CSI2132 Database I

2018winter
"Restaurant Rating Database"
Project Report

Weizhe Liang 8136867
Zaeem Qureshi 7320339
Xiaoxin Zhou 7957115

Introduction

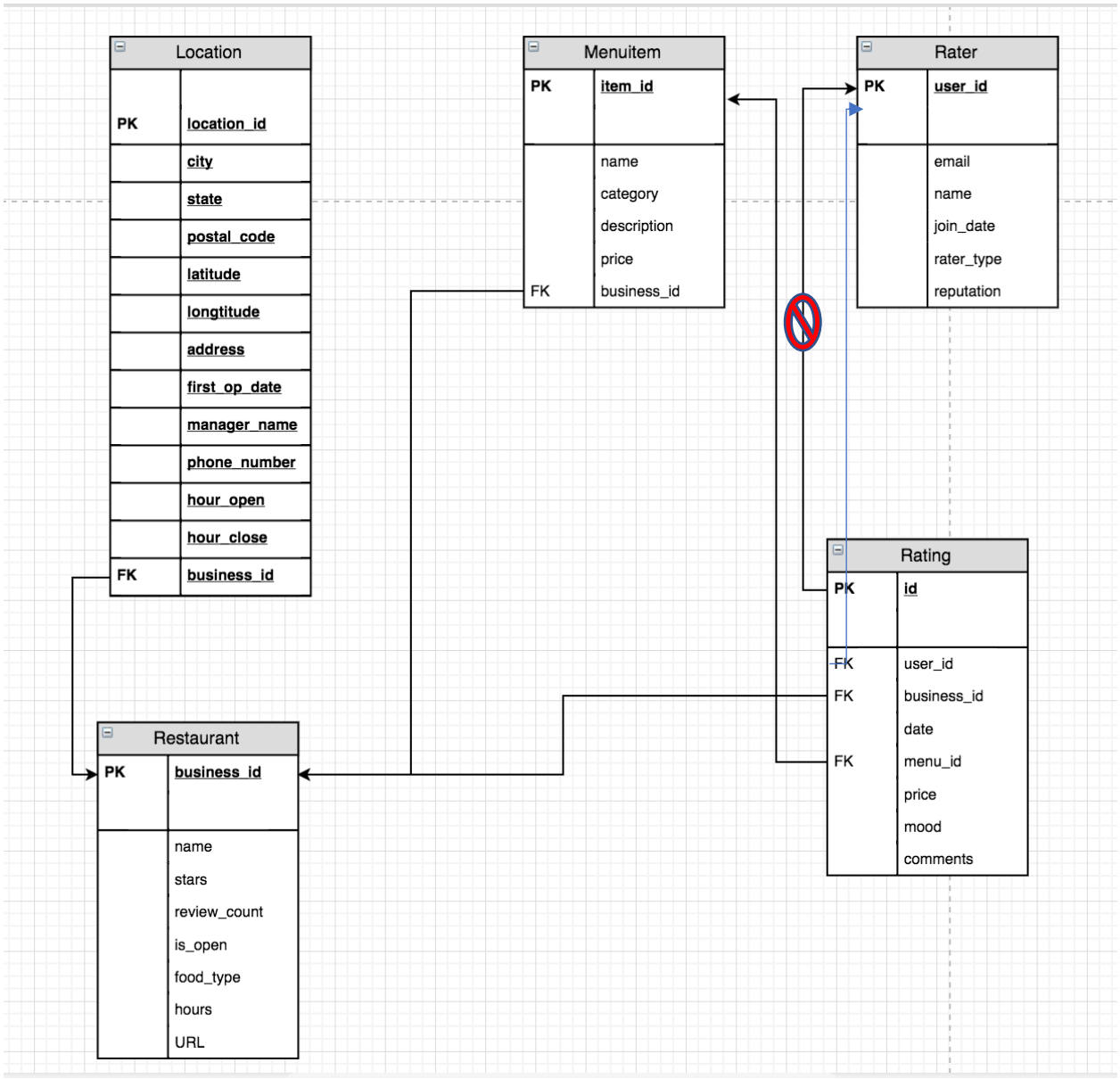
This report is helping you to understand our project structure and environment setup requirements; also, it instructs you to achieve the functions states in the project description.

Files hierarchy:

- Dataset (We found the json dataset from www.yelp.com/dataset/download , and Dish.csv online.)
    - Business.json
    - Checkin.json
    - Dish.csv
    - Photos.json
    - Review.json
    - Tip.json
    - User.json
    - …pdf
    - …pdf
- Server (Back end)
    - __init__.py
    - App.py (API)
    - Config.py (Input your database server information)
    - Forms.py
    - Model.py (Create all the tables and schemas)
    - N.py
    - Seed.py (Parse data from json dataset)
- Static (Front end)
    - Img
    - Favicon.ico
    - Style1.css
    - Styles.css
- Template (Front end)
    - all html
- Requirements.txt (all frameworks and libraries we used in this project)

Explanations Regards to Requirements in 'ProjectDescription'

1.



Create table location(Location_id integer, City varchar State varchar Postal_code varchar Latitude float Longtitude float Address varchar First_op_date DateTime Manager_name varchar Phone_number varchar(10) Hour_open varchar(7) Hour_close varchar (7) Business_id varchar Primary key(location_id) Foreign key(business_id) Reference (restaurant));	Create table menuitem(Item_id integer Name varchar category varchar description varchar price integer business_id varchar primary key (item_id) foreign key (business_id) reference (restaurant));
Create table rater(User_id varchar (22) Name varchar Email varchar (120) Join_date timestamp Reputation integer Rater_type integer Primary key (user_id));	Create table restaurant(Business_id varchar Name varchar Stars float Review_count integer Is_open integer Food_type varchar::[] Hours varchar:: [] URL varchar Primary key (business_id));
Create table rating(Id integer User_id varchar (22) Business_id varchar Date timestamp Comments varchar Mood integer Price integer Menu_id integer Primary key (id) Foreign key(user_id) Reference (raters)	

Foreign key (business_id) Reference (restaurant) Foreign key(menu_id) Reference (menuitem));	

2. We used yelp dataset, which has sufficient data.
3. Add and Delete functions in app.py

	restaurant	rater	menuitem
add	newRestaurant()	newRater()	newMenuItem()
delete	deleteRestaurant()	deleteRater()	deleteMenuItem()

4. Queries

<u>Restaurants and menus</u>

a. Select * from restaurant where business_id = ''
b. Select * from menuitem where business_id = ''
c. Select manager_name, first_op_date from location where business_id = ''
d. Select location.manager_name, location.hour_open, restaurant.URL
   From location join restaurant on location.business_id = restaurant.business_id
   Join menuitem
   Where
e. select unnest(r.food_type) , m.category , avg(m.price)
   from restaurant as r join menuitem as m on m.business_id = r.business_id
   group by (r.food_type,m.category)
   order by r.food_type asc

<u>Ratings of restaurants</u>

f. Select rater.name, rater.join_date, count(rating)
   From rater join rating on rater.user_id = rating.user_id
   Group by rater.user_id

   select count(rating.business_id) , restaurant.name ,rater.name
   from rater join (
   restaurant join rating on restaurant.business_id = rating.business_id)
   on rater.user_id = rating.user_id
   group by restaurant.name ,rater.name

g. Select restaurant.name, location.phone_number, restaurant.category
   from restaurant join location on restaurant.business_id=location.business_id
   where not exist(
   select *
   from restaurant join rating on restaurant.business_id=rating.business_id

```
        where date != '2015-01' )
```

h.
```
select rater.name, location.first_open_date, rating.mood
from rater join (location join rating on location.business_id=rating.business_id)
on rating.user_id=rater.user_id
where rater.rater_type= '0'
group by rater.name, location.first_open_date, rating.mood
having rating.mood <= avg(rating.mood)
```

i.
```
Select restaurant.name, location.phone_number, restaurant.category
from restaurant join location on restaurant.business_id=location.business_id
where not exist(
        select *
        from restaurant join rating on restaurant.business_id=rating.business_id
        where date != '2015-01' )
```

j.
```
select  cateType.reputation , cateType.name , cateType.ftype from rating join (
        select rater.reputation , rater.name, unnest(restaurant.food_type) as
ftype , restaurant.business_id
                from rater join (
        restaurant join rating
        on rating.business_id = restaurant.business_id
        ) on rater.user_id = rating.user_id
        group by( rater.reputation , rater.name , ftype ,restaurant.business_id)
) as cateType on rating.business_id = cateType.business_id
group by( cateType.ftype, cateType.reputation , cateType.name ,rating.mood)
having ( rating.mood >= avg(rating.mood))
```

<u>Raters and their ratings</u>

k.
```
select rater.join_date , rater.name , rater.reputation , detail.rname
from rater join
( select rating.user_id , rating.mood as review ,
        details.rname ,details.avr
        from rating join (
        select unnest(restaurant.food_type) as utype, restaurant.business_id,
                restaurant.name as rname , rating.date as rd ,rating.user_id as
uid ,
                avg(rating.mood) as avr from restaurant join rating on
        restaurant.business_id = rating.business_id
        group by ( utype , rating.date  , rating.user_id ,
rname ,restaurant.business_id)
) as details on rating.business_id = details.business_id) as detail
on detail.user_id = rater.user_id
group by ( rater.user_id , detail.review ,detail.rname,detail.avr )
having( detail.review >= detail.avr)
```

l.
```
select  rater.name , rater.reputation , detail.rname as restaurant_name,
detail.date as rating_date
from rater join
( select rating.user_id , rating.mood as review ,
        details.rname ,details.avr , rating.date
        from rating join (
        select unnest(restaurant.food_type) as utype, restaurant.business_id,
                restaurant.name as rname , rating.date as rd ,rating.user_id as
uid ,
                avg(rating.mood) as avr from restaurant join rating on
        restaurant.business_id = rating.business_id
        group by ( utype , rating.date  , rating.user_id ,
rname ,restaurant.business_id)
) as details on rating.business_id = details.business_id) as detail
on detail.user_id = rater.user_id
group by ( rater.user_id , detail.review ,detail.rname,detail.avr ,detail.date)
having( detail.review >= detail.avr)
```

m.
```
Select rater.name, rater.email, count(rating) as result
From rater join rating on rater.user_id=rating.user_id
Where result <= (select rating.mood
                  From rater join rating on rater.user_id=rating.user_id
                  Where rater.name = 'John')
Group by rater.name, rater.email
```

5. We use html, css combined with bootstrap on front end.