ECE 356 Lab2 Yelp

Lab section 206
Group 10

Victor Yan
Lun Jing

(a) Which user has written the greatest number of reviews?

```
EXPLAIN
SELECT name FROM user
ORDER BY review_count DESC
LIMIT 1;
```

Before adding indexes:

Running time:

```
+--------+
| name   |
+--------+
| Victor |
+--------+
1 row in set (0.70 sec)
```

Explain:

```
+----+-------------+-------+------+---------------+------+---------+------+---------+----------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows    | Extra          |
+----+-------------+-------+------+---------------+------+---------+------+---------+----------------+
|  1 | SIMPLE      | user  | ALL  | NULL          | NULL | NULL    | NULL | 1021667 | Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+---------+----------------+
1 row in set (0.08 sec)
```

Since this query is trying to sort the review_count, it will be helpful to add the review_count in the index.

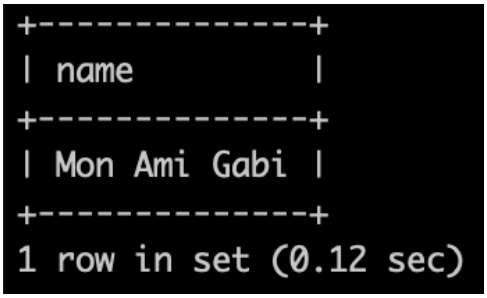
```
CREATE INDEX user_1 on user (review_count);
```

(b) Which business has received the greatest number of reviews?

```
EXPLAIN
SELECT name FROM business
ORDER BY review_count DESC
LIMIT 1;
```

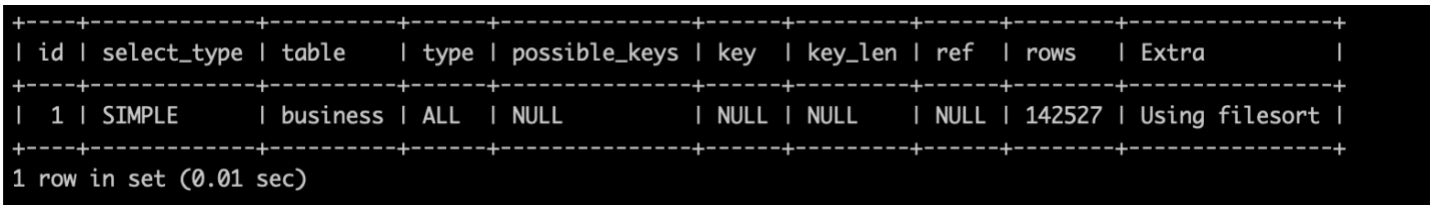
Before adding indexes:

Running time:



```
+--------------+
| name         |
+--------------+
| Mon Ami Gabi |
+--------------+
1 row in set (0.12 sec)
```

Explain:



```
+----+-------------+----------+------+---------------+------+---------+------+--------+----------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows   | Extra          |
+----+-------------+----------+------+---------------+------+---------+------+--------+----------------+
|  1 | SIMPLE      | business | ALL  | NULL          | NULL | NULL    | NULL | 142527 | Using filesort |
+----+-------------+----------+------+---------------+------+---------+------+--------+----------------+
1 row in set (0.01 sec)
```

Similar as previous one, we also add review_count into the index.

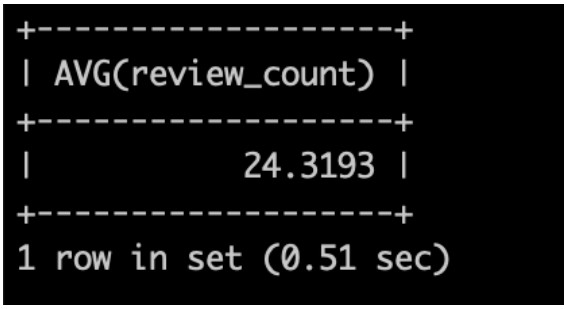
```
CREATE INDEX business_1 on business (review_count);
```

(c) What is the average number of reviews written by users?

```
EXPLAIN
SELECT AVG(review_count) FROM user;
```

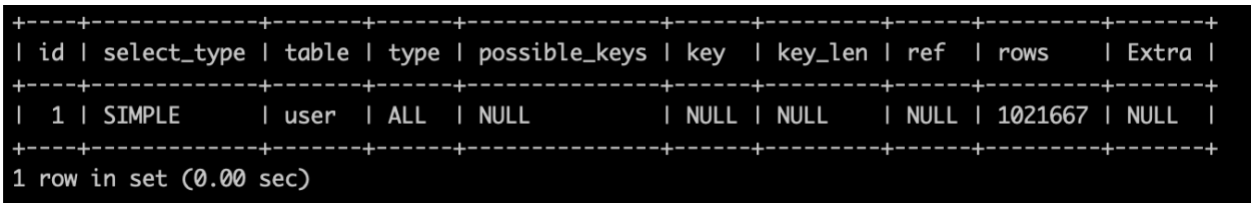
Before adding indexes:

Running time:



```
+-------------------+
| AVG(review_count) |
+-------------------+
|           24.3193 |
+-------------------+
1 row in set (0.51 sec)
```

Explain:



```
+----+-------------+-------+------+---------------+------+---------+------+---------+-------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows    | Extra |
+----+-------------+-------+------+---------------+------+---------+------+---------+-------+
|  1 | SIMPLE      | user  | ALL  | NULL          | NULL | NULL    | NULL | 1021667 | NULL  |
+----+-------------+-------+------+---------------+------+---------+------+---------+-------+
1 row in set (0.00 sec)
```

Since this query is aggregation, which always need to go through all input once, there is no need to add index for it.

(d) The average rating written by a user can be determined in two ways:
a. By direct reading from the Users table “average stars” column
b. By computing an average of the ratings issued by a user for businesses reviewed
For how many users is the difference between these two amounts larger than 0.5?

```
EXPLAIN
SELECT COUNT(*) FROM
(SELECT user_id, average_stars FROM user) as A inner join
(SELECT AVG(stars) as avg_stars, user_id FROM review
GROUP BY user_id ) as B
USING (user_id)
WHERE ABS(A.average_stars - B.avg_stars) > 0.5;
```

Before adding indexes:

Running time:

```
+----------+
| COUNT(*) |
+----------+
|       66 |
+----------+
1 row in set (15.05 sec)
```

Explain:

```
+----+-------------+------------+------+---------------+-------------+---------+-----------+---------+---------------------------------+
| id | select_type | table      | type | possible_keys | key         | key_len | ref       | rows    | Extra                           |
+----+-------------+------------+------+---------------+-------------+---------+-----------+---------+---------------------------------+
|  1 | PRIMARY     | <derived2> | ALL  | NULL          | NULL        | NULL    | NULL      | 1021667 | NULL                            |
|  1 | PRIMARY     | <derived3> | ref  | <auto_key0>   | <auto_key0> | 22      | A.user_id |      10 | Using where                     |
|  3 | DERIVED     | review     | ALL  | NULL          | NULL        | NULL    | NULL      | 1655155 | Using temporary; Using filesort |
|  2 | DERIVED     | user       | ALL  | NULL          | NULL        | NULL    | NULL      | 1021667 | NULL                            |
+----+-------------+------------+------+---------------+-------------+---------+-----------+---------+---------------------------------+
4 rows in set (0.01 sec)
```

Since this query joins two tables `USING (user_id)`, we can add indexes with user_id for each of them.

```
CREATE INDEX user_2 on user (user_id);
CREATE INDEX review_1 on review (user_id);
```

(e) What fraction of users have written more than 10 reviews?

```
EXPLAIN
SELECT
(SELECT count(*) FROM user Where review_count > 10)
/
(SELECT count( distinct user_id) FROM user) as fraction;
```

Before adding indexes:

Running time:

```
+----------+
| fraction |
+----------+
|   0.3311 |
+----------+
1 row in set (1.31 sec)
```

Explain:

```
+----+-------------+-------+-------+---------------+---------+---------+------+---------+----------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows    | Extra          |
+----+-------------+-------+-------+---------------+---------+---------+------+---------+----------------+
|  1 | PRIMARY     | NULL  | NULL  | NULL          | NULL    | NULL    | NULL |    NULL | No tables used |
|  3 | SUBQUERY    | user  | index | PRIMARY       | PRIMARY | 22      | NULL | 1021667 | Using index    |
|  2 | SUBQUERY    | user  | ALL   | NULL          | NULL    | NULL    | NULL | 1021667 | Using where    |
+----+-------------+-------+-------+---------------+---------+---------+------+---------+----------------+
3 rows in set (0.00 sec)
```

Similar as previous one, but this time we need count the number of entries that have review_count > 10 seperately. So we should create a seperate index for review_count only.

```
CREATE INDEX user_3 on user (review_count);
```

(f) What is the average length of their reviews?

```
EXPLAIN
SELECT AVG(LENGTH(text)) FROM user as U inner join review as R
USING (user_id)
WHERE U.review_count > 10;
```

Before adding indexes:

Running time:

```
+--------------------+
| AVG(LENGTH(text)) |
+--------------------+
|          698.7808 |
+--------------------+
1 row in set (54.78 sec)
```

Explain:

```
+----+-------------+-------+--------+---------------+---------+---------+----------------+---------+-------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref            | rows    | Extra       |
+----+-------------+-------+--------+---------------+---------+---------+----------------+---------+-------------+
|  1 | SIMPLE      | R     | ALL    | NULL          | NULL    | NULL    | NULL           | 1655155 | NULL        |
|  1 | SIMPLE      | U     | eq_ref | PRIMARY       | PRIMARY | 22      | Yelp.R.user_id |       1 | Using where |
+----+-------------+-------+--------+---------------+---------+---------+----------------+---------+-------------+
2 rows in set (0.01 sec)
```

Since all the text will be accessed at least once, we cannot add any index for it. However, we will need to check review table and select the entries with review_count > 10 and then join the user_id, we can use the index with review_count and user_id

```
CREATE INDEX user_4 on user (review_count, user_id);
```