Project - 2

Name: Raghav Jindal (rj2571) & Manik Goyal (mg4106)
Name of the PostgreSQL account where your database is on our server: mg4106

The 3 new features that we added are as follows:

1. **About me for a critic:** Every critic has his own 'profile page' where the users can have a look at the reviews made by the critic, his favorite restaurants and the likes that the critic has for each of his review and the critic score. It makes sense that a critic should also be able to mention something about himself. So we provide a new attribute to each critic of the type *text*. This is the *about* attribute. So, we add a new column to the critics database:

   ALTER TABLE critic ADD column about TEXT;

   Then we manually populated this new column for every critic. An example update query is given below:

   UPDATE test_critic set about = 'Hi. My name is Jennifer. I am a food critic. My favourite cuisines are Indian and Chinese. I like to eat and try out new types of dishes.' WHERE userid = 'c1';

   Since, it is a text attribute, we provide full text based search on the *about* attribute of the critics. For example, if we want to find the critics who have mentioned something about *'*indian*'* in their about me.

   *SELECT * from test_critic*
   WHERE to_tsvector('english', about) @@ to_tsquery('english', 'indians');

   This will return all the critics who have mentioned something about 'indian'.

2. **Like history for each user:**
   All the users, be it critic or a normal user have the ability to like a review. Earlier, the only thing that we were able to check was that the user should not be able to like his own review since every review is associated with a user. However, this suffered from a critical issue that a review can be liked multiple times by a single user, if that review was not made by the user.

   To prevent this, we created an array data type for each user which will store review id corresponding to the reviews that a user has already liked. The query for this is:
   ALTER table users ADD column like_history varchar(50)[ ];

   So, if we are given this, we can ensure that a review would only be liked once by a user. The following is the query which does exactly this:

   Query when a user 'n1' likes a review '1':

```
// checks that the review should not have been liked already
UPDATE reviews_gives r set likes = likes + 1
FROM users u
WHERE u.userid = 'n1' and r.revid = '1' and r.revid != any(u.like_history);

// adds the review to the like history of the user if it does not exist
UPDATE users
SET like_history = ARRAY(select DISTINCT UNNEST(like_history || '{1}'))
WHERE userid = 'n1';
```

3. **Trigger to update the restaurant rating whenever a new review is made:**

Earlier, whenever a new review was given by a normal user or a critic, we were manually updating the restaurant rating using a separate SQL query.
However, it makes more sense for the rating of a restaurant to be updated automatically when a new review is made. This can be done by creating a trigger. So, whenever a new review is given, the trigger would update the rating of the restaurant (both user and critic rating as we have 2 separate ratings for restaurant given by normal users and critics respectively). The sql code is as follows:

```
CREATE TRIGGER UPDATE_RATING
AFTER INSERT on Reviews_gives
FOR EACH ROW
EXECUTE PROCEDURE update_rating();

CREATE OR REPLACE FUNCTION update_rating()
RETURNS trigger AS
$$
BEGIN
        UPDATE Restaurant_owns
        SET u_rating = (SELECT ROUND(AVG(rating), 2)
                        FROM reviews_gives
                        WHERE userid LIKE 'n%' GROUP BY new.rid),
                c_rating = (SELECT ROUND(AVG(rating), 2)
                        FROM reviews_gives
                        WHERE userid LIKE 'c%' GROUP BY new.rid);
RETURN new;
END;
$$
Language plpgsql;
```

So when the query below inserts the new review, the trigger would automatically update the rating of the corresponding restaurant.

```
INSERT into reviews_gives VALUES ('r150', 'I will suggest everyone to try the
cheese fries and shake-shack burger with a drink of choice, they are the best combo',
10, 5, 'n11', '1');
```