Milestone 3 Task A

Group Members: Siyu Xue (leader), Wanyin Hu

1. User

Since some users can share the house address and the username, user_id functionally determines user_name and user_add. Since each user will have a unique phone number and email address, user_tel or user_email could potentially determine the user_id. However, since the value of user_tel and user_email can be null, user_id functionally determines user_email and user_tel.

```
mysql> describe user;
+------------+--------------+------+-----+----------+-------+
| Field      | Type         | Null | Key | Default  | Extra |
+------------+--------------+------+-----+----------+-------+
| user_id    | int(11)      | NO   | PRI | NULL     |       |
| user_name  | varchar(50)  | YES  |     | New User |       |
| user_tel   | int(11)      | YES  |     | NULL     |       |
| user_add   | varchar(250) | YES  |     | NULL     |       |
| user_email | varchar(100) | YES  |     | NULL     |       |
+------------+--------------+------+-----+----------+-------+
5 rows in set (0.00 sec)
```

2. Collection

Since a user can create multiple collections, owner_id (user_id) cannot functionally determine coll_id. However, coll_id can functionally determine coll_owner. Since coll_id is the primary key and collations can share the same collection name, coll_id functionally determines coll_name.

```
mysql> describe collection;
+------------+-------------+------+-----+----------------+-------+
| Field      | Type        | Null | Key | Default        | Extra |
+------------+-------------+------+-----+----------------+-------+
| coll_id    | int(11)     | NO   | PRI | NULL           |       |
| coll_name  | varchar(50) | YES  |     | New Collection |       |
| coll_owner | int(11)     | YES  | MUL | NULL           |       |
+------------+-------------+------+-----+----------------+-------+
3 rows in set (0.00 sec)
```

3. Rest_List

The initial schema of rest_list was a part of the collection. Due to the violation of 1NF, which multiple values are not allowed in the table, we separated the restaurant list from the collection table. There are only two attributes in the rest_list table: coll_id and rest_id. Because a collection can include multiple restaurants, and a restaurant can appear in multiple collections, coll_id cannot functionally determine rest_id, nor can rest_id functionally determine coll_id. We decided to keep it in this form instead of BCNF is because this table preserve the relationaship between restaurant and collection most efficiently. Although BCNF can be achieved simply by adding a "combination_id" attribute, there is no need for it; thus it would be redundant.

```
mysql> describe rest_list;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| coll_id | int(11) | NO   | MUL | NULL    |       |
| rest_id | int(11) | NO   | MUL | NULL    |       |
+---------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

4. Review

Each review has a unique review_id. It can only describes one restaurant, written by one user, have one date and rating, and one content if there is any. Therefore, review_id functionally determines rest_id, rate, user_id, content and review_date.

```
mysql> describe review;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| review_id   | int(11)     | NO   | PRI | NULL    |       |
| rest_id     | int(11)     | NO   |     | NULL    |       |
| rate        | int(11)     | NO   |     | NULL    |       |
| user_id     | int(11)     | NO   | MUL | NULL    |       |
| content     | varchar(250) | YES  |     | NULL    |       |
| review_date | date        | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.07 sec)
```

5. Restaurant

We deleted the open time from initial schema. Since each restaurant has a unique rest_id, and at the same time, each restaurant has its own address, telephone number, and address, the super key of the Restaurant table consists of rest_id, rest_tel, and rest_add. We assume it is possible for restaurants to have the same name. Therefore, any single attribute in the super key set can functionally determines the rest of the table.

```
mysql> describe restaurant;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| rest_id       | int(11)     | NO   | PRI | NULL    |       |
| rest_tel      | int(11)     | YES  |     | NULL    |       |
| rest_add      | varchar(250)| NO   |     | NULL    |       |
| rest_name     | varchar(250)| NO   |     | NULL    |       |
| rest_type     | int(11)     | YES  |     | NULL    |       |
| rest_takeout  | int(11)     | YES  |     | NULL    |       |
| rest_delivery | int(11)     | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

6. Type

The type table was not in the initial schema. We created this table because restaurant types are relatively fixed — there are only certain types exist. We create 21 different types in the table, each type has a corresponding type_id. Thus, in the restaurant table, each restaurant has a type_id instead of an actual type that is in English.

```
mysql> describe type;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| type_id | int(11)     | NO   | PRI | NULL    |       |
| type    | varchar(150)| NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```