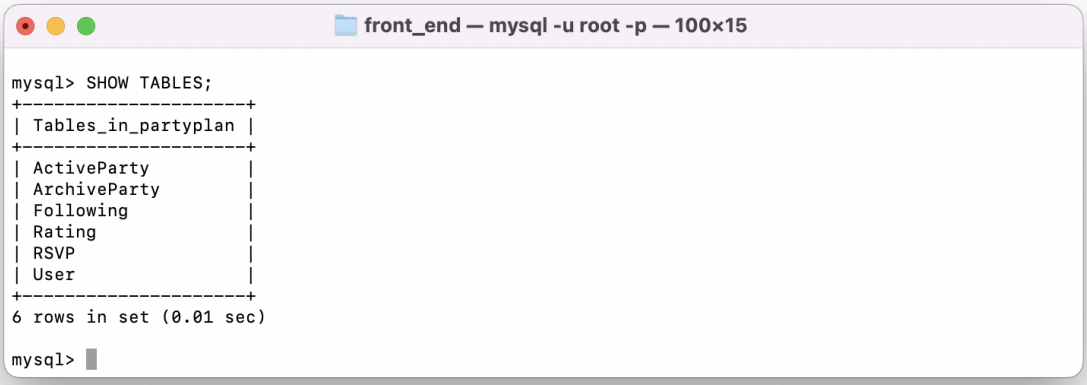
Team 018 - SQLMasters Database Implementation and Indexing

Vikrant Banerjee, Naman Jain, Carson Zuniga, Jamiel Abed

Database Implementation

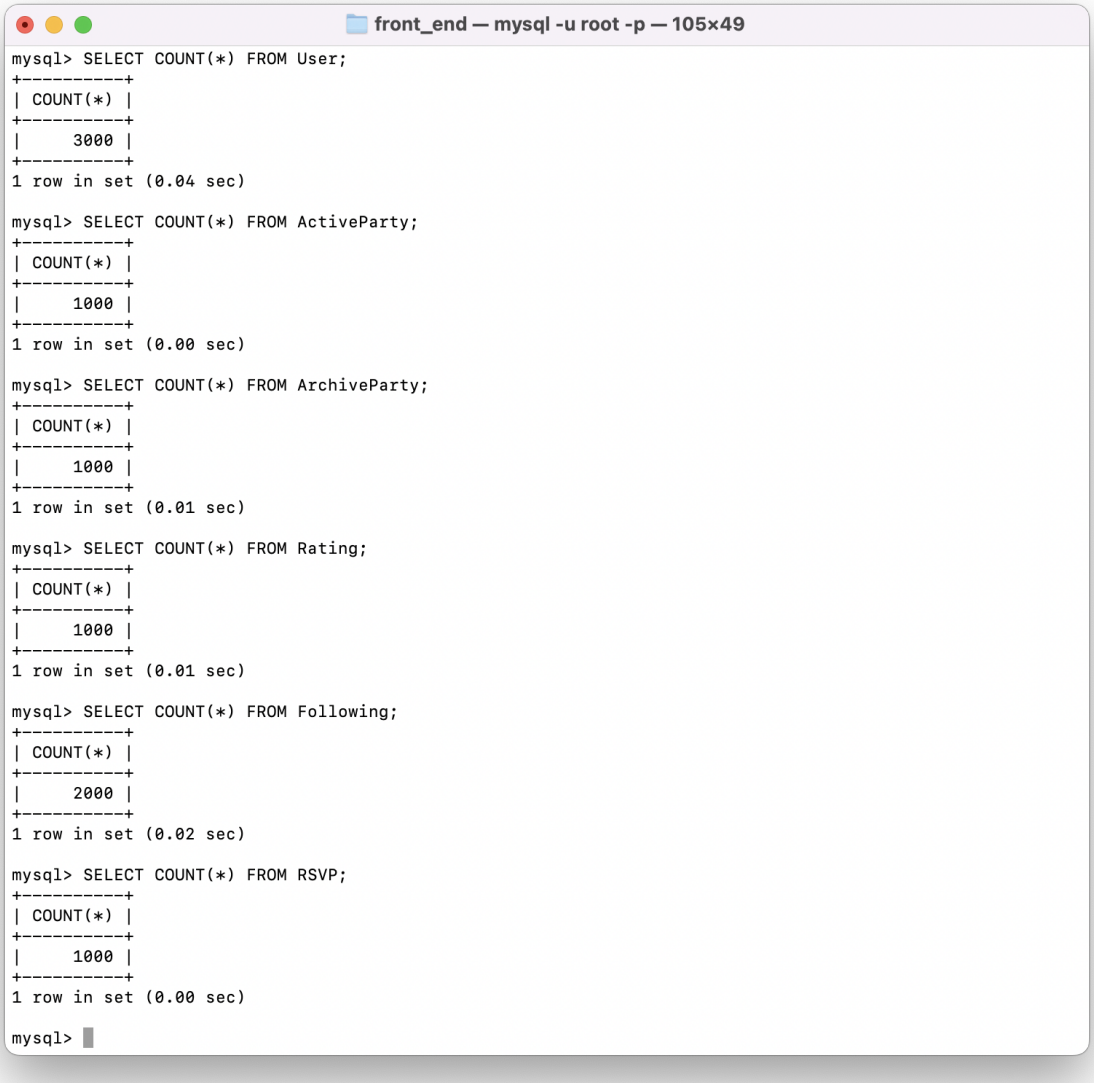


Proof of implementing the database tables locally or on GCP, you should provide a screenshot of the connection (i.e. showing your terminal information).

DDL Commands:

```
CREATE TABLE User (
        userID INT UNSIGNED NOT NULL PRIMARY KEY,
        first VARCHAR(63) NOT NULL,
        last VARCHAR(63) NOT NULL,
        email VARCHAR(63) NOT NULL,
        phone_number BIGINT UNSIGNED NOT NULL
);

CREATE TABLE ActiveParty (
        partyID INT UNSIGNED NOT NULL PRIMARY KEY,
        userID INT UNSIGNED NOT NULL,
        latitude REAL NOT NULL,
        longitude REAL NOT NULL,
        tags JSON,
        capacity INT NOT NULL,
        start_time DATE NOT NULL,
        end_time DATE NOT NULL,
        FOREIGN KEY(userID) REFERENCES User(userID) ON DELETE CASCADE
);
```

```
CREATE TABLE ArchiveParty (
        partyID INT UNSIGNED NOT NULL PRIMARY KEY,
        userID INT UNSIGNED NOT NULL,
        latitude REAL NOT NULL,
        longitude REAL NOT NULL,
        tags JSON,
        capacity INT NOT NULL,
        start_time DATE NOT NULL,
        end_time DATE NOT NULL,
        FOREIGN KEY(userID) REFERENCES User(userID) ON DELETE CASCADE
);

CREATE TABLE Rating (
        ratingID INT UNSIGNED NOT NULL PRIMARY KEY,
        safety INT UNSIGNED NOT NULL,
        fun INT UNSIGNED NOT NULL,
        userID INT UNSIGNED NOT NULL,
        FOREIGN KEY(userID) REFERENCES User(userID) ON DELETE CASCADE
);

CREATE TABLE Following (
        Followingid INT UNSIGNED NOT NULL PRIMARY KEY,
        userID INT UNSIGNED NOT NULL,
        followsID INT UNSIGNED NOT NULL,
        FOREIGN KEY(userID) REFERENCES User(userID) ON DELETE CASCADE,
        FOREIGN KEY(followsID) REFERENCES User(userID) ON DELETE CASCADE
);

CREATE TABLE RSVP (
        RSVPid INT UNSIGNED NOT NULL PRIMARY KEY,
        partyID INT UNSIGNED NOT NULL,
        userID INT UNSIGNED NOT NULL,
        reponse CHAR(11),
        FOREIGN KEY(userID) REFERENCES User(userID) ON DELETE CASCADE,
        FOREIGN KEY(partyID) REFERENCES ActiveParty(partyID) ON DELETE CASCADE
);
```

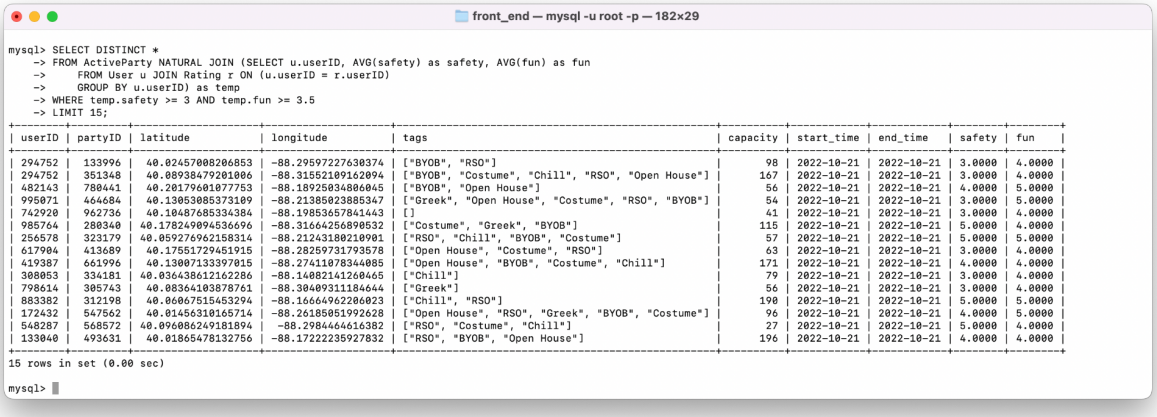


Proof for inserting at least 1000 rows in the tables.

Advanced Queries

Query 1:

```
SELECT DISTINCT *
FROM ActiveParty NATURAL JOIN (SELECT u.userID, AVG(safety) as safety, AVG(fun) as fun
        FROM User u JOIN Rating r ON (u.userID = r.userID)
        GROUP BY u.userID) as temp
WHERE temp.safety >= 3 AND temp.fun >= 3.5
LIMIT 15;
```

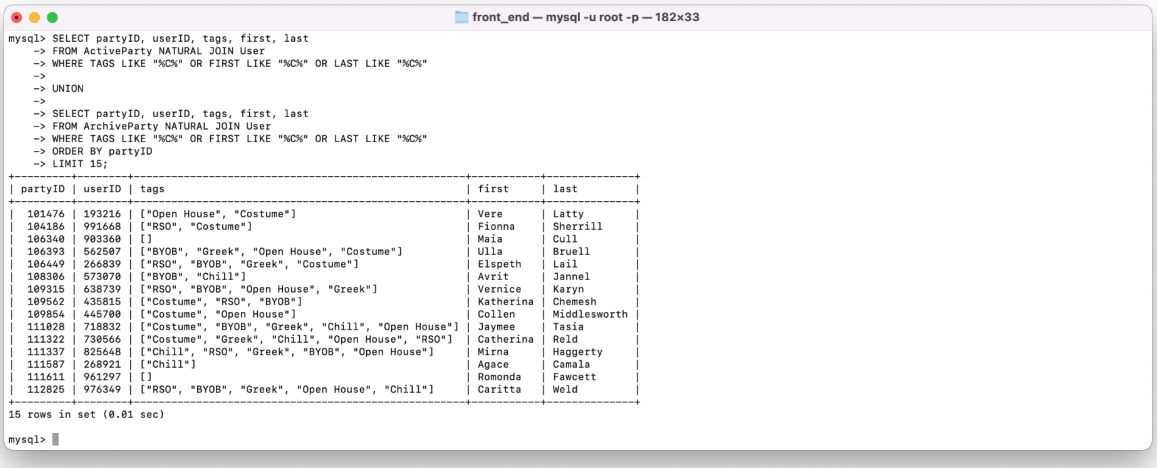



```
mysql> SELECT DISTINCT *
    -> FROM ActiveParty NATURAL JOIN (SELECT u.userID, AVG(safety) as safety, AVG(fun) as fun
    ->     FROM User u JOIN Rating r ON (u.userID = r.userID)
    ->     GROUP BY u.userID) as temp
    -> WHERE temp.safety >= 3 AND temp.fun >= 3.5
    -> LIMIT 15;
+--------+---------+-------------------+--------------------+-----------------------------------------------------+----------+------------+------------+--------+--------+
| userID | partyID | latitude          | longitude          | tags                                                | capacity | start_time | end_time   | safety | fun    |
+--------+---------+-------------------+--------------------+-----------------------------------------------------+----------+------------+------------+--------+--------+
| 294752 |  133996 |  40.02457008206853 | -88.29597227630374 | ["BYOB", "RSO"]                                     |       98 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 294752 |  351348 |  40.08938479201006 | -88.31552109162094 | ["BYOB", "Costume", "Chill", "RSO", "Open House"]   |      167 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 482143 |  780441 |  40.20179601077753 | -88.18925034806045 | ["BYOB", "Open House"]                              |       56 | 2022-10-21 | 2022-10-21 | 4.0000 | 5.0000 |
| 995071 |  464684 |  40.13053085373109 | -88.21385023885347 | ["Greek", "Open House", "Costume", "RSO", "BYOB"]   |       54 | 2022-10-21 | 2022-10-21 | 3.0000 | 5.0000 |
| 742920 |  962736 |  40.10487685334384 | -88.19853657841443 | []                                                  |       41 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 985764 |  280340 | 40.178249094536696 | -88.31644256890532 | ["Costume", "Greek", "BYOB"]                        |      115 | 2022-10-21 | 2022-10-21 | 5.0000 | 4.0000 |
| 256578 |  323179 | 40.859276962158314 | -88.21243180210901 | ["RSO", "Chill", "BYOB", "Costume"]                 |       57 | 2022-10-21 | 2022-10-21 | 5.0000 | 5.0000 |
| 617904 |  413689 |  40.17551729451915 | -88.28259731793578 | ["Open House", "Costume", "RSO"]                    |       63 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 419387 |  661996 |  40.13007133397015 | -88.27411078344085 | ["Open House", "BYOB", "Costume", "Chill"]          |      171 | 2022-10-21 | 2022-10-21 | 4.0000 | 4.0000 |
| 308053 |  334181 | 40.036438612162286 | -88.14082141260465 | ["Chill"]                                           |       79 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 798614 |  305743 |  40.08364103878761 | -88.30409311184644 | ["Greek"]                                           |       56 | 2022-10-21 | 2022-10-21 | 3.0000 | 4.0000 |
| 883382 |  312198 |  40.06067515453294 | -88.16664962206023 | ["Chill", "RSO"]                                    |      190 | 2022-10-21 | 2022-10-21 | 5.0000 | 5.0000 |
| 172432 |  547562 |  40.01456310165714 | -88.26185051992628 | ["Open House", "RSO", "Greek", "BYOB", "Costume"]   |       96 | 2022-10-21 | 2022-10-21 | 4.0000 | 5.0000 |
| 548287 |  568572 | 40.096086249181894 |  -88.2984464616382 | ["RSO", "Costume", "Chill"]                         |       27 | 2022-10-21 | 2022-10-21 | 5.0000 | 4.0000 |
| 133040 |  493631 |  40.01865478132756 | -88.17222235927832 | ["RSO", "BYOB", "Open House"]                       |      196 | 2022-10-21 | 2022-10-21 | 4.0000 | 4.0000 |
+--------+---------+-------------------+--------------------+-----------------------------------------------------+----------+------------+------------+--------+--------+
15 rows in set (0.00 sec)

mysql>
```

Query 2:
SELECT partyID, userID, tags, first, last
FROM ActiveParty NATURAL JOIN User
WHERE TAGS LIKE "%C%" OR FIRST LIKE "%C%" OR LAST LIKE "%C%"

UNION

SELECT partyID, userID, tags, first, last
FROM ArchiveParty NATURAL JOIN User
WHERE TAGS LIKE "%C%" OR FIRST LIKE "%C%" OR LAST LIKE "%C%"
LIMIT 15;



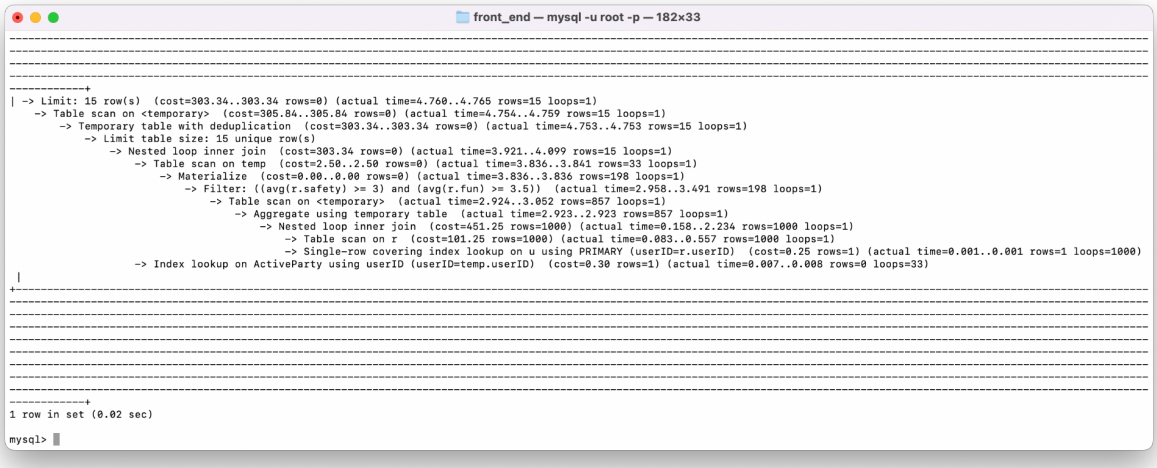

```
mysql> SELECT partyID, userID, tags, first, last
    -> FROM ActiveParty NATURAL JOIN User
    -> WHERE TAGS LIKE "%C%" OR FIRST LIKE "%C%" OR LAST LIKE "%C%"
    ->
    -> UNION
    ->
    -> SELECT partyID, userID, tags, first, last
    -> FROM ArchiveParty NATURAL JOIN User
    -> WHERE TAGS LIKE "%C%" OR FIRST LIKE "%C%" OR LAST LIKE "%C%"
    -> ORDER BY partyID
    -> LIMIT 15;
+---------+--------+-----------------------------------------------------+-----------+--------------+
| partyID | userID | tags                                                | first     | last         |
+---------+--------+-----------------------------------------------------+-----------+--------------+
|  101476 | 193216 | ["Open House", "Costume"]                           | Vere      | Latty        |
|  104186 | 991668 | ["RSO", "Costume"]                                  | Fionna    | Sherrill     |
|  106340 | 903360 | []                                                  | Maia      | Cull         |
|  106393 | 562507 | ["BYOB", "Greek", "Open House", "Costume"]          | Ulla      | Bruell       |
|  106449 | 266839 | ["RSO", "BYOB", "Greek", "Costume"]                 | Elspeth   | Lail         |
|  108306 | 573070 | ["BYOB", "Chill"]                                   | Avrit     | Jannel       |
|  109315 | 638739 | ["RSO", "BYOB", "Open House", "Greek"]              | Vernice   | Karyn        |
|  109562 | 435815 | ["Costume", "RSO", "BYOB"]                          | Katherina | Chemesh      |
|  109854 | 445700 | ["Costume", "Open House"]                           | Collen    | Middlesworth |
|  111028 | 718832 | ["Costume", "BYOB", "Greek", "Chill", "Open House"] | Jaymee    | Tasia        |
|  111322 | 730566 | ["Costume", "Greek", "Chill", "Open House", "RSO"]  | Catherina | Reld         |
|  111337 | 825648 | ["Chill", "RSO", "Greek", "BYOB", "Open House"]     | Mirna     | Haggerty     |
|  111587 | 268921 | ["Chill"]                                           | Agace     | Camala       |
|  111611 | 961297 | []                                                  | Romonda   | Fawcett      |
|  112825 | 976349 | ["RSO", "BYOB", "Greek", "Open House", "Chill"]     | Caritta   | Weld         |
+---------+--------+-----------------------------------------------------+-----------+--------------+
15 rows in set (0.01 sec)

mysql>
```

Indexing Analysis

For Query 1:

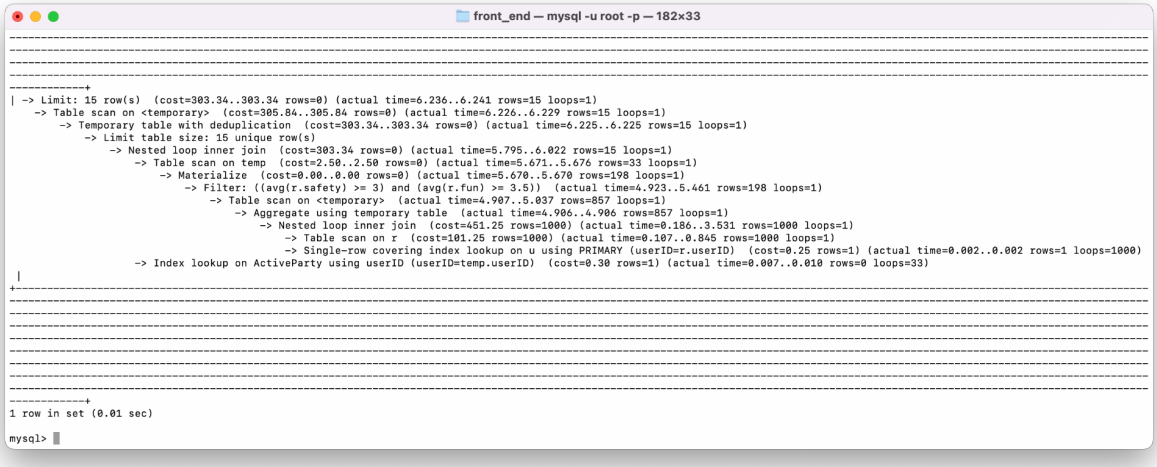
First index

CREATE INDEX rating_idx_1
ON Rating (ratingID);



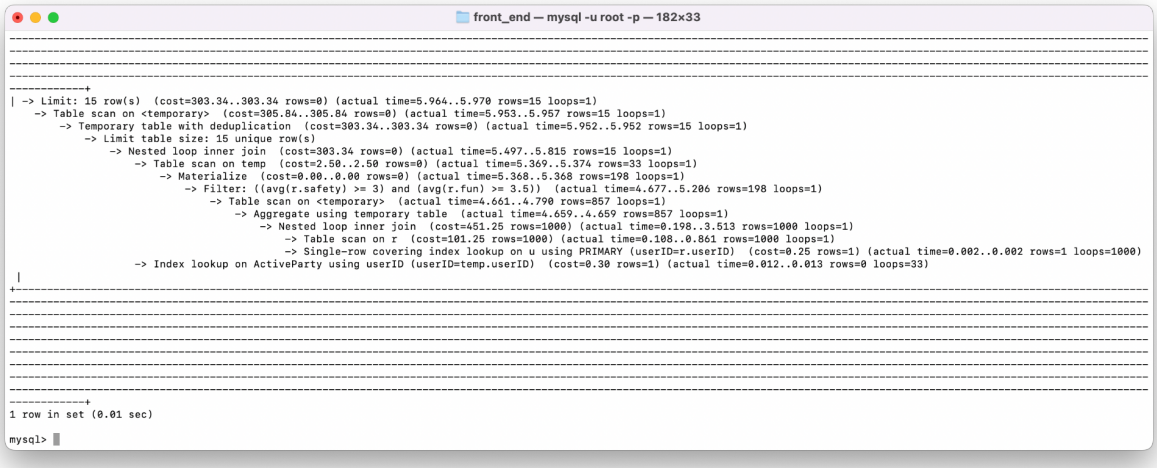
Second index
CREATE INDEX rating_idx_2
ON Rating (safety ASC, fun ASC);

Reasoning for change: Since we search against safety and fun in the query, it makes sense to create an index for these.



Third index
CREATE INDEX rating_idx_3
ON Rating (safety DESC, fun DESC);

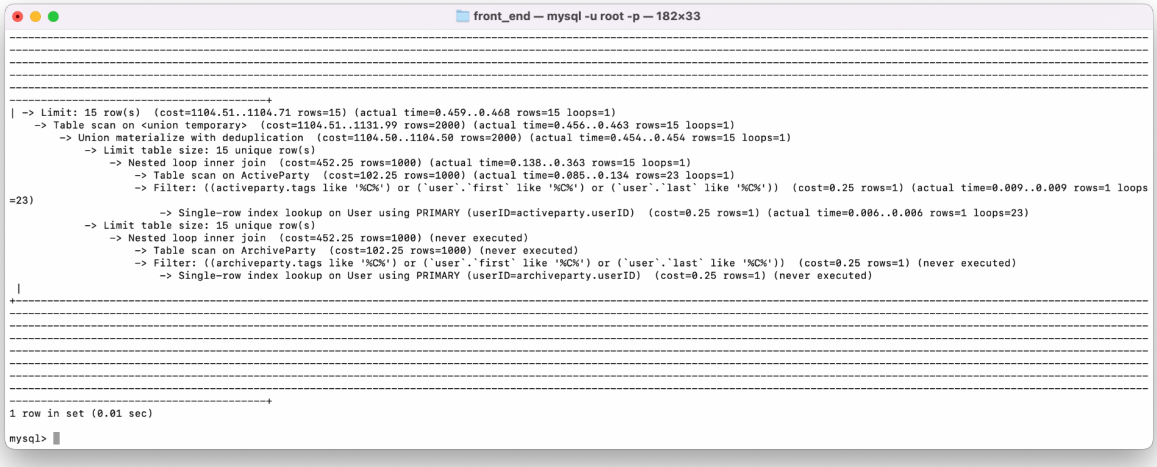
Reasoning for change: we are searching for users with only above a certain safety/fun threshold, so it makes sense to make them descending.



For Query 1, we decided to go with rating_idx_3. Although the performance difference was negligible, with more rows the minute difference in time might scale to something significant. We think that the difference was so small because there were so few rows and our query wasn't very complicated so SQL optimizes it anyways.

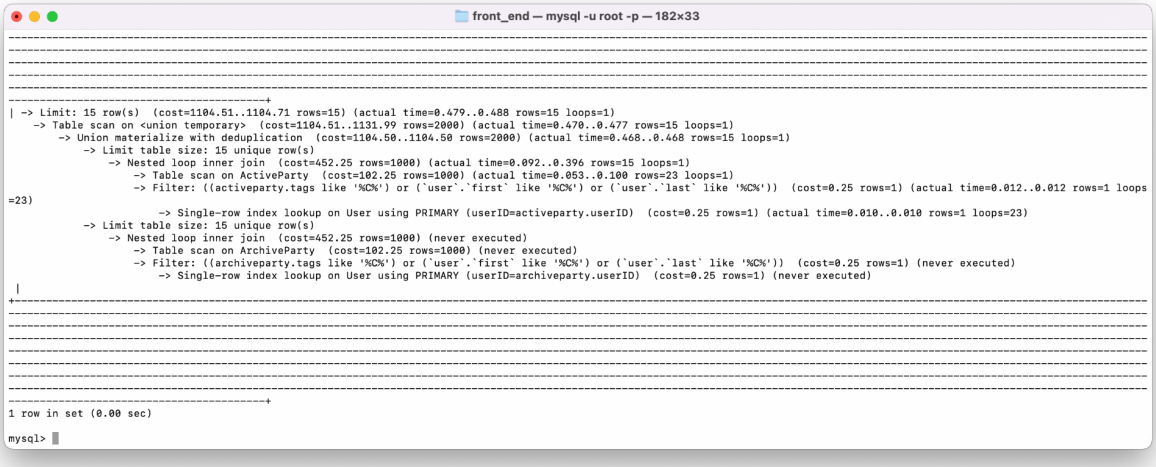
For Query 2:

First index
CREATE INDEX user_idx_1
ON User (userID);



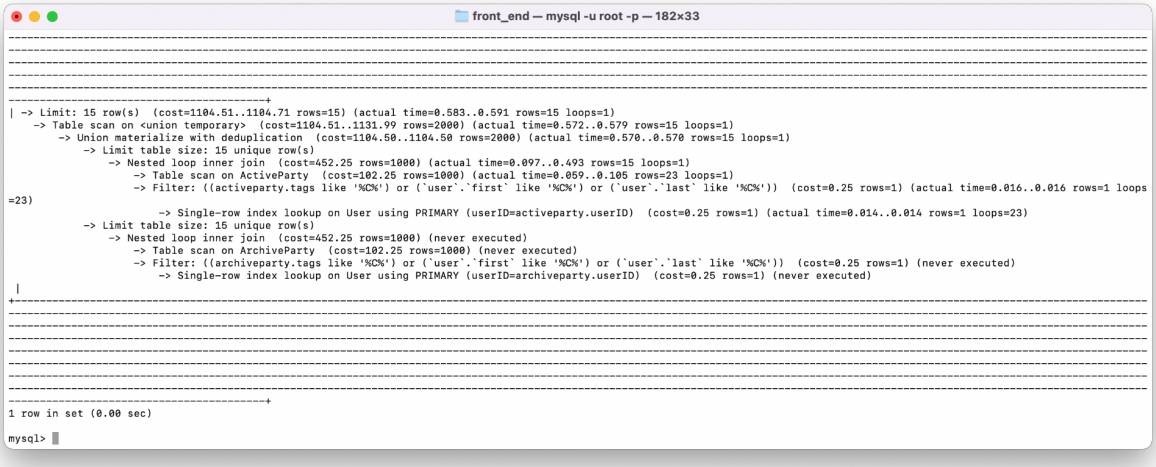
Second index
CREATE INDEX user_idx_2
ON User (first DESC, last DESC);

Reasoning for change: since we search against first and last names, it makes sense to create an index for these.



Third index
CREATE INDEX user_idx_3
ON User (first ASC, last ASC);

Reasoning for change: people are more likely to have names near the beginning of the alphabet, so it makes sense to create the order alphabetically.



For Query 2, we decided to go with user_idx_3. The performance of each index was quite similar, but we think that adding more rows would increase the difference significantly. The difference was probably so small because there were so few rows and our query wasn't very complicated so SQL optimizes it anyways.