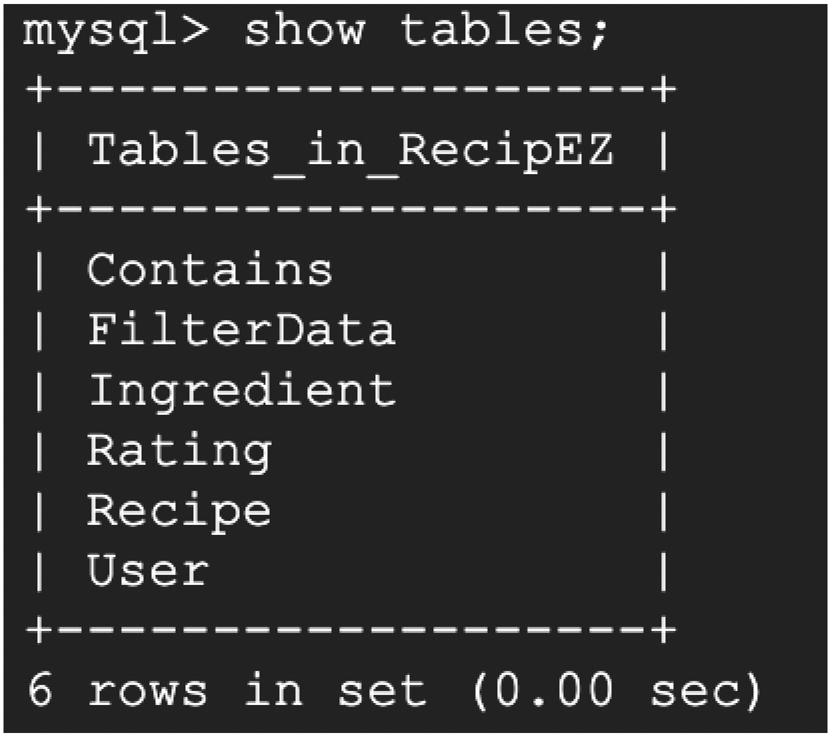
1. Implement at least 4 main tables



```
mysql> show tables;
+-------------------+
| Tables_in_RecipEZ |
+-------------------+
| Contains          |
| FilterData        |
| Ingredient        |
| Rating            |
| Recipe            |
| User              |
+-------------------+
6 rows in set (0.00 sec)
```

2. DDL commands that we used to create each tables

CREATE TABLE Ingredient(ingredientID INT, name VARCHAR(255), PRIMARY KEY(ingredientID));

CREATE TABLE Contains(recipeID INT, ingredientID INT, PRIMARY KEY(recipeID, ingredientID), FOREIGN KEY(recipeID) REFERENCES Recipe(recipeID), FOREIGN KEY(ingredientID) REFERENCES Ingredient(ingredientID));

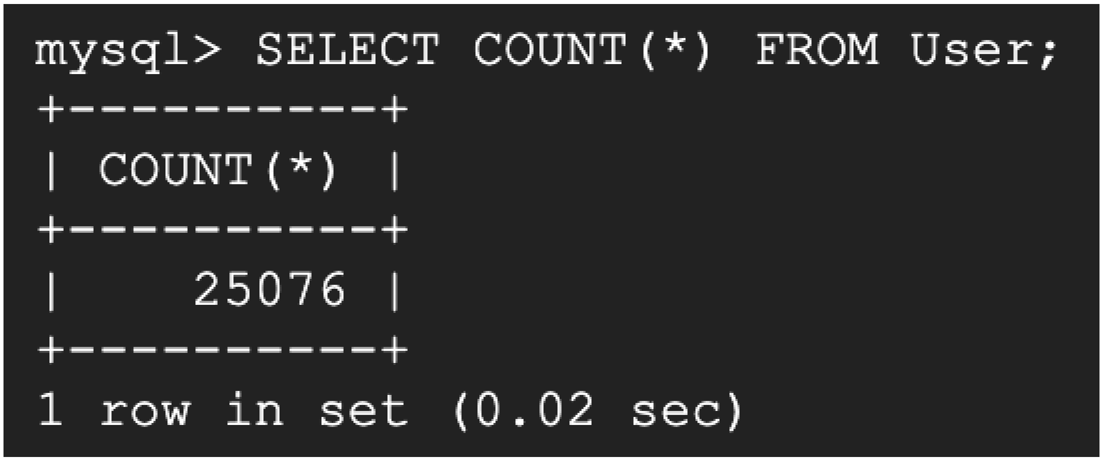
CREATE TABLE User (userID INT, name VARCHAR(255), PRIMARY KEY(userID));

CREATE TABLE Rating (ratingID INT, recipeID INT, score REAL, review VARCHAR(255), userID INT, PRIMARY KEY(ratingID), FOREIGN KEY(recipeID) REFERENCES Recipe(recipeID), FOREIGN KEY(userID) REFERENCES User(userID));

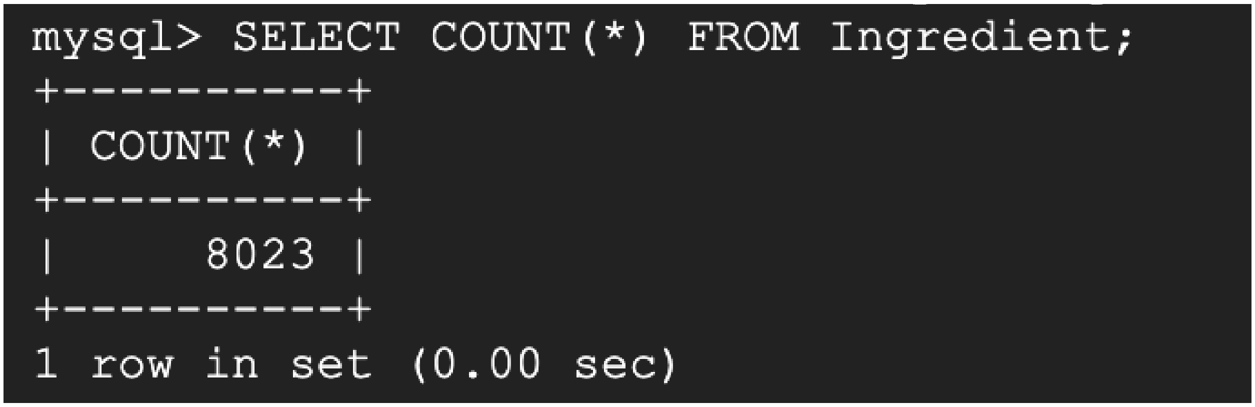
CREATE TABLE FilterData (filterdataID INT, recipeID INT, minutesToPrepare INT, numOfSteps INT, tags VARCHAR(255), calories INT, PRIMARY KEY(filterdataID), FOREIGN KEY(recipeID) REFERENCES Recipe(recipeID));

CREATE TABLE Recipe (recipeID INT, name VARCHAR(255), description VARCHAR(1024), steps VARCHAR(1024), PRIMARY KEY(recipeID));

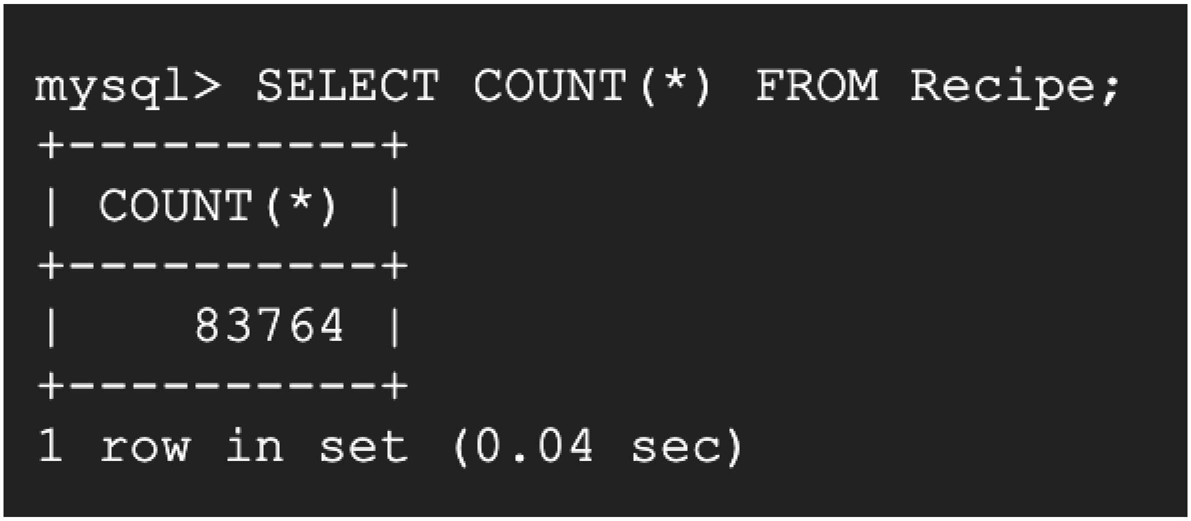
3. Insert data to tables. Insert at least 1000 rows in each table



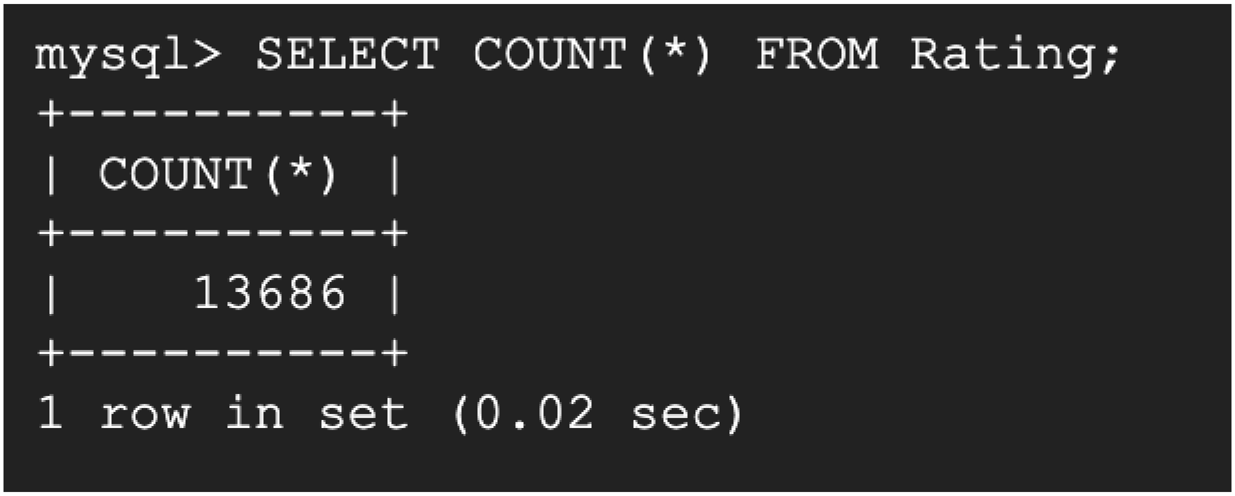
```
mysql> SELECT COUNT(*) FROM User;
+----------+
| COUNT(*) |
+----------+
|    25076 |
+----------+
1 row in set (0.02 sec)
```



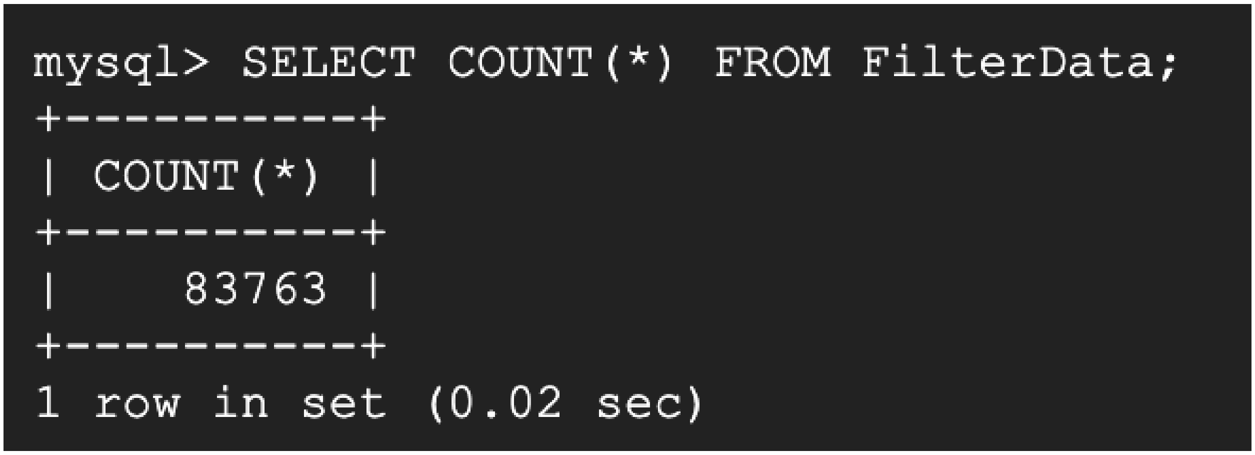
```
mysql> SELECT COUNT(*) FROM Ingredient;
+----------+
| COUNT(*) |
+----------+
|     8023 |
+----------+
1 row in set (0.00 sec)
```



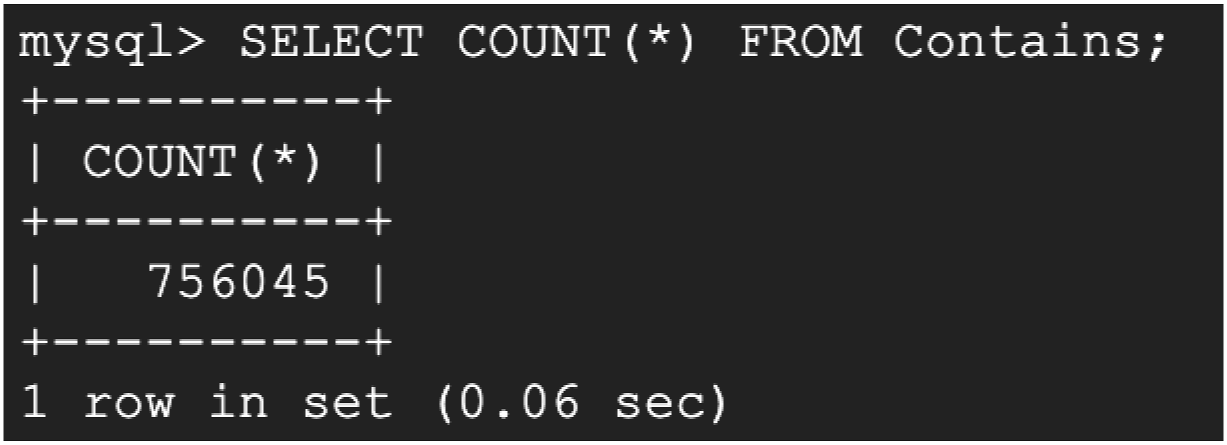
```
mysql> SELECT COUNT(*) FROM Recipe;
+----------+
| COUNT(*) |
+----------+
|    83764 |
+----------+
1 row in set (0.04 sec)
```



```
mysql> SELECT COUNT(*) FROM Rating;
+----------+
| COUNT(*) |
+----------+
|    13686 |
+----------+
1 row in set (0.02 sec)
```



```
mysql> SELECT COUNT(*) FROM FilterData;
+----------+
| COUNT(*) |
+----------+
|    83763 |
+----------+
1 row in set (0.02 sec)
```



```
mysql> SELECT COUNT(*) FROM Contains;
+----------+
| COUNT(*) |
+----------+
|   756045 |
+----------+
1 row in set (0.06 sec)
```

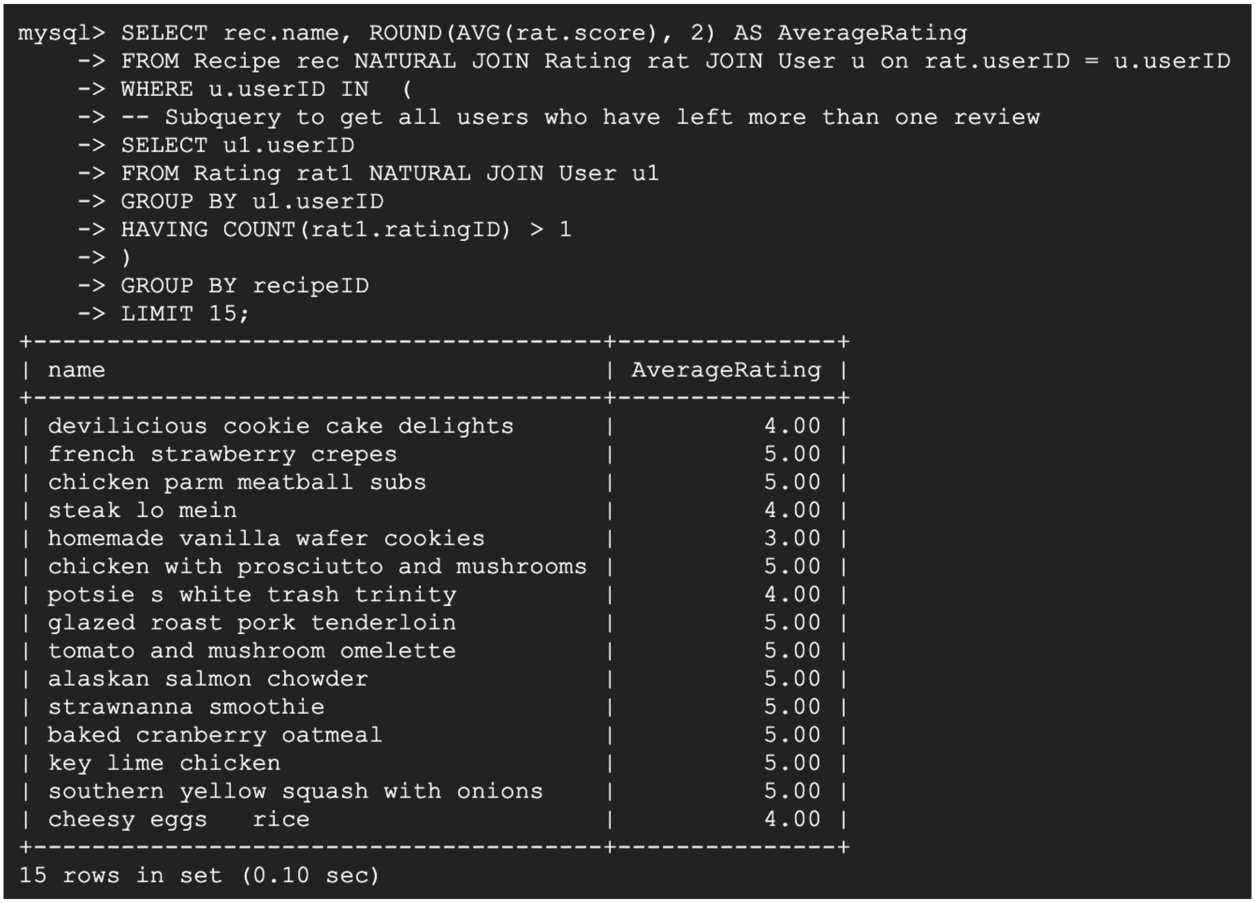
4. Two advanced SQL queries & Screenshot

-- Query to show all recipes and their average ratings by users that have given multiple ratings

SELECT rec.name, ROUND(AVG(rat.score), 2) AS AverageRating

```
FROM Recipe rec NATURAL JOIN Rating rat JOIN User u on rat.userID = u.userID
WHERE u.userID IN  (
        -- Subquery to get all users who have left more than one review
        SELECT u1.userID
        FROM Rating rat1 NATURAL JOIN User u1
        GROUP BY u1.userID
        HAVING COUNT(rat1.ratingID) > 1
)
GROUP BY recipeID
```

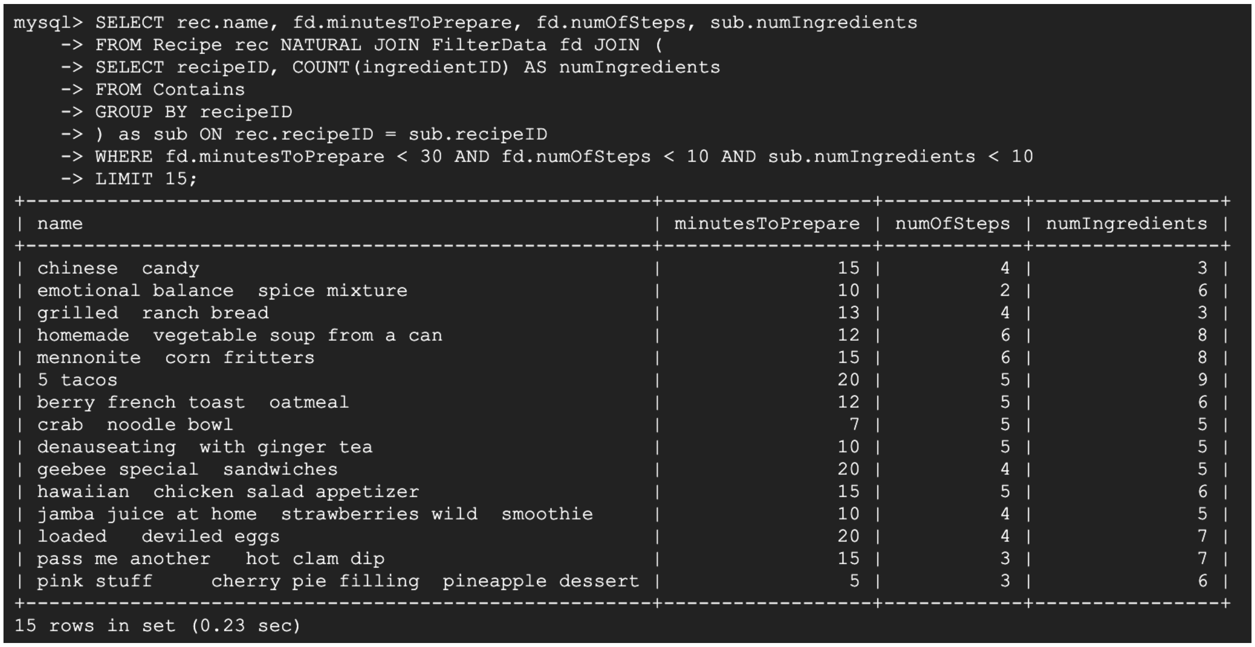
(A real query would ORDER BY AverageRate DESC but we left it unordered to show results that aren’t just 5, 5, 5…)



```
mysql> SELECT rec.name, ROUND(AVG(rat.score), 2) AS AverageRating
    -> FROM Recipe rec NATURAL JOIN Rating rat JOIN User u on rat.userID = u.userID
    -> WHERE u.userID IN  (
    -> -- Subquery to get all users who have left more than one review
    -> SELECT u1.userID
    -> FROM Rating rat1 NATURAL JOIN User u1
    -> GROUP BY u1.userID
    -> HAVING COUNT(rat1.ratingID) > 1
    -> )
    -> GROUP BY recipeID
    -> LIMIT 15;
+-------------------------------------+---------------+
| name                                | AverageRating |
+-------------------------------------+---------------+
| devilicious cookie cake delights    |          4.00 |
| french strawberry crepes            |          5.00 |
| chicken parm meatball subs          |          5.00 |
| steak lo mein                       |          4.00 |
| homemade vanilla wafer cookies      |          3.00 |
| chicken with prosciutto and mushrooms |        5.00 |
| potsie s white trash trinity        |          4.00 |
| glazed roast pork tenderloin        |          5.00 |
| tomato and mushroom omelette        |          5.00 |
| alaskan salmon chowder              |          5.00 |
| strawnanna smoothie                 |          5.00 |
| baked cranberry oatmeal             |          5.00 |
| key lime chicken                    |          5.00 |
| southern yellow squash with onions  |          5.00 |
| cheesy eggs   rice                  |          4.00 |
+-------------------------------------+---------------+
15 rows in set (0.10 sec)
```

```
-- Query to find easy recipes (easiness defined by a shorter amount of time and fewer steps/ingredients)
SELECT rec.name, fd.minutesToPrepare, fd.numOfSteps, sub.numIngredients
```

```
FROM Recipe rec NATURAL JOIN FilterData fd JOIN (
        SELECT recipeID, COUNT(ingredientID) AS numIngredients
        FROM Contains
        GROUP BY recipeID
) as sub ON rec.recipeID = sub.recipeID
WHERE fd.minutesToPrepare < 30 AND fd.numOfSteps < 10 AND sub.numIngredients < 10
```

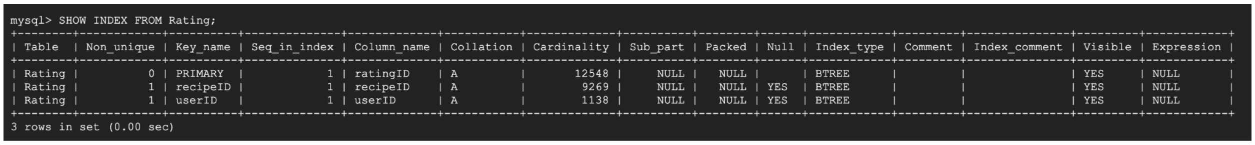


```
mysql> SELECT rec.name, fd.minutesToPrepare, fd.numOfSteps, sub.numIngredients
    -> FROM Recipe rec NATURAL JOIN FilterData fd JOIN (
    -> SELECT recipeID, COUNT(ingredientID) AS numIngredients
    -> FROM Contains
    -> GROUP BY recipeID
    -> ) as sub ON rec.recipeID = sub.recipeID
    -> WHERE fd.minutesToPrepare < 30 AND fd.numOfSteps < 10 AND sub.numIngredients < 10
    -> LIMIT 15;
+------------------------------------------------------+------------------+------------+----------------+
| name                                                 | minutesToPrepare | numOfSteps | numIngredients |
+------------------------------------------------------+------------------+------------+----------------+
| chinese  candy                                       |               15 |          4 |              3 |
| emotional balance  spice mixture                     |               10 |          2 |              6 |
| grilled  ranch bread                                 |               13 |          4 |              3 |
| homemade  vegetable soup from a can                  |               12 |          6 |              8 |
| mennonite  corn fritters                             |               15 |          6 |              8 |
| 5 tacos                                              |               20 |          5 |              9 |
| berry french toast  oatmeal                          |               12 |          5 |              6 |
| crab  noodle bowl                                    |                7 |          5 |              5 |
| denauseating  with ginger tea                        |               10 |          5 |              5 |
| geebee special  sandwiches                           |               20 |          4 |              5 |
| hawaiian  chicken salad appetizer                    |               15 |          5 |              6 |
| jamba juice at home  strawberries wild  smoothie     |               10 |          4 |              5 |
| loaded   deviled eggs                                |               20 |          4 |              7 |
| pass me another   hot clam dip                       |               15 |          3 |              7 |
| pink stuff     cherry pie filling  pineapple dessert |                5 |          3 |              6 |
+------------------------------------------------------+------------------+------------+----------------+
15 rows in set (0.23 sec)
```

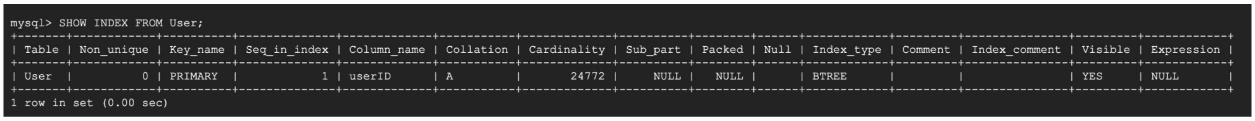
5. Indexing Analysis

a) For first query

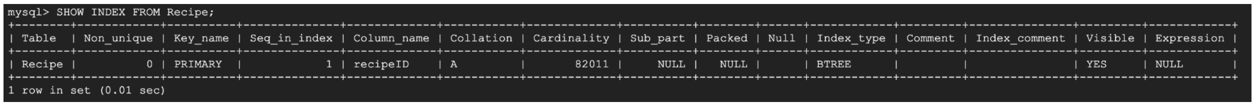
Before adding index



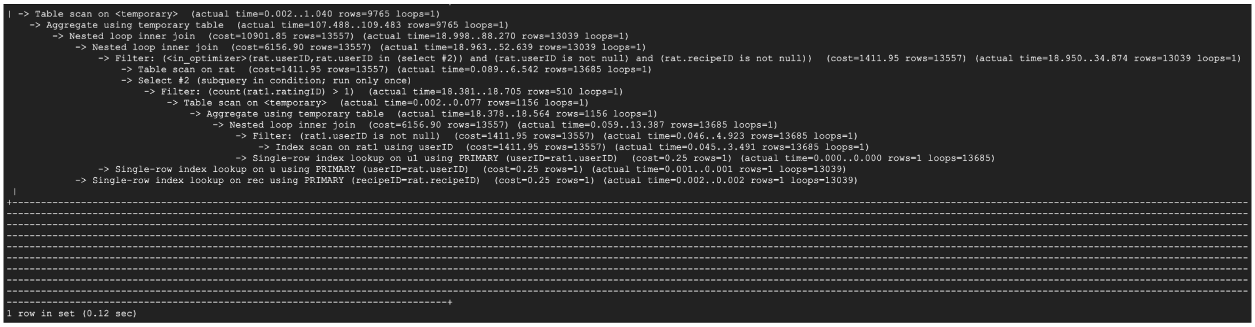
```
mysql> SHOW INDEX FROM Rating;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Rating |          0 | PRIMARY  |            1 | ratingID    | A         |       12548 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | recipeID |            1 | recipeID    | A         |        9269 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | userID   |            1 | userID      | A         |        1138 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.00 sec)
```



```
mysql> SHOW INDEX FROM User;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| User  |          0 | PRIMARY  |            1 | userID      | A         |       24772 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.00 sec)
```



```
mysql> SHOW INDEX FROM Recipe;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Recipe |          0 | PRIMARY  |            1 | recipeID    | A         |       82011 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.01 sec)
```



```
| -> Table scan on <temporary>  (actual time=0.002..1.040 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=107.488..109.483 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=18.998..88.270 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=18.963..52.639 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=18.950..34.874 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.089..6.542 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=18.381..18.705 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.002..0.077 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=18.378..18.564 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.059..13.387 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.046..4.923 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.045..3.491 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
1 row in set (0.12 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.016..57.581 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=12.998..34.776 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.052..4.237 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.025..2.629 rows=13685 loops=1)

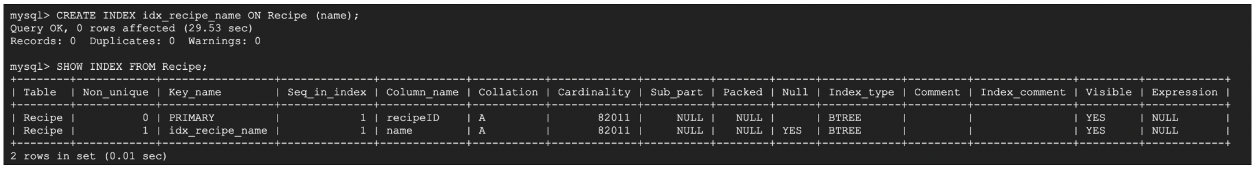
Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.25 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

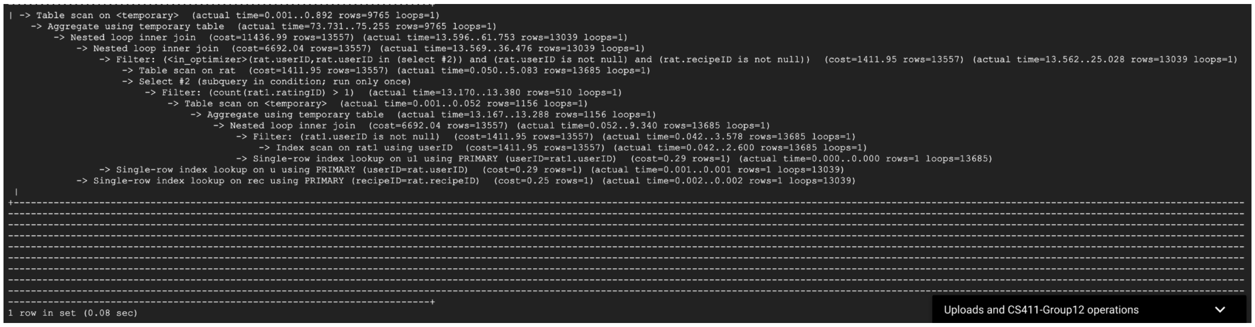
1 row in set **(0.12 sec)**

After adding index called idx_recipe_name (After drop previous index)



```
mysql> CREATE INDEX idx_recipe_name ON Recipe (name);
Query OK, 0 rows affected (29.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Recipe;
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name        | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Recipe |          0 | PRIMARY         |            1 | recipeID    | A         |       82011 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Recipe |          1 | idx_recipe_name |            1 | name        | A         |       82011 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.01 sec)
```



```
| -> Table scan on <temporary>  (actual time=0.001..0.892 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=73.731..75.255 rows=9765 loops=1)
        -> Nested loop inner join  (cost=11436.99 rows=13557) (actual time=13.596..61.753 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6692.04 rows=13557) (actual time=13.569..36.476 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.562..25.028 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.050..5.083 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=13.170..13.380 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.001..0.052 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=13.167..13.288 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6692.04 rows=13557) (actual time=0.052..9.340 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.042..3.578 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.042..2.600 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.29 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.29 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
1 row in set (0.08 sec)
```

Nested loop inner join **(cost=11436.99 rows=13557)** (actual time=13.596..61.753 rows=13039 loops=1)

Nested loop inner join **(cost=6692.04 rows=13557)** (actual time=13.569..36.476 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.050..5.083 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.042..2.600 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.29 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

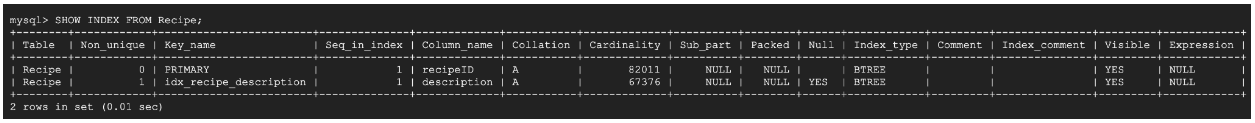
Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.29 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039

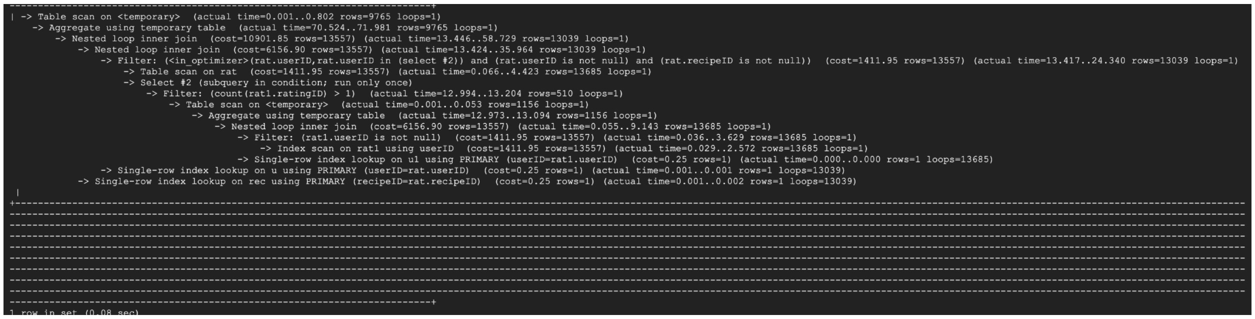
1 row in set **(0.08 sec)**

-> Our group did not choose the **name** of the **Recipe** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings worse effects to our query. As you can see, every cost and number of searched rows are increased after adding **idx_recipe_name** index except time in the last row. Since the **name** of the **Recipe** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **Recipe name**, we think this brings a worse effect. Also, none of GROUP BY clauses use the **Recipe name**, we think that it didn't improve query performance.

After adding index called idx_recipe_description (After drop previous index)



```
mysql> SHOW INDEX FROM Recipe;
+--------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name               | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Recipe |          0 | PRIMARY                |            1 | recipeID    | A         |       82011 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Recipe |          1 | idx_recipe_description |            1 | description | A         |       67376 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.01 sec)
```



```
-------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.802 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=70.524..71.981 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=13.446..58.729 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=13.424..35.964 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.417..24.340 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.066..4.423 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=12.994..13.204 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.001..0.053 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=12.973..13.094 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.055..9.143 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.036..3.629 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.029..2.572 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=13039)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
1 row in set (0.08 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.446..58.729 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=13.424..35.964 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.066..4.423 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.029..2.572 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

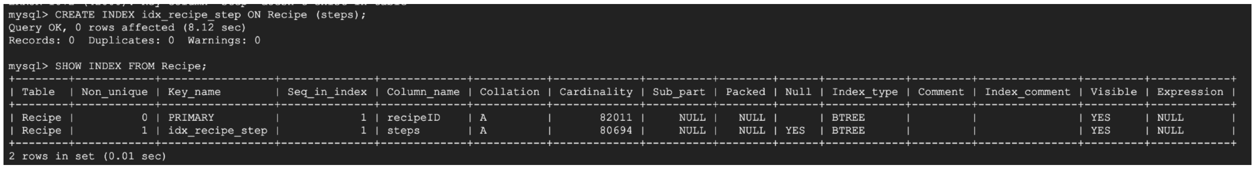
Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.25 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.001..0.002 rows=1 loops=13039)

1 row in set **(0.08 sec)**

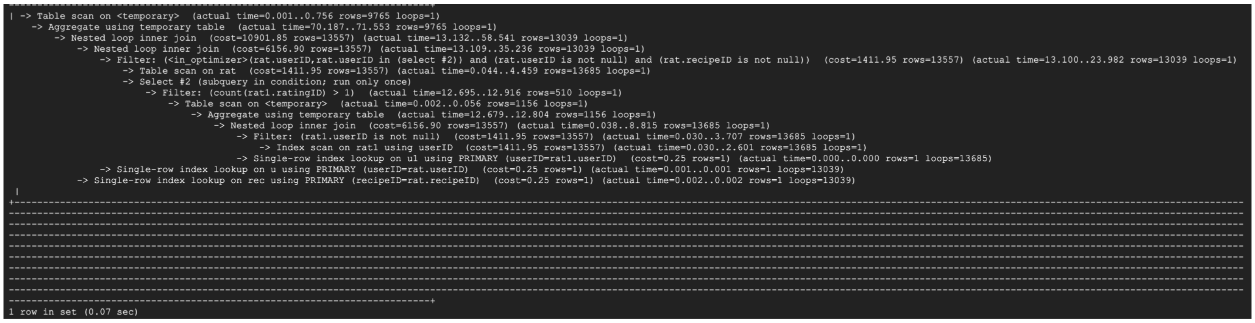
-> Our group did not choose the **description** of the **Recipe** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings no effects to our query. As you can see, every cost and number of searched rows remain the same after adding **idx_recipe_description** index except time in the last row. Since the **description** of the **Recipe** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **Recipe description**, we think adding this as index did not bring any better results. Also, none of **Aggregation via GROUP BY** clauses use the **Recipe description**, we think that it didn't improve query performance.

After adding index called idx_recipe_steps (After drop previous index)



```
mysql> CREATE INDEX idx_recipe_step ON Recipe (steps);
Query OK, 0 rows affected (8.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Recipe;
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name        | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Recipe |          0 | PRIMARY         |            1 | recipeID    | A         |       82011 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Recipe |          1 | idx_recipe_step |            1 | steps       | A         |       80694 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.01 sec)
```



```
| -> Table scan on <temporary>  (actual time=0.001..0.756 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=70.187..71.553 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=13.132..58.541 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=13.109..35.236 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.100..23.982 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.044..4.459 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=12.695..12.916 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.002..0.056 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=12.679..12.804 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.038..8.815 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.030..3.707 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.030..2.601 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------+
1 row in set (0.07 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.132..58.541 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=13.109..35.236 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.044..4.459 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.030..2.601 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

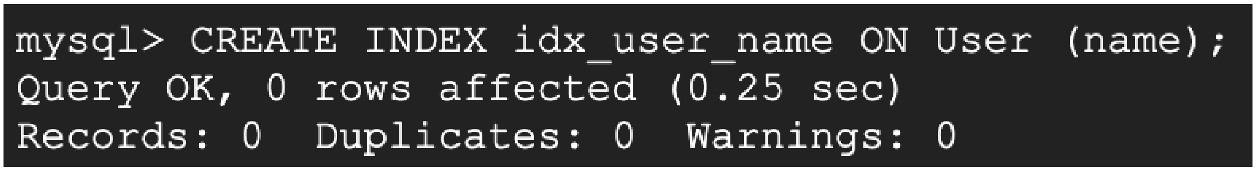
Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.25 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

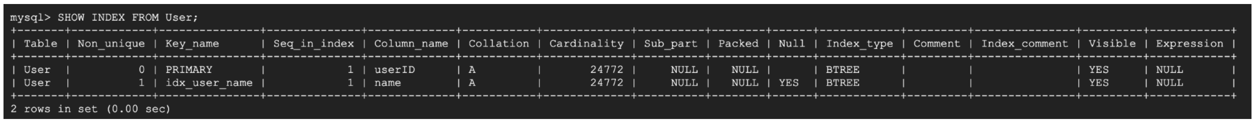
1 row in set **(0.07 sec)**

-> Our group did not choose the **steps** of the **Recipe** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings no effects to our query. As you can see, every cost and number of searched rows remain the same after adding **idx_recipe_steps** index except time in the last row. Since the **steps** of the **Recipe** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **Recipe steps**, we think adding this as index did not bring any better results. Also, none of **Aggregation via GROUP BY** clauses use the **Recipe steps**, we think that it didn't improve query performance.

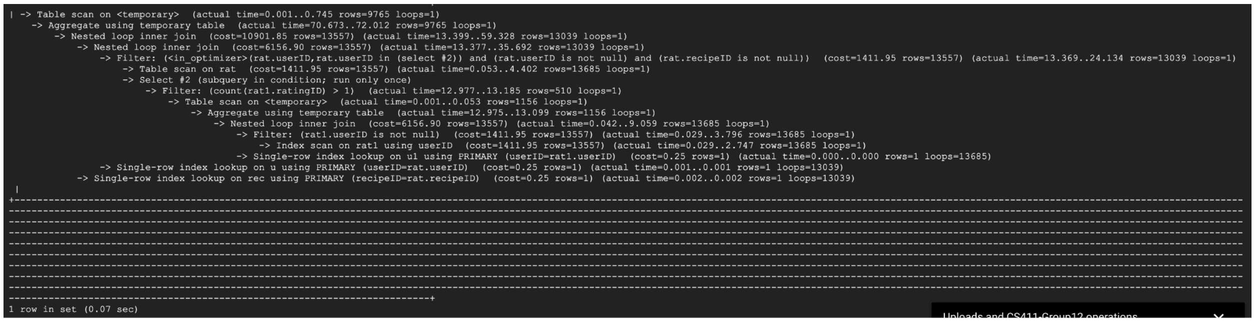
After adding index called idx_user_name (After drop previous index)



```
mysql> CREATE INDEX idx_user_name ON User (name);
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
mysql> SHOW INDEX FROM User;
+-------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| User  |          0 | PRIMARY       |            1 | userID      | A         |       24772 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| User  |          1 | idx_user_name |            1 | name        | A         |       24772 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+-------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
2 rows in set (0.00 sec)
```



```
| -> Table scan on <temporary>  (actual time=0.001..0.745 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=70.673..72.012 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=13.399..59.328 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=13.377..35.692 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.369..24.134 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.053..4.402 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=12.977..13.185 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.001..0.053 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=12.975..13.099 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.042..9.059 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.029..3.796 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.029..2.747 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+--------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------+
1 row in set (0.07 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.399..59.328 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=13.377..35.692 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.053..4.402 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.029..2.747 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

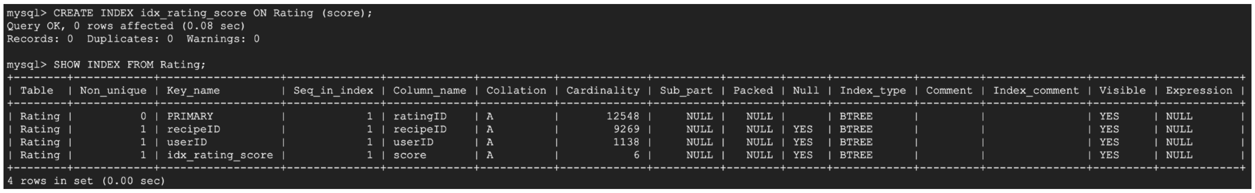
Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.25 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

1 row in set **(0.07 sec)**

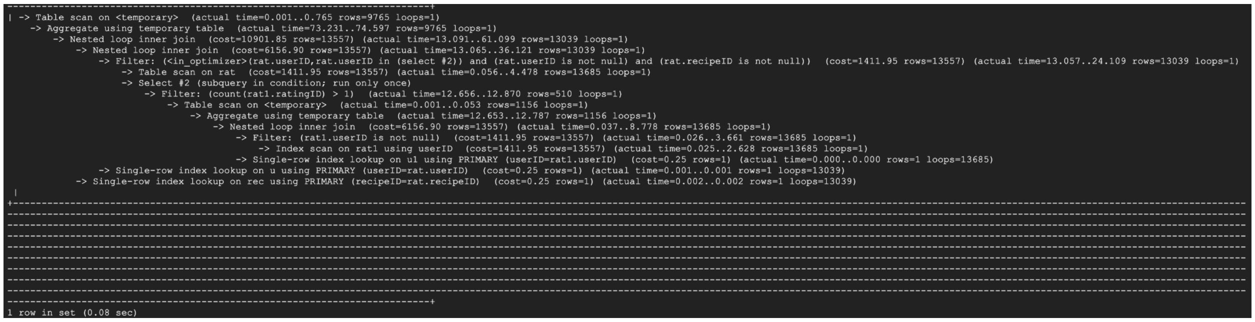
-> Our group did not choose the **name** of the **User** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings no effects to our query. As you can see, every cost and number of searched rows remain the same after adding **idx_user_name** index except time in the last row. Since the **name** of the **User** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **User name**, we think adding this as index did not bring any better results. Also, none of **Aggregation via GROUP BY** clauses use the **User name**, we think that it didn't improve query performance.

After adding index called idx_rating_score (After drop previous index)



```
mysql> CREATE INDEX idx_rating_score ON Rating (score);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Rating;
+--------+------------+------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name         | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Rating |          0 | PRIMARY          |            1 | ratingID    | A         |       12548 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | recipeID         |            1 | recipeID    | A         |        9269 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | userID           |            1 | userID      | A         |        1138 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | idx_rating_score |            1 | score       | A         |           6 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.00 sec)
```



```
------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.765 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=73.231..74.597 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=13.091..61.099 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=13.065..36.121 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.057..24.109 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.056..4.478 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=12.656..12.870 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.001..0.053 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=12.653..12.787 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.037..8.778 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.026..3.661 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.025..2.628 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+--------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------+
1 row in set (0.08 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.091..61.099 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=13.065..36.121 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.056..4.478 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.025..2.628 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

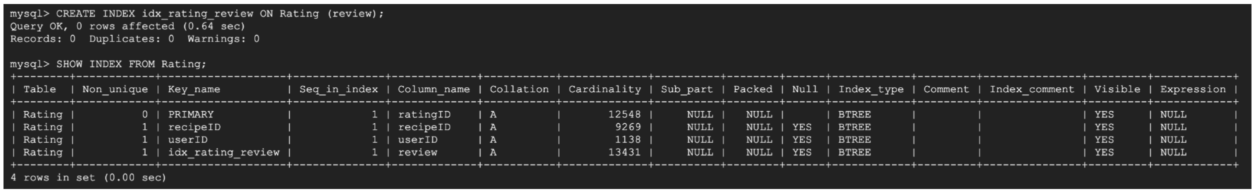
Single-row index lookup on u using PRIMARY (userID=rat.userID) **(cost=0.25 rows=1)** (actual time=0.001..0.001 rows=1 loops=13039)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

1 row in set **(0.08 sec)**

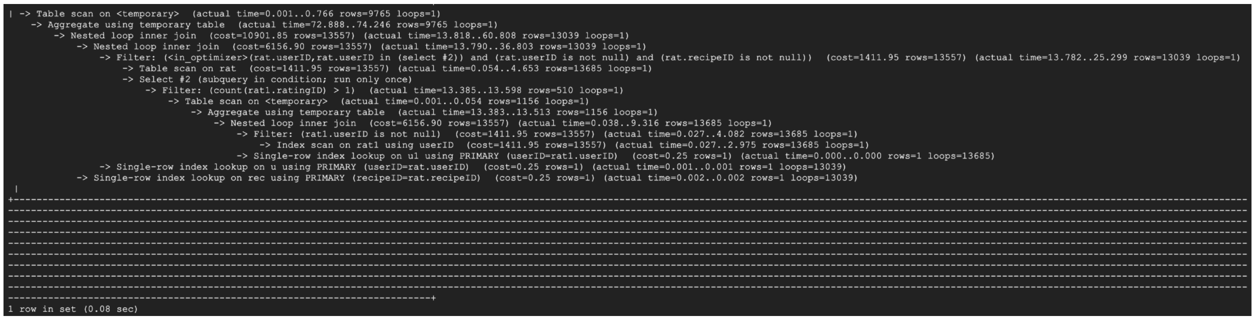
-> Our group did not choose the **score** of the **Rating** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings no effects to our query. As you can see, every cost and number of searched rows remain the same after adding **idx_rating_score** index except time in the last row. Since the **score** of the **Rating** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **Rating score**, we think adding this as index did not bring any better results. Also, none of **Aggregation via GROUP BY** clauses use the **Rating score**, we think that it didn't improve query performance.

After adding index called idx_rating_review (After drop previous index)



```
mysql> CREATE INDEX idx_rating_review ON Rating (review);
Query OK, 0 rows affected (0.64 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Rating;
+--------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name          | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Rating |          0 | PRIMARY           |            1 | ratingID    | A         |       12548 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | recipeID          |            1 | recipeID    | A         |        9269 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | userID            |            1 | userID      | A         |        1138 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| Rating |          1 | idx_rating_review |            1 | review      | A         |       13431 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+--------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.00 sec)
```



```
| -> Table scan on <temporary>  (actual time=0.001..0.766 rows=9765 loops=1)
    -> Aggregate using temporary table  (actual time=72.888..74.246 rows=9765 loops=1)
        -> Nested loop inner join  (cost=10901.85 rows=13557) (actual time=13.818..60.808 rows=13039 loops=1)
            -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=13.790..36.803 rows=13039 loops=1)
                -> Filter: (<in_optimizer>(rat.userID,rat.userID in (select #2)) and (rat.userID is not null) and (rat.recipeID is not null))  (cost=1411.95 rows=13557) (actual time=13.782..25.299 rows=13039 loops=1)
                    -> Table scan on rat  (cost=1411.95 rows=13557) (actual time=0.054..4.653 rows=13685 loops=1)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: (count(rat1.ratingID) > 1)  (actual time=13.385..13.598 rows=510 loops=1)
                            -> Table scan on <temporary>  (actual time=0.001..0.054 rows=1156 loops=1)
                                -> Aggregate using temporary table  (actual time=13.383..13.513 rows=1156 loops=1)
                                    -> Nested loop inner join  (cost=6156.90 rows=13557) (actual time=0.038..9.316 rows=13685 loops=1)
                                        -> Filter: (rat1.userID is not null)  (cost=1411.95 rows=13557) (actual time=0.027..4.082 rows=13685 loops=1)
                                            -> Index scan on rat1 using userID  (cost=1411.95 rows=13557) (actual time=0.027..2.975 rows=13685 loops=1)
                                        -> Single-row index lookup on u1 using PRIMARY (userID=rat1.userID)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=13685)
                -> Single-row index lookup on u using PRIMARY (userID=rat.userID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=13039)
            -> Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=13039)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------+
1 row in set (0.08 sec)
```

Nested loop inner join **(cost=10901.85 rows=13557)** (actual time=13.818..60.808 rows=13039 loops=1)

Nested loop inner join **(cost=6156.90 rows=13557)** (actual time=13.790..36.803 rows=13039 loops=1)

Table scan on rat **(cost=1411.95 rows=13557)** (actual time=0.054..4.653 rows=13685 loops=1)

Index scan on rat1 using userID **(cost=1411.95 rows=13557)** (actual time=0.027..2.975 rows=13685 loops=1)

Single-row index lookup on u1 using PRIMARY (userID=rat1.userID) **(cost=0.25 rows=1)** (actual time=0.000..0.000 rows=1 loops=13685)

Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

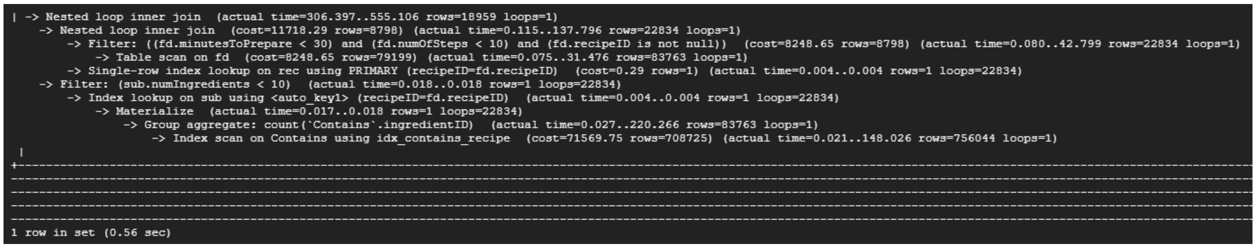
Single-row index lookup on rec using PRIMARY (recipeID=rat.recipeID) **(cost=0.25 rows=1)** (actual time=0.002..0.002 rows=1 loops=13039)

1 row in set **(0.08 sec)**

-> Our group did not choose the **review** of the **Rating** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings no effects to our query. As you can see, every cost and number of searched rows remain the same after adding **idx_rating_review** index except time in the last row. Since the **review** of the **Rating** table is not the primary key of any tables we used in advanced query and recipe is not uniquely identified with **Rating review**, we think adding this as index did not bring any better results. Also, none of **Aggregation via GROUP BY** clauses use the **Rating review**, we think that it didn't improve query performance.

b) For second query

Before adding index



```
| -> Nested loop inner join  (actual time=306.397..555.106 rows=18959 loops=1)
    -> Nested loop inner join  (cost=11718.29 rows=8798) (actual time=0.115..137.796 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=8798) (actual time=0.080..42.799 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.075..31.476 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.29 rows=1) (actual time=0.004..0.004 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.018..0.018 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.004..0.004 rows=1 loops=22834)
            -> Materialize  (actual time=0.017..0.018 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.027..220.266 rows=83763 loops=1)
                    -> Index scan on Contains using idx_contains_recipe  (cost=71569.75 rows=708725) (actual time=0.021..148.026 rows=756044 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.56 sec)
```

Nested loop inner join (actual time=306.397..555.106 rows=18959 loops=1)

Nested loop inner join (cost=11718.29 rows=8798) (actual time=0.115..137.796 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=8798) (actual time=0.080..42.799 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.075..31.476 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.29 rows=1) (actual time=0.004..0.004 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.018..0.018 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.004..0.004 rows=1 loops=22834)

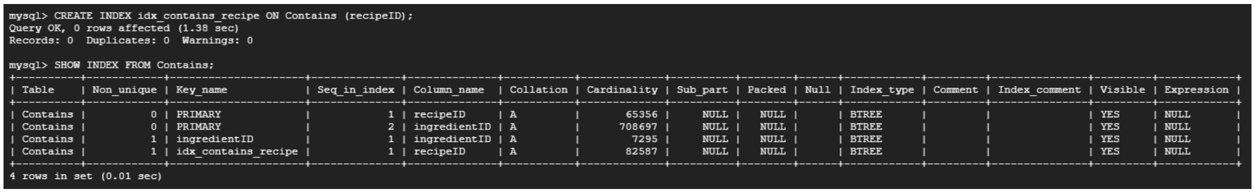
Materialize (actual time=0.017..0.018 rows=1 loops=22834)

Group aggregate: count(`Contains`.ingredientID) (actual time=0.027..220.266 rows=83763 loops=1)

Index scan on Contains using idx_contains_recipe (cost=71569.75 rows=708725) (actual time=0.021..148.026 rows=756044 loops=1)

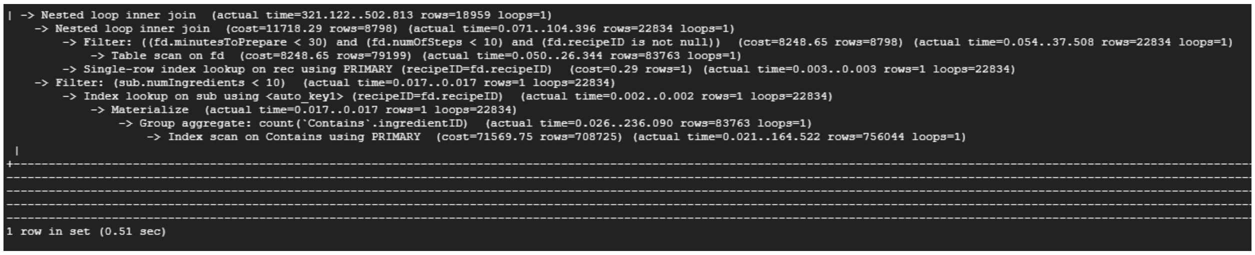
1 row in set (0.56 sec)

After adding index called idx_contains_recipe



```
mysql> CREATE INDEX idx_contains_recipe ON Contains (recipeID);
Query OK, 0 rows affected (1.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Contains;
+----------+------------+---------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name            | Seq_in_index | Column_name  | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+---------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Contains |          0 | PRIMARY             |            1 | recipeID     | A         |       65356 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          0 | PRIMARY             |            2 | ingredientID | A         |      708697 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          1 | ingredientID        |            1 | ingredientID | A         |        7295 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          1 | idx_contains_recipe |            1 | recipeID     | A         |       82587 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+---------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.01 sec)
```



```
| -> Nested loop inner join  (actual time=321.122..502.813 rows=18959 loops=1)
    -> Nested loop inner join  (cost=11718.29 rows=8798) (actual time=0.071..104.396 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=8798) (actual time=0.054..37.508 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.050..26.344 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.017..0.017 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.002..0.002 rows=1 loops=22834)
            -> Materialize  (actual time=0.017..0.017 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.026..236.090 rows=83763 loops=1)
                    -> Index scan on Contains using PRIMARY  (cost=71569.75 rows=708725) (actual time=0.021..164.522 rows=756044 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.51 sec)
```

Nested loop inner join (actual time=321.122..502.813 rows=18959 loops=1)

Nested loop inner join (cost=11718.29 rows=8798) (actual time=0.071..104.396 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=8798) (actual time=0.054..37.508 rows=22834 loops=1)

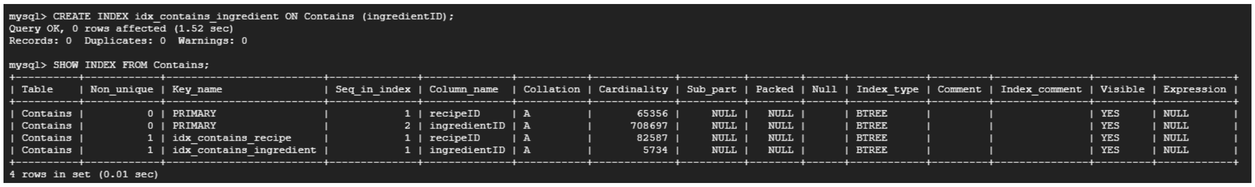
Table scan on fd (cost=8248.65 rows=79199) (actual time=0.050..26.344 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.017..0.017 rows=1 loops=22834)

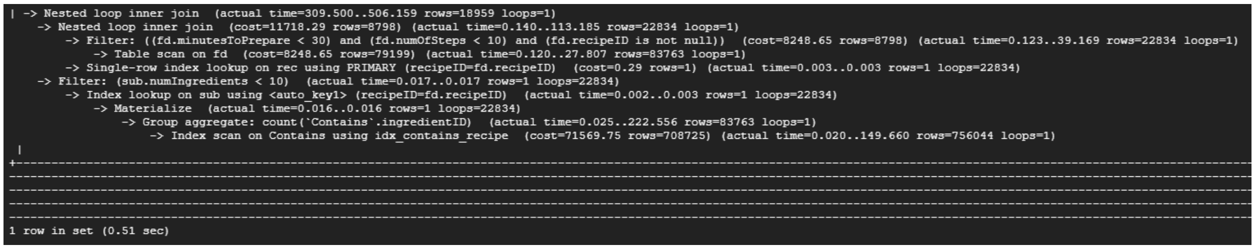
1 row in set (0.51 sec)

After adding index called idx_contains_ingredient



```
mysql> CREATE INDEX idx_contains_ingredient ON Contains (ingredientID);
Query OK, 0 rows affected (1.52 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM Contains;
+----------+------------+-------------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name                | Seq_in_index | Column_name  | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+-------------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Contains |          0 | PRIMARY                 |            1 | recipeID     | A         |       65356 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          0 | PRIMARY                 |            2 | ingredientID | A         |      708697 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          1 | idx_contains_recipe     |            1 | recipeID     | A         |       82587 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| Contains |          1 | idx_contains_ingredient |            1 | ingredientID | A         |        5734 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+-------------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.01 sec)
```



```
| -> Nested loop inner join  (actual time=309.500..506.159 rows=18959 loops=1)
    -> Nested loop inner join  (cost=11718.29 rows=8798) (actual time=0.140..113.185 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=8798) (actual time=0.123..39.169 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.120..27.807 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.017..0.017 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.002..0.003 rows=1 loops=22834)
            -> Materialize  (actual time=0.016..0.016 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.025..222.556 rows=83763 loops=1)
                    -> Index scan on Contains using idx_contains_recipe  (cost=71569.75 rows=708725) (actual time=0.020..149.660 rows=756044 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.51 sec)
```

Nested loop inner join (actual time=309.500..506.159 rows=18959 loops=1)

Nested loop inner join (cost=11718.29 rows=8798) (actual time=0.140..113.185 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=8798) (actual time=0.123..39.169 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.120..27.807 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.017..0.017 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.002..0.003 rows=1 loops=22834)

Materialize (actual time=0.016..0.016 rows=1 loops=22834)

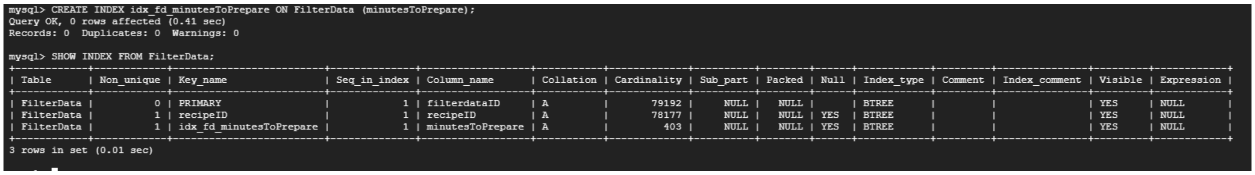
Group aggregate: count(`Contains`.ingredientID) (actual time=0.025..222.556 rows=83763 loops=1)

Index scan on Contains using idx_contains_recipe (cost=71569.75 rows=708725) (actual time=0.020..149.660 rows=756044 loops=1)

1 row in set (0.51 sec)

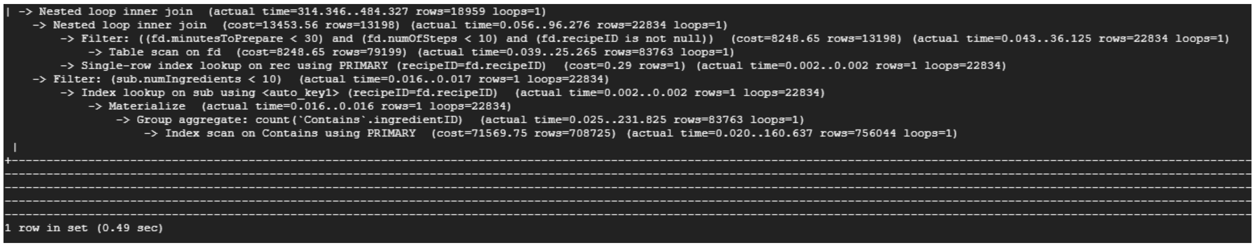
-> Our group did not choose the **recipeID or ingredientID** of the **Contains** table as an index. After comparing cost and number of rows searched using **EXPLAIN ANALYZE** command, we found out that query performance brings little performance benefit. We believe the reason for this is because the two foreign keys in the Contains table which connect recipes to their ingredients are already being used as a primary key for the table as a whole. As such, they are essentially already being indexed.

After adding index called idx_fd_minutesToPrepare



```
mysql> CREATE INDEX idx_fd_minutesToPrepare ON FilterData (minutesToPrepare);
Query OK, 0 rows affected (0.41 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM FilterData;
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name                | Seq_in_index | Column_name      | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| FilterData |          0 | PRIMARY                 |            1 | filterdataID     | A         |       79192 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | recipeID                |            1 | recipeID         | A         |       78177 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | idx_fd_minutesToPrepare |            1 | minutesToPrepare | A         |         403 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.01 sec)
```



```
| -> Nested loop inner join  (actual time=314.346..484.327 rows=18959 loops=1)
    -> Nested loop inner join  (cost=13453.56 rows=13198) (actual time=0.056..96.276 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=13198) (actual time=0.043..36.125 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.039..25.265 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.29 rows=1) (actual time=0.002..0.002 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.016..0.017 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.002..0.002 rows=1 loops=22834)
            -> Materialize  (actual time=0.016..0.016 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.025..231.825 rows=83763 loops=1)
                    -> Index scan on Contains using PRIMARY  (cost=71569.75 rows=708725) (actual time=0.020..160.637 rows=756044 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.49 sec)
```

Nested loop inner join (actual time=314.346..484.327 rows=18959 loops=1)

Nested loop inner join (cost=13453.56 rows=13198) (actual time=0.056..96.276 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=13198) (actual time=0.043..36.125 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.039..25.265 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.29 rows=1) (actual time=0.002..0.002 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.016..0.017 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.002..0.002 rows=1 loops=22834)

Materialize (actual time=0.016..0.016 rows=1 loops=22834)

Group aggregate: count(`Contains`.ingredientID) (actual time=0.025..231.825 rows=83763 loops=1)

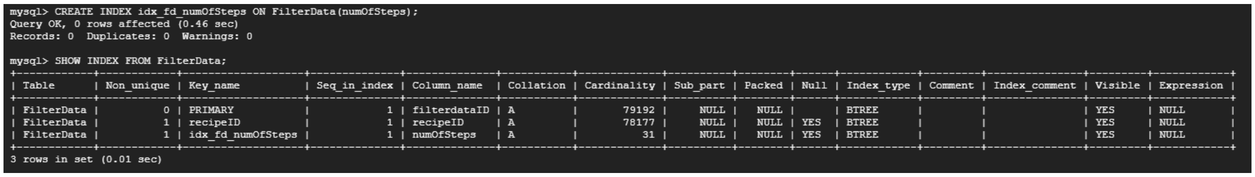
Index scan on Contains using PRIMARY (cost=71569.75 rows=708725) (actual time=0.020..160.637 rows=756044 loops=1)

1 row in set (0.49 sec)

-> Our group chose the **minutesToPrepare** of the **FilterData** table as an index. This is because of all the potential indexes we tried, this one had the best performance. This could possibly because the very first filtering step we do in the query is the **WHERE fd.minutesToPrepare < 30** one, so indexing it increases our performance by the most whereas subsequent filtering in

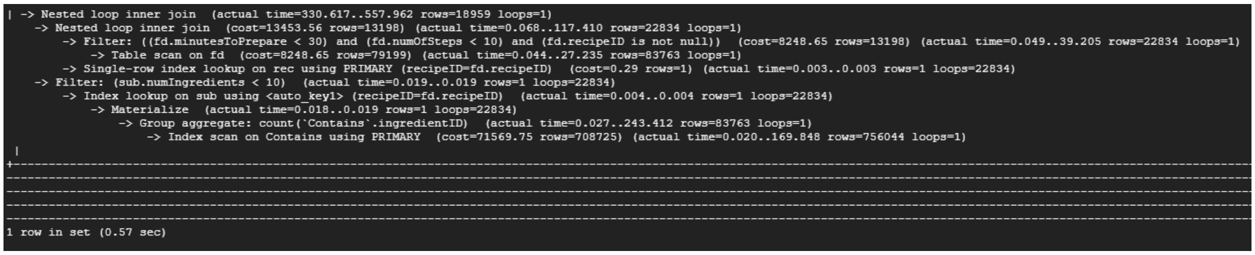
WHERE has less of an effect. Ultimately, this index still isn't a major improvement to our SQL query performance on the database.

After adding index called idx_fd_numOfSteps



```
mysql> CREATE INDEX idx_fd_numOfSteps ON FilterData(numOfSteps);
Query OK, 0 rows affected (0.46 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM FilterData;
+------------+------------+-------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name          | Seq_in_index | Column_name  | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+-------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| FilterData |          0 | PRIMARY           |            1 | filterdataID | A         |       79192 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | recipeID          |            1 | recipeID     | A         |       78177 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | idx_fd_numOfSteps |            1 | numOfSteps   | A         |          31 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+-------------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.01 sec)
```



```
| -> Nested loop inner join  (actual time=330.617..557.962 rows=18959 loops=1)
    -> Nested loop inner join  (cost=13453.56 rows=13198) (actual time=0.068..117.410 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=13198) (actual time=0.049..39.205 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.044..27.235 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.019..0.019 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.004..0.004 rows=1 loops=22834)
            -> Materialize  (actual time=0.018..0.019 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.027..243.412 rows=83763 loops=1)
                    -> Index scan on Contains using PRIMARY  (cost=71569.75 rows=708725) (actual time=0.020..169.848 rows=756044 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.57 sec)
```

Nested loop inner join (actual time=330.617..557.962 rows=18959 loops=1)

Nested loop inner join (cost=13453.56 rows=13198) (actual time=0.068..117.410 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=13198) (actual time=0.049..39.205 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.044..27.235 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.29 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.019..0.019 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.004..0.004 rows=1 loops=22834)

Materialize (actual time=0.018..0.019 rows=1 loops=22834)

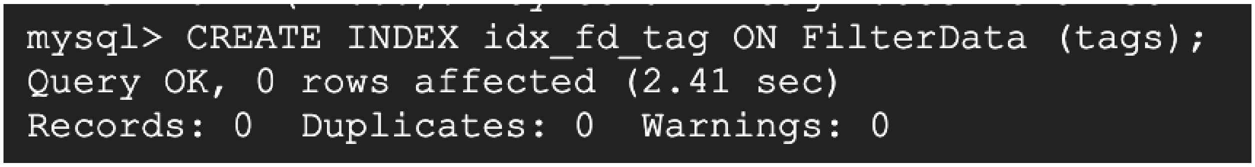
Group aggregate: count(`Contains`.ingredientID) (actual time=0.027..243.412 rows=83763 loops=1)

Index scan on Contains using PRIMARY (cost=71569.75 rows=708725) (actual time=0.020..169.848 rows=756044 loops=1)

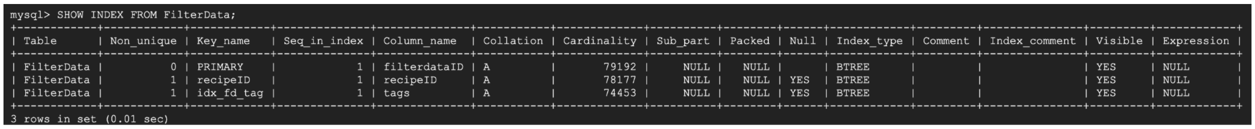
1 row in set (0.57 sec)

-> Our group did not choose the **numOfSteps** of the **FilterData** table as an index. As mentioned in the discussion for the index we added for minutesToPrepare, this index does not result in better performance and results in basically the same exact performance as the original query with default indexing). We believe that this lack of improvement is because most of the rows are pruned out during the first filtering step based on fd.minutesToPrepare so additional indexing does not do much. Therefore, we also did not attempt to analyze an index on a further filtering step on numIngredients.

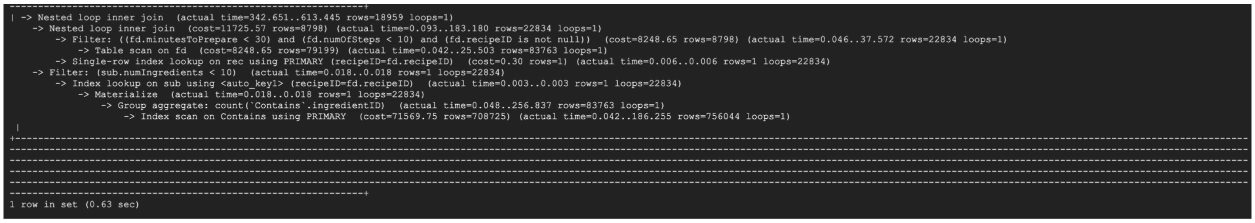
After adding index called idx_fd_tag



```
mysql> CREATE INDEX idx_fd_tag ON FilterData (tags);
Query OK, 0 rows affected (2.41 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
mysql> SHOW INDEX FROM FilterData;
+------------+------------+------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name   | Seq_in_index | Column_name  | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| FilterData |          0 | PRIMARY    |            1 | filterdataID | A         |       79192 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | recipeID   |            1 | recipeID     | A         |       78177 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | idx_fd_tag |            1 | tags         | A         |       74453 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+------------+--------------+--------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.01 sec)
```



```
------------------------------------------------------------+
| -> Nested loop inner join  (actual time=342.651..613.445 rows=18959 loops=1)
    -> Nested loop inner join  (cost=11725.57 rows=8798) (actual time=0.093..183.180 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=8798) (actual time=0.046..37.572 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.042..25.503 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.30 rows=1) (actual time=0.006..0.006 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.018..0.018 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.003..0.003 rows=1 loops=22834)
            -> Materialize  (actual time=0.018..0.018 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.048..256.837 rows=83763 loops=1)
                    -> Index scan on Contains using PRIMARY  (cost=71569.75 rows=708725) (actual time=0.042..186.255 rows=756044 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------+
1 row in set (0.63 sec)
```

Nested loop inner join (actual time=342.651..613.445 rows=18959 loops=1)

Nested loop inner join (cost=11725.57 rows=8798) (actual time=0.093..183.180 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=8798) (actual time=0.046..37.572 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.042..25.503 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.30 rows=1) (actual time=0.006..0.006 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.018..0.018 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.003..0.003 rows=1 loops=22834)

Materialize (actual time=0.018..0.018 rows=1 loops=22834)

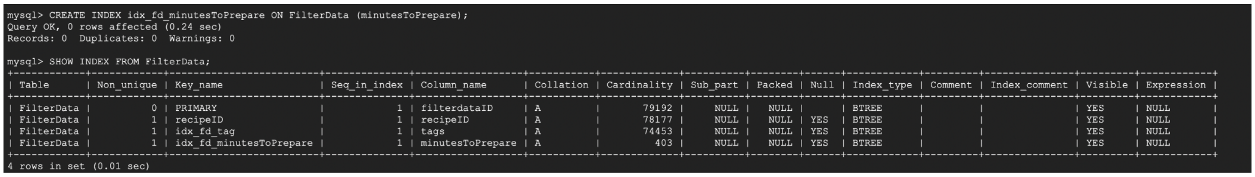
Group aggregate: count(`Contains`.ingredientID) (actual time=0.048..256.837 rows=83763 loops=1)

Index scan on Contains using PRIMARY (cost=71569.75 rows=708725) (actual time=0.042..186.255 rows=756044 loops=1)

1 row in set (0.63 sec)

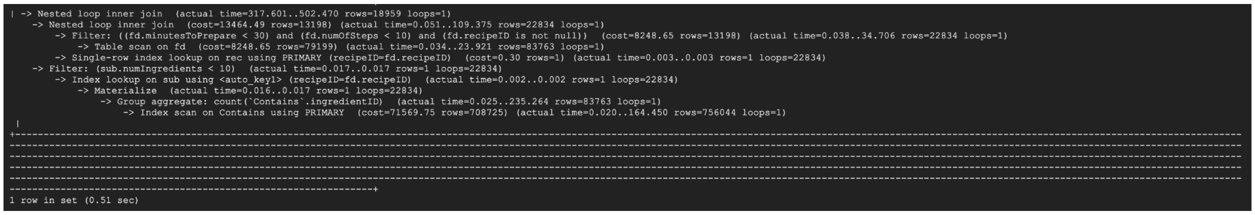
-> Our group did not choose the **idx_fd_tag** of the **FilterData** table as an index. As mentioned in previous analysis, this index does not result in better performance and results in basically the same exact performance as the original query with default indexing). We believe that this lack of improvement is because our advanced query doesn’t select tags from the FilterData table. So adding **idx_fd_tag** as index brings did not give any better performance.

After adding index called idx_fd_tag AND idx_fd_numOfSteps



```
mysql> CREATE INDEX idx_fd_minutesToPrepare ON FilterData (minutesToPrepare);
Query OK, 0 rows affected (0.24 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW INDEX FROM FilterData;
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table      | Non_unique | Key_name                | Seq_in_index | Column_name      | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| FilterData |          0 | PRIMARY                 |            1 | filterdataID     | A         |       79192 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | recipeID                |            1 | recipeID         | A         |       78177 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | idx_fd_tag              |            1 | tags             | A         |       74453 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
| FilterData |          1 | idx_fd_minutesToPrepare |            1 | minutesToPrepare | A         |         403 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+------------+------------+-------------------------+--------------+------------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
4 rows in set (0.01 sec)
```



```
| -> Nested loop inner join  (actual time=317.601..502.470 rows=18959 loops=1)
    -> Nested loop inner join  (cost=13464.49 rows=13198) (actual time=0.051..109.375 rows=22834 loops=1)
        -> Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null))  (cost=8248.65 rows=13198) (actual time=0.038..34.706 rows=22834 loops=1)
            -> Table scan on fd  (cost=8248.65 rows=79199) (actual time=0.034..23.921 rows=83763 loops=1)
        -> Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID)  (cost=0.30 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)
    -> Filter: (sub.numIngredients < 10)  (actual time=0.017..0.017 rows=1 loops=22834)
        -> Index lookup on sub using <auto_key1> (recipeID=fd.recipeID)  (actual time=0.002..0.002 rows=1 loops=22834)
            -> Materialize  (actual time=0.016..0.017 rows=1 loops=22834)
                -> Group aggregate: count(`Contains`.ingredientID)  (actual time=0.025..235.264 rows=83763 loops=1)
                    -> Index scan on Contains using PRIMARY  (cost=71569.75 rows=708725) (actual time=0.020..164.450 rows=756044 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.51 sec)
```

Nested loop inner join (actual time=317.601..502.470 rows=18959 loops=1)

Nested loop inner join (cost=13464.49 rows=13198) (actual time=0.051..109.375 rows=22834 loops=1)

Filter: ((fd.minutesToPrepare < 30) and (fd.numOfSteps < 10) and (fd.recipeID is not null)) (cost=8248.65 rows=13198) (actual time=0.038..34.706 rows=22834 loops=1)

Table scan on fd (cost=8248.65 rows=79199) (actual time=0.034..23.921 rows=83763 loops=1)

Single-row index lookup on rec using PRIMARY (recipeID=fd.recipeID) (cost=0.30 rows=1) (actual time=0.003..0.003 rows=1 loops=22834)

Filter: (sub.numIngredients < 10) (actual time=0.017..0.017 rows=1 loops=22834)

Index lookup on sub using <auto_key1> (recipeID=fd.recipeID) (actual time=0.002..0.002 rows=1 loops=22834)

Materialize (actual time=0.016..0.017 rows=1 loops=22834)

Group aggregate: count(`Contains`.ingredientID) (actual time=0.025..235.264 rows=83763 loops=1)

Index scan on Contains using PRIMARY (cost=71569.75 rows=708725) (actual time=0.020..164.450 rows=756044 loops=1)

1 row in set (0.51 sec)

-> Our group did not choose the **idx_fd_tag** and **idx_fd_numOfSteps** of the **FilterData** table as an index at the same time. As mentioned in previous analysis, idx_fd_numOfSteps gives better performance but idx_fd_tag doesn’t. We believe that since idx_fd_tag already doesn’t give better performance, adding **idx_fd_numOfSteps** as an index is meaningless.

RUBRIC :

1. **Does not have a submission located within the doc folder (-0.5%)**
2. **Database implementation is worth 7% and is graded (as a group) as follows:**
    1. **+3% for implementing the database tables locally or on GCP, you should provide a screenshot of the connection (i.e. showing your terminal information)**
    2. **+2.5% for providing the DDL commands for your tables. (-0.5% for each mistake)**
    3. **+1.5% for inserting at least 1000 rows in the tables. (You should do a count query to show this, 1% for each table)**
3. **Advanced Queries are worth 7% and are graded (as a group) as follows:**
    1. **+5% for developing two advanced queries (see point 4 for this stage, 2.5% each)**
    2. **+2% for providing screenshots with the top 15 rows of the query results (1% each)**
4. **Indexing Analysis is worth 8% and is graded (as a group) as follows:**
    1. **+3% on trying at least three different indexing designs (excluding the default index) for each advanced query.**
    2. **+4% on the indexing analysis reports.**
    3. **+1% on the accuracy and thoroughness of the analyses.**