ORM for databases
!pip3 install ipython-sql # SQL magic function
```

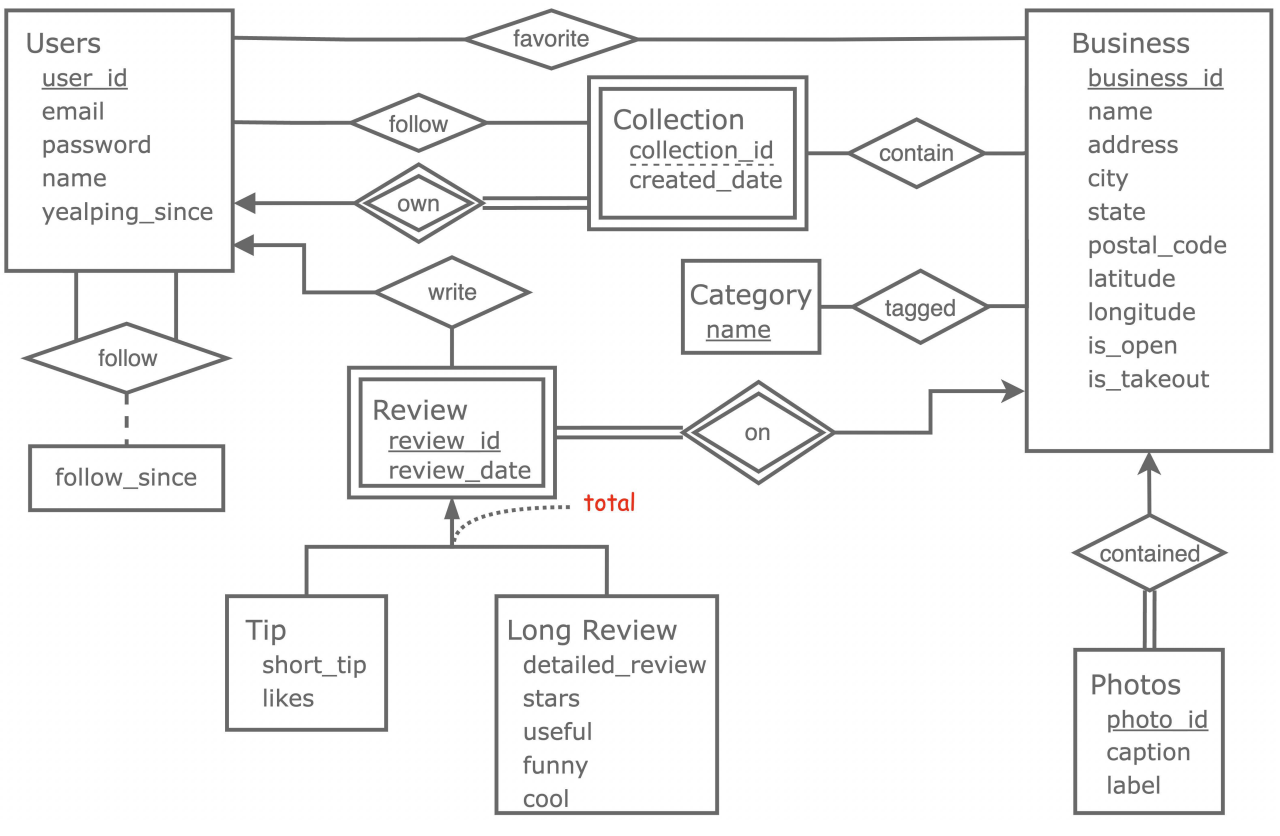
In [1]:
```
%load_ext sql
%sql postgresql://by2325:0316@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/proj1part2
```

```
/usr/local/lib/python3.7/dist-packages/psycopg2/__init__.py:144: UserWarning
: The psycopg2 wheel package will be renamed from release 2.8; in order to k
eep installing from binary please use "pip install psycopg2-binary" instead.
For details see: <http://initd.org/psycopg/docs/install.html#binary-install-
from-pypi>.
  """)
```

Out[1]: 'Connected: by2325@proj1part2'

1 SQL schema

1.1 Latest ER diagram



All the changes we made from Part 1:

1. Delete attrbute `hour` in `Business` that requires json-type data importing with additional package like `pgspecial` (not allowed according to Ed #431 (https://edstem.org/us/courses/17037/discussion/1311446))
2. Change several `int` type id column into `text` to be consistent to the original dataset
3. Add `email` attribute of `Users` table in order to allow costumer to log in. The user_id will be a long text that only used in back-end.
4. Constrain the `password` attribute of `Users` to be in length [8, 16] in order to be more robust

1.2 CREATE commands

In [2]:
```
%%sql
DROP TABLE IF EXISTS Users, Collection_of_User, Review_of_Business, Category,
Business, Photo_contained_Business CASCADE;
CREATE TABLE Users(
    user_id text PRIMARY KEY,
    email text UNIQUE NOT NULL,
    name text NOT NULL,
    password text NOT NULL,
    yealping_since date,
    CHECK (length(password) >= 8 AND length(password) <= 16)
);

CREATE TABlE Collection_of_User(
    user_id text,
    collection_id int,
    created_date date,
    PRIMARY KEY(user_id, collection_id),
    FOREIGN KEY(user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

CREATE TABlE Business(
    business_id text PRIMARY KEY,
    name text,
    address text,
    city text,
    state text,
    postal_code text,
    latitude numeric(4),
    longitude numeric(4),
    is_open boolean,
    is_takeout boolean
);

CREATE TABlE Review_of_Business(
    review_id int PRIMARY KEY,
    review_date date,
    business_id text NOT NULL,
    -- Attributes of Tip
    short_tip text,
    likes int,
    -- Attributes of Long Review
    detailed_review text,
    stars int,
    useful int,
    funny int,
    cool int,

    CHECK (stars >= 0 AND stars <= 5),
    CHECK (
           ((short_tip IS NULL AND likes IS NULL)
            OR
            (detailed_review IS NULL AND stars IS NULL AND useful IS NULL AND
funny IS NULL AND cool IS NULL))
           AND
           ((short_tip IS NOT NULL)
            OR
            (detailed_review IS NOT NULL))
          ),
    CHECK (length(detailed_review) >= 30 OR detailed_review is NULL),

    FOREIGN KEY(business_id) REFERENCES Business(business_id) ON DELETE CASCAD
E
);

CREATE TABlE Category(
    name varchar(255) PRIMARY KEY
);

CREATE TABLE Photo_contained_Business(
    photo_id text PRIMARY KEY,
    business_id text NOT NULL,
    caption text,
    label text,
    FOREIGN KEY(business_id) REFERENCES Business
        ON DELETE CASCADE
);
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[2]:
```
[]
```

In [3]:
```
%%sql
DROP TABLE IF EXISTS Users_favorite_Business, Users_follow_Collection, Collect
ion_contain_Business,
                     Users_write_Review, Collection_contain_Business,
                     Users_follow_Users, Business_tagged_Category CASCADE;

CREATE TABLE Users_favorite_Business(
    user_id text REFERENCES Users(user_id),
    business_id text REFERENCES Business(business_id),
    PRIMARY KEY(user_id, business_id)
);

CREATE TABLE Users_follow_Collection(
    fan_user_id text REFERENCES Users(user_id),
    followee_user_id text,
    collection_id int,
    PRIMARY KEY(fan_user_id, followee_user_id, collection_id),
    FOREIGN KEY(followee_user_id, collection_id) REFERENCES Collection_of_User
(user_id, collection_id)
);

CREATE TABLE Collection_contain_Business(
    collection_owner_id text,
    collection_id int,
    business_id text REFERENCES Business(business_id),
    PRIMARY KEY(collection_owner_id, collection_id, business_id),
    FOREIGN KEY(collection_owner_id, collection_id) REFERENCES Collection_of_U
ser(user_id, collection_id)
);


CREATE TABLE Users_write_Review(
    user_id text NOT NULL REFERENCES Users(user_id) ON DELETE CASCADE, -- allo
w users to cancel their account
    review_id int REFERENCES Review_of_Business(review_id) ON DELETE CASCADE,
    PRIMARY KEY(review_id)
);

CREATE TABLE Business_tagged_Category(
    business_id text REFERENCES Business,
    name text REFERENCES Category,
    PRIMARY KEY(business_id, name)
);

CREATE TABLE Users_follow_Users(
    follwee_user_id text REFERENCES Users(user_id),
    fan_user_id text REFERENCES Users(user_id),
    follow_since date,
    PRIMARY KEY (follwee_user_id, fan_user_id)
);
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
Done.
Done.
Done.
Done.
Done.
Done.
Done.
```

Out[3]:
```
[]
```

2 Data Preparation

In this part, we clean the Yelp dataset which contains mostly real-word data. That being said, we still need to generate some of the features, like `password` of User but all with reasonable assumptions described.

To re-run the code, download the Yelp Dataset (https://www.yelp.com/dataset/documentation/main) into a folder named "data" in the parent level of this file.

To access the cleaned data output from this part, we upload the dataframes in pickle format into a Google Drive (https://drive.google.com/drive/folders/1RvL6q7U1eeMvFVpNyH-dJ1AdkIf_E7l0?usp=sharing).

In [1]:
```
import pickle
pickle.HIGHEST_PROTOCOL = 4
import pandas as pd
import numpy as np
import random, string
from tqdm import tqdm
from datetime import datetime
```

2.1 Entity

Business

In [2]:
```
business = pd.read_json("../data/yelp_dataset/yelp_academic_dataset_business.j
son", lines=True)
```

In [3]:
```
# Only include restaurant of main category
business_category = business[['business_id', 'categories']]
business_category = business_category.assign(category = business.categories.st
r.split(', ')).explode('category').drop('categories', axis=1)
MAIN_FOOD_CATEGORIES = """Bars
Sandwiches
Fast Food
Pizza
Coffee & Tea
Breakfast & Brunch
Burgers
Mexican
Specialty Food
Italian
Seafood
Chicken Wings
Chinese
Salad
Bakeries
Cafes""".split('\n')
business_category = business_category[business_category.category.isin(MAIN_FOO
D_CATEGORIES)]
business = business.merge(business_category[['business_id']].drop_duplicates()
,
                          on="business_id",
                          how="inner").drop(['categories'], axis=1)

# Select N_USER business with moderate amount of reviews
## Note: We finished the data cleaning pipeline, but for the purpose of this h
omework, we only includes 100 restaurant and 200 users that has wrote reviews
on them!
## For project 1 part 3, we will request more resources to allow us populate t
he whole dataset ^.^
N_BUSINESS = 100
business = business[(business['review_count'] >= business['review_count'].quan
tile(.25)) &
                    (business['review_count'] <= business['review_count'].quan
tile(.75))].sample(n=N_BUSINESS, random_state=4111)
business = business.drop(['review_count', 'stars', 'hours'], axis=1).reset_ind
ex(drop=True)

# Extract whether allow takeout information
attribute_values = ['False' if attributes is None else attributes.get('Restaur
antsTakeOut') for attributes in business.attributes]
business['is_takeout'] =  ['False' if value is None or value == 'None' else va
lue for value in attribute_values]
business['is_open'] = business['is_open'].astype(bool)
business = business.drop('attributes', axis=1)
```

In [4]:
```
business.head(5)
```

Out[4]:

	business_id	name	address	city	state	postal_code	latitude	long
0	2gTQ0X9iRTs5zIKs97IWOA	Wayback Burgers	200 West Alexander St	Plant City	FL	33563	27.989818	-82.1
1	TlaXelvS_0yQhjWTLHCLSA	Sciarrinos Pizza	2310 Carpenter Station Rd	Wilmington	DE	19810	39.818019	-75.4
2	sBRKxVbRBmeXY9mVwliqYw	Someone's in the Kitchen	109 Walton Ferry Rd	Hendersonville	TN	37075	36.303316	-86.6
3	KfBpr_NoldM9w0CugY32ow	Wawa	2177 Gulf To Bay Blvd	Clearwater	FL	33765	27.959926	-82.7
4	t24_JnNptChMXityiA6mgQ	Forest Hills Brick Oven Pizza	905 W Linebaugh Ave	Tampa	FL	33612	28.040184	-82.4

In [5]:
```
business.to_pickle('data/Business.pickle')
```

Business_tagged_Category

In [6]:
```
business_category = business_category.merge(business[['business_id']],
                                            on="business_id",
                                            how="inner")
```

In [7]:
```
business_category.head(5)
```

Out[7]:

	business_id	category
0	TEP-73fGvgSUmtQFaOTA_g	Burgers
1	L0-MS0MbQhEWAPLkjCqhpg	Mexican
2	L0-MS0MbQhEWAPLkjCqhpg	Seafood
3	L0-MS0MbQhEWAPLkjCqhpg	Burgers
4	NKplFLr1UebQxMLQAxALPQ	Pizza

In [8]:
```
business_category.to_pickle('data/Business_tagged_Category.pickle')
```

Category

In [9]:
```
category = pd.DataFrame({'name': MAIN_FOOD_CATEGORIES})
```

In [10]:
```
category.head(5)
```

Out[10]:

	name
0	Bars
1	Sandwiches
2	Fast Food
3	Pizza
4	Coffee & Tea

In [11]:
```
category.to_pickle('data/Category.pickle')
```

Users

In [12]:
```
N_USER = 200
size = 500000
# Sample N_USER users among those who have commented on those business
review_json_path = '../data/yelp_dataset/yelp_academic_dataset_review.json'
review = pd.read_json(review_json_path, lines=True,
                      dtype={'review_id':str,'user_id':str,
                             'business_id':str,'stars':int,
                             'date':str,'text':str,'useful':int,
                             'funny':int,'cool':int},
                      chunksize=size)
chunk_list = []
for chunk in tqdm(review):
    chunk = chunk[['business_id', 'user_id']].merge(business[['business_id']],
how='inner')
    chunk_list.append(chunk)
user_id_list = pd.concat(chunk_list, ignore_index=True, axis=0)[["user_id"]].d
rop_duplicates().sample(n=N_USER)
```

```
14it [01:17,  5.55s/it]
```

In [13]:
```
#users_raw = pd.read_json("../data/yelp_dataset/yelp_academic_dataset_user.jso
n", lines=True)
#users_raw.to_pickle("../data/users_raw.pickle")
users_raw = pd.read_pickle('../data/users_raw.pickle')
users = pd.merge(users_raw, user_id_list, how="inner")
```

In [14]:
```
# Extract date of registrating
users["yealping_since"] = pd.to_datetime(users["yelping_since"]).dt.date

# Generate a password for existing users
def password_generator(prefix, length_min=8, length_max=16):
    chars = string.ascii_letters + string.digits + '!@#$%*'
    # Generate a [1, 6], but mostly 1~3 random suffix
    random_suffix_length = round(np.clip(np.random.normal(2, 1), a_min=1, a_ma
x=6))
    random_suffix_length = max(length_min-len(prefix), random_suffix_length)
    random_suffix_length = min(length_max-len(prefix), random_suffix_length)
    random_suffix = ''.join(random.choice(chars) for _ in range(random_suffix_
length))
    return prefix+random_suffix

users["password"] = users.apply(lambda x: password_generator(prefix=x['name'])
, axis=1)

# Generate an email address for existing users
EMAIL_DOMAIN_OPTIONS = ["@gmail.com", "@hotmail.com", "@outlook.com", "@inbox.
com", "@qq.com"]
users['email'] = users.groupby("name")["name"].rank(method="first", ascending=
True).astype(int)
users['email'] = users.apply(lambda x: x['name'] + "_" +str(x['email']) + rand
om.choice(EMAIL_DOMAIN_OPTIONS), axis=1).str.replace("_1@", "@", regex=False)

# Select the columns
users = users[["user_id", "email", "name", "password", "yealping_since"]].rese
t_index(drop=True)
```

In [15]:
```
users.to_pickle('data/Users.pickle')
```

Collection_of_User

Assume 85% of users don't have collection, and 15% of the active users have min(1, $N(2, 4)$) collections that created from a random date from after signup date till the globally maximum signup date.

In [16]:
```
collection = users[['user_id', 'yealping_since']].sample(frac=0.15).reset_inde
x(drop=True)

# Generate local id for each user_id
collection['n_collection'] = pd.Series(np.around(np.random.normal(2, 4, len(co
llection)))).clip(1, 20)
collection['collection_id'] = collection['n_collection'].apply(lambda x: list(
range(1, int(x)+1)))
collection = collection.explode('collection_id').reset_index(drop=True)

# Generate a random collection create date
def random_dates(start, end=max(users.yealping_since), n=10):
    random.seed(4111)
    d = random.randint(0, (end - start).days)
    return start + pd.DateOffset(days=d)
collection['created_time'] = collection.apply(lambda x: random_dates(x['yealpi
ng_since']), axis=1)

# Select columns
collection = collection[["user_id", "collection_id", "created_time"]]
```

In [17]:
```
collection.head()
```

Out[17]:

	user_id	collection_id	created_time
0	k6AwCajLT06J6cwC3SqFcg	1	2020-10-07
1	73oVGO52bkKmjgx4Nwat4Q	1	2021-01-11
2	73oVGO52bkKmjgx4Nwat4Q	2	2021-01-11
3	73oVGO52bkKmjgx4Nwat4Q	3	2021-01-11
4	73oVGO52bkKmjgx4Nwat4Q	4	2021-01-11

In [18]:
```
collection.to_pickle("data/Collection_of_User.pickle")
```

Review_of_Business

In [19]:
```
size = 500000
review_json_path = '../data/yelp_dataset/yelp_academic_dataset_review.json'
review = pd.read_json(review_json_path, lines=True, dtype={'review_id':str,'user_id':str,'business_id':str,'stars':int,'date':str,'text':str,'useful':int,'funny':int,'cool':int},chunksize=size)

chunk_list = []
for chunk in review:
    chunk = chunk.drop(['review_id'], axis=1)
    chunk = chunk.rename(columns={'stars': 'review_stars'})
    chunk = chunk.rename(columns={'text': 'review'})
    chunk_merged = pd.merge(chunk, business[['business_id']], on='business_id', how='inner')
    chunk_merged = pd.merge(chunk_merged, users[['user_id']], on='user_id', how='inner')
    print(f"{chunk_merged.shape[0]} out of {size:,} related reviews")
    chunk_list.append(chunk_merged)

review_df = pd.concat(chunk_list, ignore_index=True, join='outer', axis=0)
```

```
16 out of 500,000 related reviews
17 out of 500,000 related reviews
9 out of 500,000 related reviews
10 out of 500,000 related reviews
16 out of 500,000 related reviews
15 out of 500,000 related reviews
15 out of 500,000 related reviews
10 out of 500,000 related reviews
15 out of 500,000 related reviews
30 out of 500,000 related reviews
23 out of 500,000 related reviews
5 out of 500,000 related reviews
14 out of 500,000 related reviews
14 out of 500,000 related reviews
```

In [20]:
```
size = 500000
tip_json_path = '../data/yelp_dataset/yelp_academic_dataset_tip.json'
tip = pd.read_json(tip_json_path, lines=True, dtype={'text':str,'date':str,'compliment_count':int,'business_id':str,'user_id': int},chunksize=size)

chunk_list_tip = []
for chunk in tip:
    chunk = chunk.rename(columns={'text': 'tip'})
    chunk_merged = pd.merge(chunk, business[['business_id']], on='business_id', how='inner')
    chunk_merged = pd.merge(chunk_merged, users[['user_id']], on='user_id', how='inner')
    print(f"{chunk_merged.shape[0]} out of {size:,} related tips")
    chunk_list_tip.append(chunk_merged)

tip_df = pd.concat(chunk_list_tip, ignore_index=True, join='outer', axis=0)
```

```
8 out of 500,000 related tips
2 out of 500,000 related tips
```

In [21]:
```
review_df = review_df[review_df['review'].str.len()>=30]
review_dfs = pd.merge(tip_df, review_df, on=['user_id','business_id','date'], how='outer')
review_dfs = review_dfs.rename(columns={'tip': 'short_tip','date':'review_date','review_stars':'stars','review':'detailed_review','compliment_count':'likes'})
review_dfs['review_date'] =  pd.to_datetime(review_dfs['review_date']).dt.date
# review_dfs[['likes','stars','useful','funny','cool','detailed_review']] = review_dfs[['likes','stars','useful','funny','cool', 'detailed_review']]
#review_dfs= review_dfs.where(review_dfs.notnull(),None)
```

In [22]:
```
# Add globally unique review_id
review_dfs['review_id'] = [i + 100000 for i in range(len(review_dfs))]

# Select columns
review = review_dfs[['review_id','review_date','business_id','short_tip','likes','detailed_review','stars','useful','funny','cool']]
```

In [23]:
```
review.head()
```

Out[23]:

	review_id	review_date	business_id	short_tip	likes	detailed_review	stars	useful	fu
0	100000	2013-05-13	inyckJCTAiQ8ro5ShDi6OQ	What's not to like....fresh produce, vegetable...	0.0	NaN	NaN	NaN	
1	100001	2013-06-11	BFPxxguGxBZQ0kR0rWePTQ	Good as any.....	0.0	NaN	NaN	NaN	
2	100002	2015-09-03	xuFjcrdGxISZVfLVl0mttA	Starter menu trying out a few key entrees and ...	0.0	NaN	NaN	NaN	
3	100003	2015-08-24	xuFjcrdGxISZVfLVl0mttA	If you need an accessible table I suggest call...	0.0	NaN	NaN	NaN	
4	100004	2014-02-08	MN5A-cUnGnffkN2wf6Y6MQ	I have an addiction to their crab rangoons. No...	0.0	NaN	NaN	NaN	

In [24]:
```
review.to_pickle('data/Review_of_Business.pickle')
```

Users_write_Review

In [25]:
```
users_write_review = review_dfs[['user_id', 'review_id']]
```

In [26]:
```
users_write_review.head()
```

Out[26]:

	user_id	review_id
0	RKULSOrlvvYpDmtuYXEXzA	100000
1	RKULSOrlvvYpDmtuYXEXzA	100001
2	Kgb1KdaTrRnGqQ4misL12w	100002
3	96f2e36vpRvMBEZ92xiwIQ	100003
4	kkNFDL_bfM4BP65UEaDs_w	100004

In [27]:
```
users_write_review.to_pickle('data/Users_write_Review.pickle')
```

Photo_contained_Business

In [28]:
```
photo_json_path = '../data/yelp_photos/photos.json'
photo = pd.read_json(photo_json_path, lines=True, dtype={'photo_id':str,'business_id':str,'caption':str,'label':str})
photo = pd.merge(photo, business[['business_id']], on=['business_id'], how='inner')
```

In [29]:
```
photo.head()
```

Out[29]:

	photo_id	business_id	caption	label
0	YytrZGwGLAscS-270DUU_w	tLMWVzUBGjklGlyEQQLxXQ		outside
1	26WQjQQO6dpXQh__MQfzfA	tLMWVzUBGjklGlyEQQLxXQ		inside
2	lPDxpMwbSldbX5atTBNUng	tLMWVzUBGjklGlyEQQLxXQ	Aja channelside in downtown Tampa	inside
3	AFo1Dt0NUc2MBcdBSahYSA	tLMWVzUBGjklGlyEQQLxXQ	Upstairs VIP	inside
4	L96SgH_HdVQe0y3XrdmcGg	OLGqB9dRca8Vib7IMdtC8A	NYE	drink

In [30]:
```
photo.to_pickle("data/Photo_contained_Business.pickle")
```

2.2 Relationship

Users_favorite_Business

Assume 60% of users don't have collection, and 40% of the active users have min(1, $N(3, 5)$) business favorated that they have reviewed, which happened at a random date from after signup date till the globally maximum signup date.

In [31]:
```
user_business = users[['user_id']].sample(frac=0.4).reset_index(drop=True)

# Generate local id for each user_id
user_business['n_bz_follow'] = pd.Series(np.around(np.random.normal(3, 5, len(user_business)))).clip(lower=1).astype('int')
user_business['business_id'] = user_business['n_bz_follow'].apply(lambda n: business.sample(n).business_id.tolist())
user_business = user_business.explode(["business_id"])
user_business = user_business[['user_id', 'business_id']]
```

In [32]:
```
user_business.head(5)
```

Out[32]:

	user_id	business_id
0	SyOG4eHZK3wv3MnpiZVX9w	HCUsFoHYsMc_-qvIVA19IA
1	K-B9Ir8e0B-aro8p1VYDqw	Kt6aKSP97edaiSs-hEZzNw
1	K-B9Ir8e0B-aro8p1VYDqw	u48EHQFGHF5FwDTqAFXVwQ
1	K-B9Ir8e0B-aro8p1VYDqw	79sRRGDXhjRvxvBkQ4W9CQ
1	K-B9Ir8e0B-aro8p1VYDqw	aq5Y8xr0pwrwRI9yL4KgoA

In [33]:
```
user_business.to_pickle("data/Users_favorite_Business.pickle")
```

Users_follow_Collection

Assume 80% of users don't follow any collection, and 20% of the active users follow max(1, N(2, 5)) collections (include theirselves)>

In [34]:
```
users_collection = pd.DataFrame()
users_collection['fan_user_id'] = users[['user_id']]

# Random some collections for user to follow
users_collection['n_collection_follow'] = np.round(np.maximum(1, np.random.nor
mal(2, 5, len(users_collection)))).astype('int')
def get_collection(fan):
    df = collection.sample(fan['n_collection_follow'])
    fan['followee_user_id'] = df.user_id.tolist()
    fan['collection_id'] = df.collection_id.tolist()
    return fan
users_collection = users_collection.apply(get_collection, axis=1).explode(["fo
llowee_user_id", "collection_id"])
users_collection = users_collection[['fan_user_id','followee_user_id','collect
ion_id']]
```

In [35]:
```
collection.head()
```

Out[35]:

	user_id	collection_id	created_time
0	k6AwCajLT06J6cwC3SqFcg	1	2020-10-07
1	73oVGO52bkKmjgx4Nwat4Q	1	2021-01-11
2	73oVGO52bkKmjgx4Nwat4Q	2	2021-01-11
3	73oVGO52bkKmjgx4Nwat4Q	3	2021-01-11
4	73oVGO52bkKmjgx4Nwat4Q	4	2021-01-11

In [36]:
```
users_collection.to_pickle('data/Users_follow_Collection.pickle')
```

Collection_contain_Business

Assume every collections collects max(1, N(3, 5)) business.

In [37]:
```
collection_business = pd.DataFrame()
collection_business['collection_owner_id'] = collection['user_id']
collection_business['collection_id'] = collection['collection_id']
collection_business['n_business_contain'] = np.round(np.maximum(1, np.random.n
ormal(3, 5, len(collection_business)))).astype('int')
collection_business['business_id'] = collection_business['n_business_contain']
.apply(lambda n: business.sample(n).business_id.tolist())
collection_business = collection_business.explode('business_id')
collection_business = collection_business[['collection_owner_id','collection_i
d', 'business_id']]
```

In [38]:
```
collection_business.head()
```

Out[38]:

	collection_owner_id	collection_id	business_id
0	k6AwCajLT06J6cwC3SqFcg	1	kS-UWuhV8kxAdMN3RW0Ssw
0	k6AwCajLT06J6cwC3SqFcg	1	MSnWYdS0w5m9JLcr1wHO4w
0	k6AwCajLT06J6cwC3SqFcg	1	tLMWVzUBGjklGlyEQQLxXQ
0	k6AwCajLT06J6cwC3SqFcg	1	9TvKg94l4PA2xuViDlvOuA
0	k6AwCajLT06J6cwC3SqFcg	1	BXPyrf12pvtP6fXNvYZUeg

In [39]:
```
collection_business.to_pickle('data/Collection_contain_Business.pickle')
```

Users_follow_Users

Assume 70% of users don't follow any users, and 30% of the active users have min(0, $N(3, 5)$) fans, and follow_since happened at a random date from after signup date till the globally maximum signup date.

In [40]:
```
users_follow= pd.DataFrame()
users_follow = users[['user_id']].sample(frac=0.3).reset_index(drop=True) # on
ly 30%users have fans
users_follow = users_follow.rename(columns={'user_id': 'fan_user_id'})

# Generate follower
users_follow['n_followers'] = np.round(np.maximum(1, np.random.normal(3, 5, le
n(users_follow)))).astype('int')
def get_fan(fan):
    df = users.sample(fan['n_followers'])
    fan['followee_user_id'] = df.user_id.tolist()
    #fan['collection_id'] = df.collection_id.tolist()
    return fan
users_follow = users_follow.apply(get_fan, axis=1).explode(["followee_user_id"
])

# Generate follow date
def random_dates(start, end=max(users.yealping_since), n=10):
    random.seed(4111)
    d = random.randint(0, (end - start).days)
    return start + pd.DateOffset(days=d)
users_follow['follow_since'] = users.apply(lambda x: random_dates(x['yealping_
since']), axis=1)

users_follow = users_follow[['fan_user_id','followee_user_id','follow_since']]
```

In [41]:
```
users_follow.head()
```

Out[41]:

	fan_user_id	followee_user_id	follow_since
0	7YSylmBoZxHTWF3j-2a4zA	OiqOaU31KVRB7RfyvZeM1w	2020-04-10
1	uOkJt5kTu7wafarASZRvqQ	ixh7QWfylnpeFQ1rGG-8eg	2019-12-02
1	uOkJt5kTu7wafarASZRvqQ	og-NLmnYAAr1CC1iYDluog	2019-12-02
1	uOkJt5kTu7wafarASZRvqQ	hdrfHfM-wd9P5VCmMnJv1A	2019-12-02
1	uOkJt5kTu7wafarASZRvqQ	RdLTUqd662yMT1IcCG5sQw	2019-12-02

In [42]:
```
users_follow.to_pickle("data/Users_follow_Users.pickle")
```

3 Data Population

To avoid directly populating sql table by pandas dataframe, we first generate this code automatically for all tables:

```
for each table:
    for each row in table:
        for each column in the table:
            extract the value
        %sql insert the row
```

and then run it!

3.1 Generate data population SQLs automatically

To avoid using additional package to achieve data importing from dataframe/csv, we automatic the method suggested in [ED #450 (https://edstem.org/us/courses/17037/discussion/1317472)](https://edstem.org/us/courses/17037/discussion/1317472) that INSERT tuples table-by-table and line-by-line.

In [4]:
```
import pandas as pd
import os
from google.colab import drive
drive.mount('/content/drive')
table_names = ['Users',
               'Collection_of_User',
               'Business',
               'Review_of_Business',
               'Category',
               'Photo_contained_Business',
               'Users_favorite_Business',
               'Users_follow_Collection',
               'Users_write_Review',
               'Collection_contain_Business',
               'Business_tagged_Category',
               'Users_follow_Users']

folder_path = "/content/drive/MyDrive/COMS 4111/Project 1/data"
file_names = [table_name + '.pickle' for table_name in table_names]
file_paths = [folder_path + '/' + file_name for file_name in file_names]
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [5]:
```
def get_code(table_name):
    df = pd.read_pickle(folder_path + '/' + table_name + '.pickle')
    data_import_code = f'{table_name} = pd.read_pickle(folder_path + "/{table_
name}.pickle")'
    for_loop_code = f'for index, row in {table_name}.iterrows():'
    clean_null_code = f'{table_name} = {table_name}.astype(object).where({tabl
e_name}.notna(), None)'
    value_extract_code = '\n'.join([f'\tvalue{col_idx} = row["{column}"]' for
col_idx, column in enumerate(df.columns)])
    value_inser_code = f'\t%sql INSERT INTO {table_name} VALUES (' + ', '.join
([f':value{i}' for i in range(len(df.columns))]) + ')'
    print('\n'.join([data_import_code, clean_null_code, for_loop_code, value_e
xtract_code, value_inser_code]))

for table_name in table_names:
    print(f'# Populate {table_name} table')
    get_code(table_name)
    print('')
```

```
# Populate Users table
Users = pd.read_pickle(folder_path + "/Users.pickle")
Users = Users.astype(object).where(Users.notna(), None)
for index, row in Users.iterrows():
        value0 = row["user_id"]
        value1 = row["email"]
        value2 = row["name"]
        value3 = row["password"]
        value4 = row["yealping_since"]
        %sql INSERT INTO Users VALUES (:value0, :value1, :value2, :value3, :
value4)

# Populate Collection_of_User table
Collection_of_User = pd.read_pickle(folder_path + "/Collection_of_User.pickl
e")
Collection_of_User = Collection_of_User.astype(object).where(Collection_of_U
ser.notna(), None)
for index, row in Collection_of_User.iterrows():
        value0 = row["user_id"]
        value1 = row["collection_id"]
        value2 = row["created_time"]
        %sql INSERT INTO Collection_of_User VALUES (:value0, :value1, :value
2)

# Populate Business table
Business = pd.read_pickle(folder_path + "/Business.pickle")
Business = Business.astype(object).where(Business.notna(), None)
for index, row in Business.iterrows():
        value0 = row["business_id"]
        value1 = row["name"]
        value2 = row["address"]
        value3 = row["city"]
        value4 = row["state"]
        value5 = row["postal_code"]
        value6 = row["latitude"]
        value7 = row["longitude"]
        value8 = row["is_open"]
        value9 = row["is_takeout"]
        %sql INSERT INTO Business VALUES (:value0, :value1, :value2, :value3
, :value4, :value5, :value6, :value7, :value8, :value9)

# Populate Review_of_Business table
Review_of_Business = pd.read_pickle(folder_path + "/Review_of_Business.pickl
e")
Review_of_Business = Review_of_Business.astype(object).where(Review_of_Busin
ess.notna(), None)
for index, row in Review_of_Business.iterrows():
        value0 = row["review_id"]
        value1 = row["review_date"]
        value2 = row["business_id"]
        value3 = row["short_tip"]
        value4 = row["likes"]
        value5 = row["detailed_review"]
        value6 = row["stars"]
        value7 = row["useful"]
        value8 = row["funny"]
        value9 = row["cool"]
        %sql INSERT INTO Review_of_Business VALUES (:value0, :value1, :value
2, :value3, :value4, :value5, :value6, :value7, :value8, :value9)

# Populate Category table
Category = pd.read_pickle(folder_path + "/Category.pickle")
Category = Category.astype(object).where(Category.notna(), None)
for index, row in Category.iterrows():
        value0 = row["name"]
        %sql INSERT INTO Category VALUES (:value0)

# Populate Photo_contained_Business table
Photo_contained_Business = pd.read_pickle(folder_path + "/Photo_contained_Bu
siness.pickle")
Photo_contained_Business = Photo_contained_Business.astype(object).where(Pho
to_contained_Business.notna(), None)
for index, row in Photo_contained_Business.iterrows():
        value0 = row["photo_id"]
        value1 = row["business_id"]
        value2 = row["caption"]
        value3 = row["label"]
        %sql INSERT INTO Photo_contained_Business VALUES (:value0, :value1,
:value2, :value3)

# Populate Users_favorite_Business table
Users_favorite_Business = pd.read_pickle(folder_path + "/Users_favorite_Busi
ness.pickle")
Users_favorite_Business = Users_favorite_Business.astype(object).where(Users
_favorite_Business.notna(), None)
for index, row in Users_favorite_Business.iterrows():
        value0 = row["user_id"]
        value1 = row["business_id"]
        %sql INSERT INTO Users_favorite_Business VALUES (:value0, :value1)

# Populate Users_follow_Collection table
Users_follow_Collection = pd.read_pickle(folder_path + "/Users_follow_Collec
tion.pickle")
Users_follow_Collection = Users_follow_Collection.astype(object).where(Users
_follow_Collection.notna(), None)
for index, row in Users_follow_Collection.iterrows():
        value0 = row["fan_user_id"]
        value1 = row["followee_user_id"]
        value2 = row["collection_id"]
        %sql INSERT INTO Users_follow_Collection VALUES (:value0, :value1, :
value2)

# Populate Users_write_Review table
Users_write_Review = pd.read_pickle(folder_path + "/Users_write_Review.pickl
e")
Users_write_Review = Users_write_Review.astype(object).where(Users_write_Rev
iew.notna(), None)
for index, row in Users_write_Review.iterrows():
        value0 = row["user_id"]
        value1 = row["review_id"]
```

```
        %sql INSERT INTO Users_write_Review VALUES (:value0, :value1)

# Populate Collection_contain_Business table
Collection_contain_Business = pd.read_pickle(folder_path + "/Collection_cont
ain_Business.pickle")
Collection_contain_Business = Collection_contain_Business.astype(object).whe
re(Collection_contain_Business.notna(), None)
for index, row in Collection_contain_Business.iterrows():
        value0 = row["collection_owner_id"]
        value1 = row["collection_id"]
        value2 = row["business_id"]
        %sql INSERT INTO Collection_contain_Business VALUES (:value0, :value
1, :value2)

# Populate Business_tagged_Category table
Business_tagged_Category = pd.read_pickle(folder_path + "/Business_tagged_Ca
tegory.pickle")
Business_tagged_Category = Business_tagged_Category.astype(object).where(Bus
iness_tagged_Category.notna(), None)
for index, row in Business_tagged_Category.iterrows():
        value0 = row["business_id"]
        value1 = row["category"]
        %sql INSERT INTO Business_tagged_Category VALUES (:value0, :value1)

# Populate Users_follow_Users table
Users_follow_Users = pd.read_pickle(folder_path + "/Users_follow_Users.pickl
e")
Users_follow_Users = Users_follow_Users.astype(object).where(Users_follow_Us
ers.notna(), None)
for index, row in Users_follow_Users.iterrows():
        value0 = row["fan_user_id"]
        value1 = row["followee_user_id"]
        value2 = row["follow_since"]
        %sql INSERT INTO Users_follow_Users VALUES (:value0, :value1, :value
2)
```

3.2 Run the above code to populate the data

In [6]:

```
# Populate Users table
Users = pd.read_pickle(folder_path + "/Users.pickle")
Users = Users.astype(object).where(Users.notna(), None)
for index, row in Users.iterrows():
        value0 = row["user_id"]
        value1 = row["email"]
        value2 = row["name"]
        value3 = row["password"]
        value4 = row["yealping_since"]
        %sql INSERT INTO Users VALUES (:value0, :value1, :value2, :value3, :va
lue4)

# Populate Collection_of_User table
Collection_of_User = pd.read_pickle(folder_path + "/Collection_of_User.pickle"
)
Collection_of_User = Collection_of_User.astype(object).where(Collection_of_Use
r.notna(), None)
for index, row in Collection_of_User.iterrows():
        value0 = row["user_id"]
        value1 = row["collection_id"]
        value2 = row["created_time"]
        %sql INSERT INTO Collection_of_User VALUES (:value0, :value1, :value2)

# Populate Business table
Business = pd.read_pickle(folder_path + "/Business.pickle")
Business = Business.astype(object).where(Business.notna(), None)
for index, row in Business.iterrows():
        value0 = row["business_id"]
        value1 = row["name"]
        value2 = row["address"]
        value3 = row["city"]
        value4 = row["state"]
        value5 = row["postal_code"]
        value6 = row["latitude"]
        value7 = row["longitude"]
        value8 = row["is_open"]
        value9 = row["is_takeout"]
        %sql INSERT INTO Business VALUES (:value0, :value1, :value2, :value3,
:value4, :value5, :value6, :value7, :value8, :value9)

# Populate Review_of_Business table
Review_of_Business = pd.read_pickle(folder_path + "/Review_of_Business.pickle"
)
Review_of_Business = Review_of_Business.astype(object).where(Review_of_Busines
s.notna(), None)
for index, row in Review_of_Business.iterrows():
        value0 = row["review_id"]
        value1 = row["review_date"]
        value2 = row["business_id"]
        value3 = row["short_tip"]
        value4 = row["likes"]
        value5 = row["detailed_review"]
        value6 = row["stars"]
        value7 = row["useful"]
        value8 = row["funny"]
        value9 = row["cool"]
        %sql INSERT INTO Review_of_Business VALUES (:value0, :value1, :value2,
:value3, :value4, :value5, :value6, :value7, :value8, :value9)

# Populate Category table
Category = pd.read_pickle(folder_path + "/Category.pickle")
Category = Category.astype(object).where(Category.notna(), None)
for index, row in Category.iterrows():
        value0 = row["name"]
        %sql INSERT INTO Category VALUES (:value0)

# Populate Photo_contained_Business table
Photo_contained_Business = pd.read_pickle(folder_path + "/Photo_contained_Busi
```

```
ness.pickle")
Photo_contained_Business = Photo_contained_Business.astype(object).where(Photo
_contained_Business.notna(), None)
for index, row in Photo_contained_Business.iterrows():
        value0 = row["photo_id"]
        value1 = row["business_id"]
        value2 = row["caption"]
        value3 = row["label"]
        %sql INSERT INTO Photo_contained_Business VALUES (:value0, :value1, :v
alue2, :value3)

# Populate Users_favorite_Business table
Users_favorite_Business = pd.read_pickle(folder_path + "/Users_favorite_Busine
ss.pickle")
Users_favorite_Business = Users_favorite_Business.astype(object).where(Users_f
avorite_Business.notna(), None)
for index, row in Users_favorite_Business.iterrows():
        value0 = row["user_id"]
        value1 = row["business_id"]
        %sql INSERT INTO Users_favorite_Business VALUES (:value0, :value1)

# Populate Users_follow_Collection table
Users_follow_Collection = pd.read_pickle(folder_path + "/Users_follow_Collecti
on.pickle")
Users_follow_Collection = Users_follow_Collection.astype(object).where(Users_f
ollow_Collection.notna(), None)
for index, row in Users_follow_Collection.iterrows():
        value0 = row["fan_user_id"]
        value1 = row["followee_user_id"]
        value2 = row["collection_id"]
        %sql INSERT INTO Users_follow_Collection VALUES (:value0, :value1, :va
lue2)

# Populate Users_write_Review table
Users_write_Review = pd.read_pickle(folder_path + "/Users_write_Review.pickle"
)
Users_write_Review = Users_write_Review.astype(object).where(Users_write_Revie
w.notna(), None)
for index, row in Users_write_Review.iterrows():
        value0 = row["user_id"]
        value1 = row["review_id"]
        %sql INSERT INTO Users_write_Review VALUES (:value0, :value1)

# Populate Collection_contain_Business table
Collection_contain_Business = pd.read_pickle(folder_path + "/Collection_contai
n_Business.pickle")
Collection_contain_Business = Collection_contain_Business.astype(object).where
(Collection_contain_Business.notna(), None)
for index, row in Collection_contain_Business.iterrows():
        value0 = row["collection_owner_id"]
        value1 = row["collection_id"]
        value2 = row["business_id"]
        %sql INSERT INTO Collection_contain_Business VALUES (:value0, :value1,
:value2)

# Populate Business_tagged_Category table
Business_tagged_Category = pd.read_pickle(folder_path + "/Business_tagged_Cate
gory.pickle")
Business_tagged_Category = Business_tagged_Category.astype(object).where(Busin
ess_tagged_Category.notna(), None)
for index, row in Business_tagged_Category.iterrows():
        value0 = row["business_id"]
        value1 = row["category"]
        %sql INSERT INTO Business_tagged_Category VALUES (:value0, :value1)

# Populate Users_follow_Users table
Users_follow_Users = pd.read_pickle(folder_path + "/Users_follow_Users.pickle"
)
Users_follow_Users = Users_follow_Users.astype(object).where(Users_follow_User
s.notna(), None)
for index, row in Users_follow_Users.iterrows():
        value0 = row["fan_user_id"]
        value1 = row["followee_user_id"]
        value2 = row["follow_since"]
        %sql INSERT INTO Users_follow_Users VALUES (:value0, :value1, :value2)
```

```
Streaming output truncated to the last 5000 lines.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
```

```
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
```

```
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
1 rows affected.
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
```

```
# Initial SQL INSERT code generating method we try without using the "for looop %s
ql"
table_nam = "Users"
for index, row in Users[:5].iterrows():
    INSERT_str = f"INSERT INTO {table_name} VALUES ("
    for column, column_type in zip(Users.columns, Users.dtypes):
        if column_type != 'object':
            INSERT_str += str(row[column]) + ", "
        else:
            INSERT_str += "'" + str(row[column]) + "', "
    INSERT_str = INSERT_str[:-2] + ");"
    print(INSERT_str)
```

4 Interesting SELECT queries.

4.1 Find the top 5 rated open resturant in average that has at least 5 review!

In [7]:
```
%%sql
SELECT name, address, city, round(AVG(stars), 2) AS average_stars
FROM Review_of_Business JOIN Business USING(business_id)
WHERE detailed_review IS NOT NULL AND is_open = True
GROUP BY business_id, name, address, city
HAVING count(*) >= 5
ORDER BY average_stars DESC LIMIT 5
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
5 rows affected.
```

Out[7]:

name	address	city	average_stars
BRŪ Florida Growler Bar	8729 Gunn Hwy	Odessa	5.00
Il Brother's Pizza & Restaurant	485 Baltimore Pike	Glen Mills	4.60
Tandoori Of India	1100 Jackson St	Philadelphia	4.00
Flying Cupcake Bakery	4026 E 82nd St	Indianapolis	3.20
Port of Subs	720 S Meadows Pkwy, Ste 4	Reno	3.00

4.2 Find the TOP 5 users with most fans

In [8]:
```
%%sql
WITH user_fans_cnt AS(
    SELECT follwee_user_id as user_id, COUNT(fan_user_id) as fans_cnt
    FROM Users_follow_Users
    GROUP BY follwee_user_id)
SELECT user_id, name, email, yealping_since, fans_cnt
FROM user_fans_cnt JOIN Users USING(user_id)
ORDER BY fans_cnt DESC, yealping_since ASC
LIMIT 5
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
5 rows affected.
```

Out[8]:

user_id	name	email	yealping_since	fans_cnt
FX0m5j4zH_g4ZDSJ2LZ1ww	Mekayla	Mekayla@inbox.com	2013-06-02	11
AmM4Lfk15XfqRGCINWgkOQ	Stewart	Stewart@inbox.com	2010-07-13	10
YJZbWx8BhQyBu28mZ6XtAQ	Chiddy	Chiddy@gmail.com	2013-02-15	10
sTQ6Z-ON9pWvlla1kpSB0A	Brian	Brian_2@outlook.com	2016-01-01	10
KxN2A12tA4u-9ldFQTfjqw	Janice	Janice@qq.com	2014-06-29	9

4.3 For all the categories, calculate the number of followers users through `Collection` that contain the restaurant

In [9]:
```
%%sql
WITH collection_cnt AS(
    SELECT followee_user_id AS collection_owner_id, collection_id, COUNT(fan_u
ser_id) AS fans_cnt
    FROM Users_follow_Collection
    GROUP BY followee_user_id, collection_id)

SELECT cate.name,
       COUNT(CONCAT(collection_owner_id, collection_id)) AS n_collections,
       SUM(fans_cnt) as total_fans_among_all_collections
FROM collection_cnt
    JOIN Collection_contain_Business USING(collection_owner_id, collection_id)
    JOIN Business USING(business_id)
    JOIN Business_tagged_Category cate USING(business_id)
WHERE is_open = True
GROUP BY cate.name
ORDER BY total_fans_among_all_collections DESC
```

```
 * postgresql://by2325:***@w4111.cisxo09blonu.us-east-1.rds.amazonaws.com/pr
oj1part2
16 rows affected.
```

Out[9]:

name	n_collections	total_fans_among_all_collections
Coffee & Tea	30	307
Fast Food	25	263
Sandwiches	26	258
Salad	24	248
Breakfast & Brunch	23	232
Bakeries	24	229
Mexican	21	219
Pizza	21	206
Specialty Food	17	167
Bars	16	167
Burgers	13	132
Chinese	12	112
Italian	11	107
Cafes	9	78
Chicken Wings	8	77
Seafood	7	72