**CSI2132 Database I**

**2018winter**

**“Restaurant Rating Database”**

**Project Report**

**Weizhe Liang 8136867**

**Zaeem Qureshi 7320339**

**Xiaoxin Zhou 7957115**

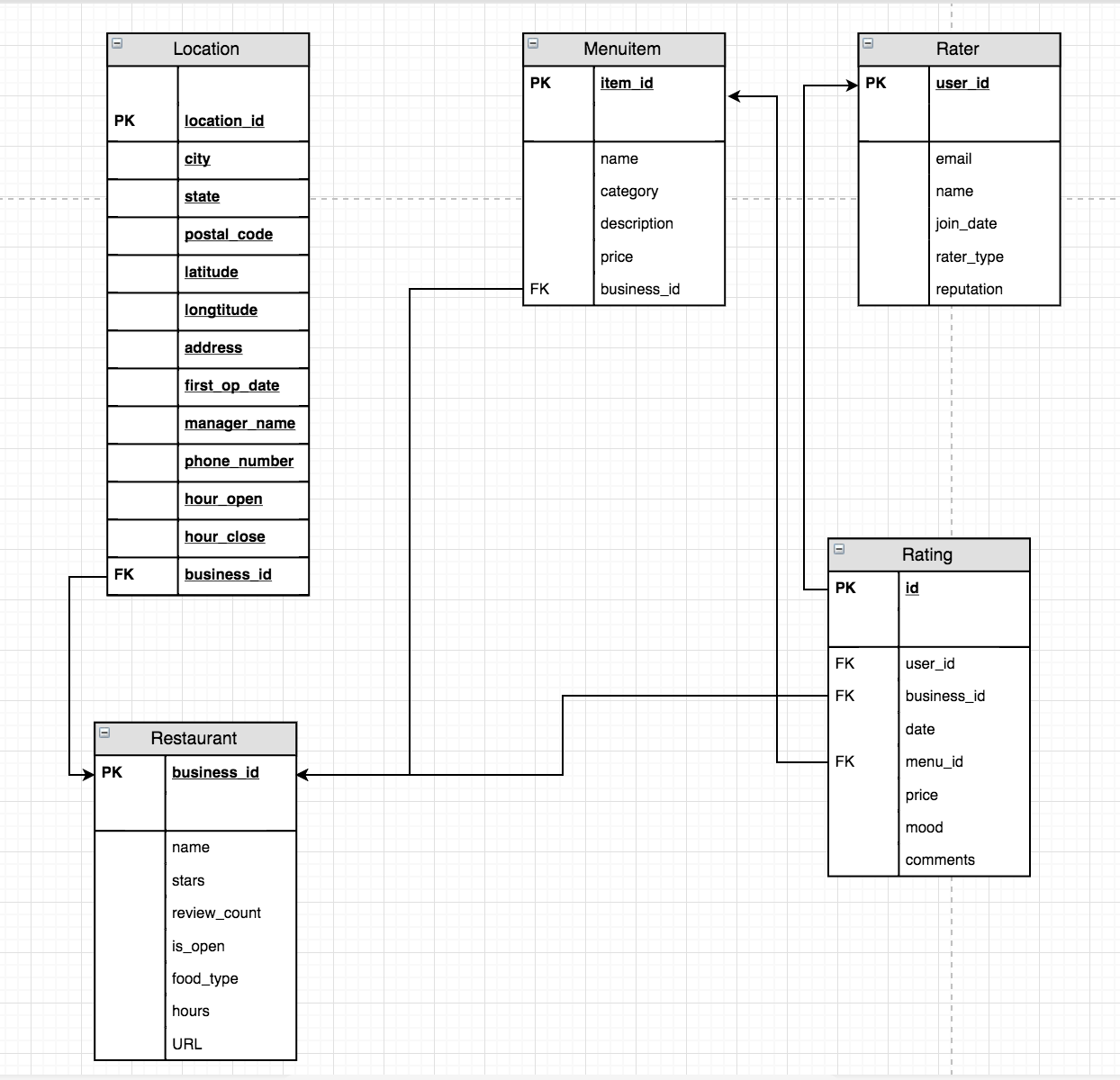
**Introduction**

**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](http://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’**



|  |  |
| --- | --- |
| 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)  ); |  |

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

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

1. Queries

Restaurants and menus

1. Select \* from restaurant where business\_id = ‘’
2. Select \* from menuitem where business\_id = ‘’
3. Select manager\_name, first\_op\_date from location where business\_id = ‘’
4. Select location.manager\_name, location.hour\_open, restaurant.URL

From location join restaurant on location.business\_id = restaurant.business\_id

Join menuitem

Where

1. 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

Ratings of restaurants

1. 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

1. 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’ )

1. 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)

1. 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’ )

1. 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))

Raters and their ratings

1. 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)

1. 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)

1. check HTMl
2. 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

1. Select rater.name, rater.rater\_type, rater.em
2. We use html, css combined with bootstrap on front end.