# Recommendation

Zhejing Hu
Yang Lyu
Lan Jiang
Chi Ma

# Motivation

- Many people use yelp today for convenience
- Yelp make recommendation based on location rather than your taste
- Serve as an alternative to yelp core functionality

# Project Outline

**Data Preprocessing**

1

Remove irrelevant columns and filtering.

**Recommendation Engine**

2

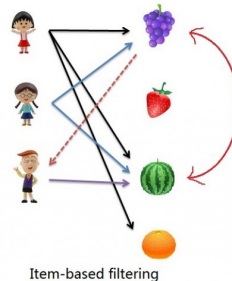Building different recommendation algorithms and evaluate performance

**Ui Design**

3

Front-end deployment and design

python anywhere

Cloud

Flask
web development,
one drop at a time

Front-end

MySQL®

Database

Collaborative Filtering          Content Based          SVD

User-based filtering          Item-based filtering

$$M = U\Sigma V^{*}$$

Back-end

Recommendation

# MySQL database

```
5    (        SELECT user_id
6             FROM df_review2
7             GROUP BY user_id
8             HAVING COUNT(*) > 10
9    )
10   ORDER BY user_id
11   #import df_review
12
13   CREATE TABLE df_review(business_id varchar(50),cool int, date date, funny int, review_id varchar(50),stars int, text text, useful int, user_id varchar(50));
14
15
16   #import df_business
17
18   CREATE TABLE df_business(address varchar(50), attributes text, business_id varchar(50), categories text, city varchar(20), hours text, is_open int, latitude float, longitude float, name varchar(100),
     neighborhood varchar(500), postal_code varchar(50),review_count int,stars float,state varchar(2), PRIMARY KEY(business_id));
19
20   #check drop null df_business("categories")
21
22   SELECT * FROM df_business WHERE categories IS NULL OR categories = "";
23
24   #Create new df_vegas
25   CREATE TABLE df_vegas LIKE df_business
26
27   INSERT INTO df_vegas(address, attributes, business_id, categories, city, hours, is_open, latitude, longitude, name, neighborhood, postal_code, review_count, stars, state)
28   SELECT address, attributes, business_id, categories, city, hours, is_open, latitude, longitude, name, neighborhood, postal_code, review_count, stars, state
29   FROM df_business
30   WHERE city = "Las Vegas"
31   AND categories LIKE "%Restaurants%"
32   AND review_count > 10;
33
34   #Create table df_left like df_vegas with features: business_id,name,stars,address,longitude,latitude
35   |
36   CREATE TABLE df_left(business_id varchar(50), name varchar(100),avg_stars float,address varchar(50),longitude float, latitude float, PRIMARY KEY(business_id));
37
38   # fill values
39
40   INSERT INTO df_left(address,business_id,latitude,longitude,name,avg_stars)
41   SELECT address,business_id,latitude,longitude,name,stars
42   FROM df_vegas;
43
44   #Create Table df_right like df_review with features: business_id, review_id, stars, text, user_id
45
46   CREATE TABLE df_right(business_id varchar(50), review_id varchar(50), stars float,text text,user_id varchar(50));
47
48   #fill values
49
50   INSERT INTO df_right(business_id,review_id,stars,text,user_id)
51   SELECT business_id,review_id,stars,text,user_id
52   FROM df_review;
53
54
```
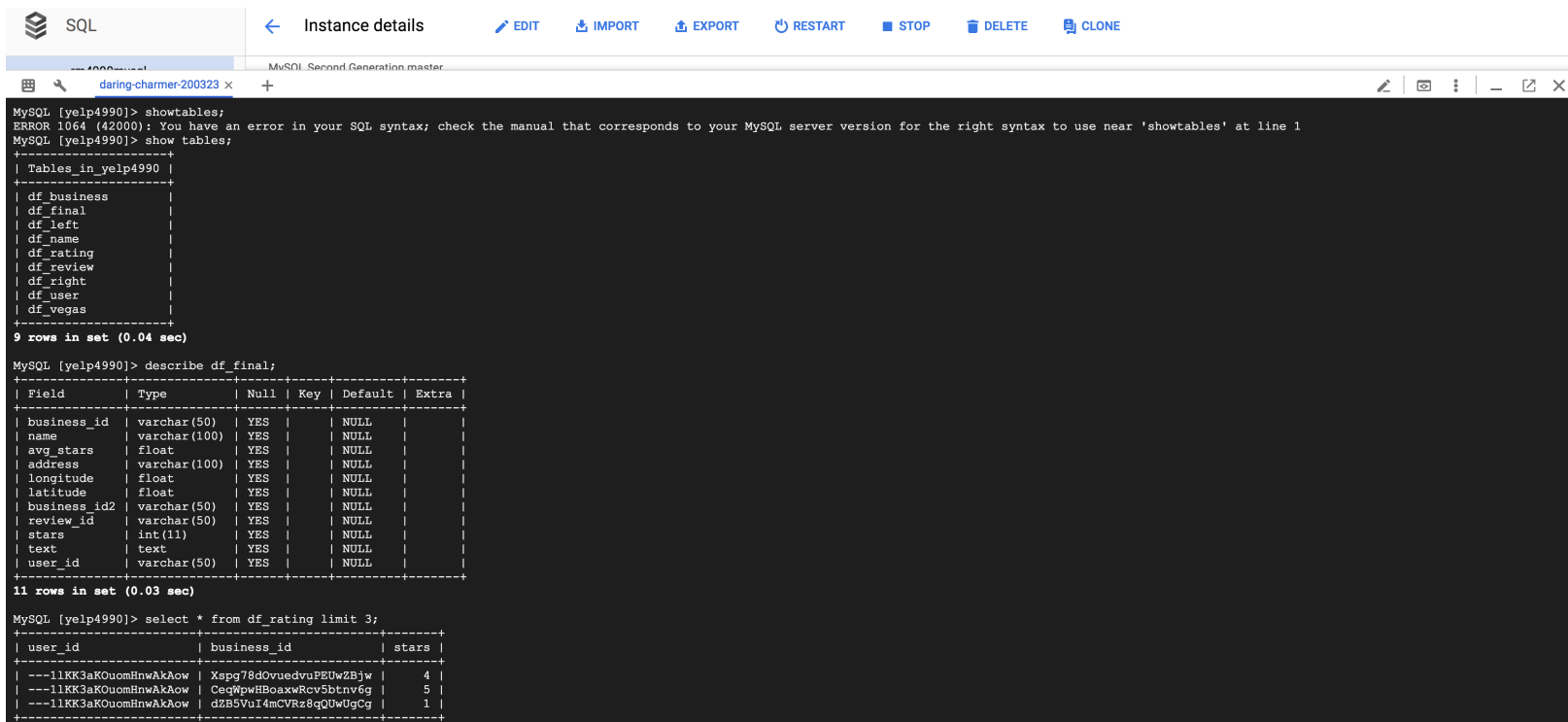
Type to enter a caption.

# MySQL database

```
48    WRITE VALUES
49
50    INSERT INTO df_right(business_id,review_id,stars,text,user_id)
51    SELECT business_id,review_id,stars,text,user_id
52    FROM df_review;
53
54
55    #CREATE df_final inner join df_left & df_right;
56    CREATE TABLE df_final(business_id varchar(50), name varchar(100), avg_stars float, address varchar(100), longitude float, latitude float, business_id2 varchar(50),review_id varchar(50),stars int,text
      text,user_id varchar(50));
57
58    INSERT INTO df_final(business_id, name, avg_stars, address, longitude, latitude, business_id2, review_id, stars, text, user_id)
59    SELECT *
60    FROM df_left
61    INNER JOIN df_right
62    On df_left.business_id = df_right.business_id2;
63
64    #CREATE df_rating
65
66    CREATE TABLE df_rating(user_id varchar(50),business_id varchar(50),stars float);
67
68    INSERT INTO df_rating(user_id, business_id, stars)
69    SELECT user_id, business_id, stars
70    FROM df_final
71    WHERE user_id IN
72        (       SELECT user_id
73                FROM df_final
74                GROUP BY user_id
75                HAVING COUNT(*) > 10
76        )
77     And  business_id IN
78        (       SELECT business_id
79                FROM df_final
80                GROUP BY business_id
81                HAVING COUNT(*) > 15
82        )
83    ORDER BY user_id
84
85
86    #CREATE df_name
87
88    CREATE TABLE df_name(business_id varchar(50),name varchar(100));
89
90    INSERT INTO df_name(business_id, name)
91    SELECT business_id, name
92    FROM df_final
93    WHERE user_id IN
94        (       SELECT user_id
95                FROM df_final
96                GROUP BY user_id
97                HAVING COUNT(*) > 10
98        )
99     And  business_id IN
100       (       SELECT business_id
101               FROM df_final
```

Type to enter a caption.

# MySQL database

[Demo](#)