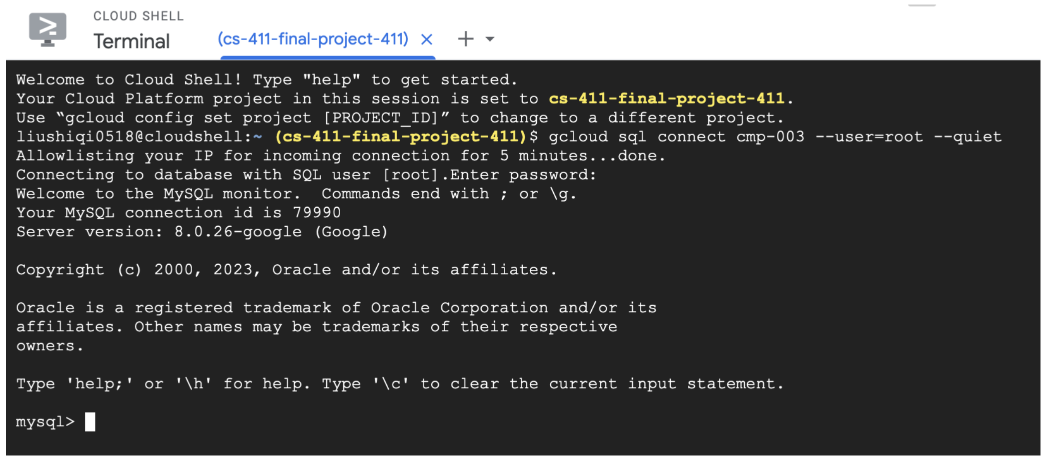
Body Watch Stage 3: Implementation in GCP

Screenshot of Connection to GCP:



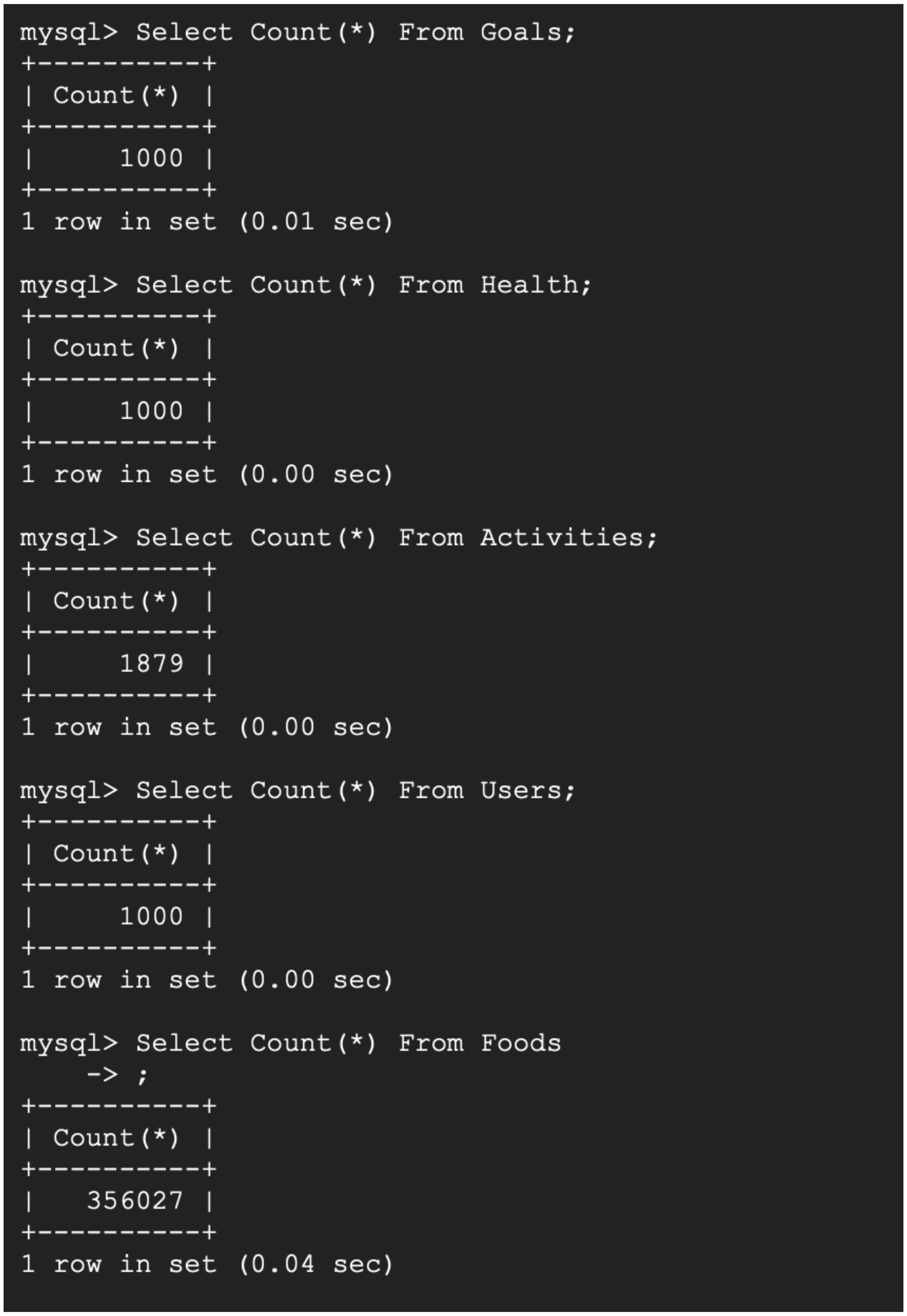
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cs-411-final-project-411.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
liushiqi0518@cloudshell:~ (cs-411-final-project-411)$ gcloud sql connect cmp-003 --user=root --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 79990
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Table DDL Commands:

```
CREATE TABLE `Activities` (
  `start_time` varchar(20) NOT NULL,
  `user_id` int NOT NULL,
  `exercise` varchar(100) NOT NULL,
  `end_time` varchar(20) NOT NULL,
  `date` varchar(20) NOT NULL,
  `calories_burned` int NOT NULL,
  `steps` int NOT NULL,
  `avg_heart_rate` int NOT NULL,
  PRIMARY KEY (`start_time`,`exercise`),
  KEY `user_id` (`user_id`),
  KEY `idx_burned_calories` (`calories_burned`),
  CONSTRAINT `Activities_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `Users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;


CREATE TABLE `Users` (
  `id` int NOT NULL,
  `first_name` varchar(100) NOT NULL,
  `last_name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `phone_number` varchar(100) NOT NULL,
  `weight` int NOT NULL,
  `height` int NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

CREATE TABLE `Health` (
  `user_id` int NOT NULL,
  `calories_burned` int NOT NULL,
  `steps` int NOT NULL,
  `date` varchar(50) NOT NULL,
  `avg_heart_rate` int NOT NULL,
  PRIMARY KEY (`date`,`user_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `Health_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `Users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

CREATE TABLE `Goals` (
  `user_id` int NOT NULL,
  `timeline` varchar(20) NOT NULL,
  `calories_goal` int NOT NULL,
  `steps_goal` int NOT NULL,
  `weight_goal` int NOT NULL,
  `protein_goal` int NOT NULL,
  `carb_goal` int NOT NULL,
  `fat_goal` int NOT NULL,
  PRIMARY KEY (`timeline`,`user_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `Goals_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `Users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

CREATE TABLE `Foods` (
  `FoodId` int NOT NULL,
  `ProdName` varchar(255) NOT NULL,
  `GenericName` varchar(255) NOT NULL,
  `Quantity` varchar(255) NOT NULL,
  `IngredientsText` varchar(255) NOT NULL,
  `ServSize` varchar(255) NOT NULL,
  `ukGrade` varchar(50) NOT NULL,
  `frGrade` varchar(50) NOT NULL,
  `ImageURL` varchar(255) NOT NULL,
  `Energy100g` decimal(8,2) NOT NULL,
  `EnergyFat100g` decimal(8,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
```

Note that all Users, Activities, Health, and Goals data is mock and randomized within realistic constraints. We plan to implement smart watch data in the future, however until we can collect 1000 real data points, we will use mock data.

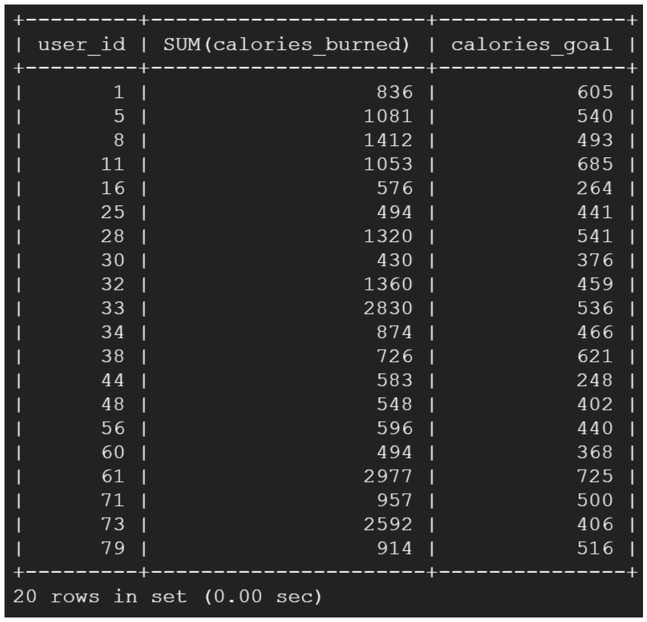
Count Data in Tables:



```
mysql> Select Count(*) From Goals;
+----------+
| Count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.01 sec)

mysql> Select Count(*) From Health;
+----------+
| Count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

mysql> Select Count(*) From Activities;
+----------+
| Count(*) |
+----------+
|     1879 |
+----------+
1 row in set (0.00 sec)

mysql> Select Count(*) From Users;
+----------+
| Count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)

mysql> Select Count(*) From Foods
    -> ;
+----------+
| Count(*) |
+----------+
|   356027 |
+----------+
1 row in set (0.04 sec)
```

Advanced SQL Queries:

1. The first Advanced SQL Query focuses on identifying those who have met their daily goals.
   SELECT user_id, SUM(calories_burned), calories_goal
   FROM Goals LEFT JOIN Activities USING (user_id)
   WHERE Goals.timeline LIKE "%Daily"
   GROUP BY user_id, calories_goal
   HAVING SUM(calories_burned) > calories_goal
   LIMIT 20;

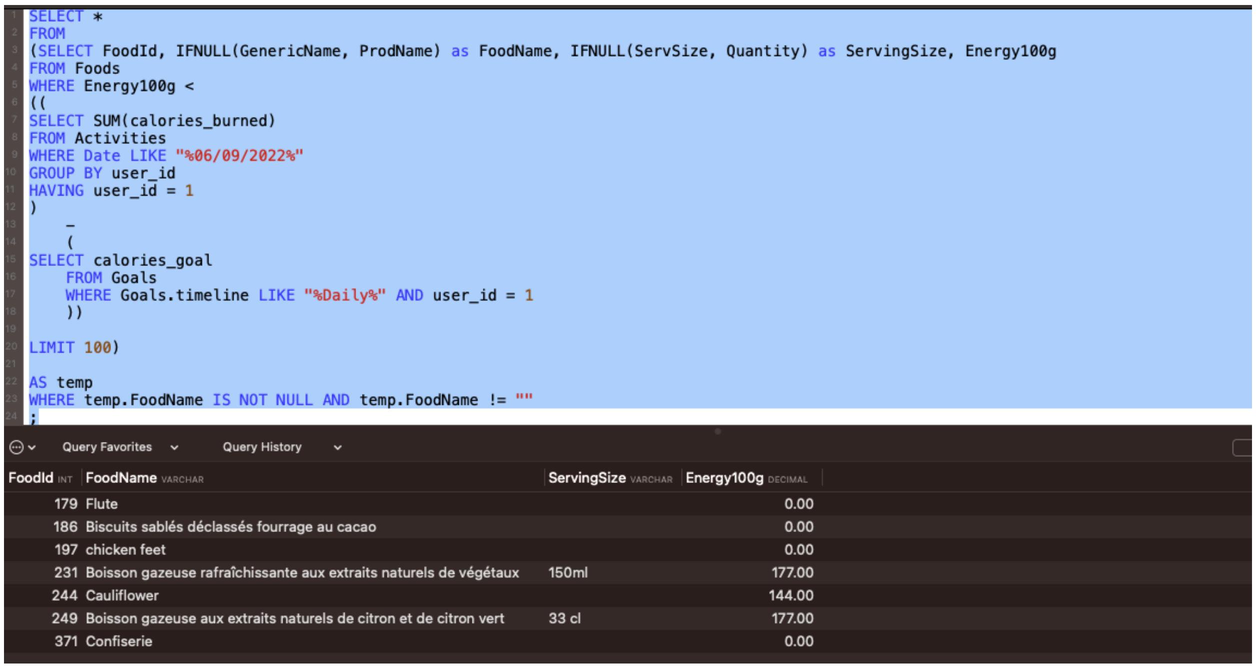


```
+---------+---------------------+---------------+
| user_id | SUM(calories_burned) | calories_goal |
+---------+---------------------+---------------+
|       1 |                 836 |           605 |
|       5 |                1081 |           540 |
|       8 |                1412 |           493 |
|      11 |                1053 |           685 |
|      16 |                 576 |           264 |
|      25 |                 494 |           441 |
|      28 |                1320 |           541 |
|      30 |                 430 |           376 |
|      32 |                1360 |           459 |
|      33 |                2830 |           536 |
|      34 |                 874 |           466 |
|      38 |                 726 |           621 |
|      44 |                 583 |           248 |
|      48 |                 548 |           402 |
|      56 |                 596 |           440 |
|      60 |                 494 |           368 |
|      61 |                2977 |           725 |
|      71 |                 957 |           500 |
|      73 |                2592 |           406 |
|      79 |                 914 |           516 |
+---------+---------------------+---------------+
20 rows in set (0.00 sec)
```

   296 rows if we do not limit the number of rows.

2. The second advanced SQL query will be to suggest foods for a user who has met their goal and can indulge in a yummy meal. (User 1!). In the future, we plan to use a cursor to make food suggestions for all who meet their goals.
   SELECT *
   FROM
   (SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName,
   IFNULL(ServSize, Quantity) as ServingSize, Energy100g
   FROM Foods
   WHERE Energy100g <
   ((
   SELECT SUM(calories_burned)
   FROM Activities
   WHERE Date LIKE "%06/09/2022%"
   GROUP BY user_id
   HAVING user_id = 1
)

```
        -
(
SELECT calories_goal
FROM Goals
WHERE Goals.timeline LIKE "%Daily%" AND user_id = 1
)))
AS temp
WHERE temp.FoodName IS NOT NULL AND temp.FoodName != ""
;
```



```
SELECT *
FROM
(SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName, IFNULL(ServSize, Quantity) as ServingSize, Energy100g
FROM Foods
WHERE Energy100g <
((
SELECT SUM(calories_burned)
FROM Activities
WHERE Date LIKE "%06/09/2022%"
GROUP BY user_id
HAVING user_id = 1
)
    -
    (
SELECT calories_goal
    FROM Goals
    WHERE Goals.timeline LIKE "%Daily%" AND user_id = 1
    ))

LIMIT 100)

AS temp
WHERE temp.FoodName IS NOT NULL AND temp.FoodName != ""
;
```

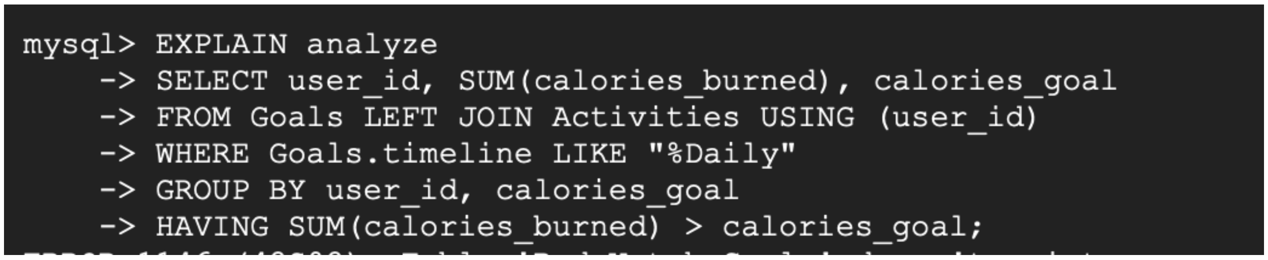
FoodId INT	FoodName VARCHAR	ServingSize VARCHAR	Energy100g DECIMAL
179	Flute		0.00
186	Biscuits sablés déclassés fourrage au cacao		0.00
197	chicken feet		0.00
231	Boisson gazeuse rafraîchissante aux extraits naturels de végétaux	150ml	177.00
244	Cauliflower		144.00
249	Boisson gazeuse aux extraits naturels de citron et de citron vert	33 cl	177.00
371	Confiserie		0.00

For this query, our output is less than 15 rows because of data with no food names. As you can see, there is also data with 0 calories which will need to be filtered out as well.

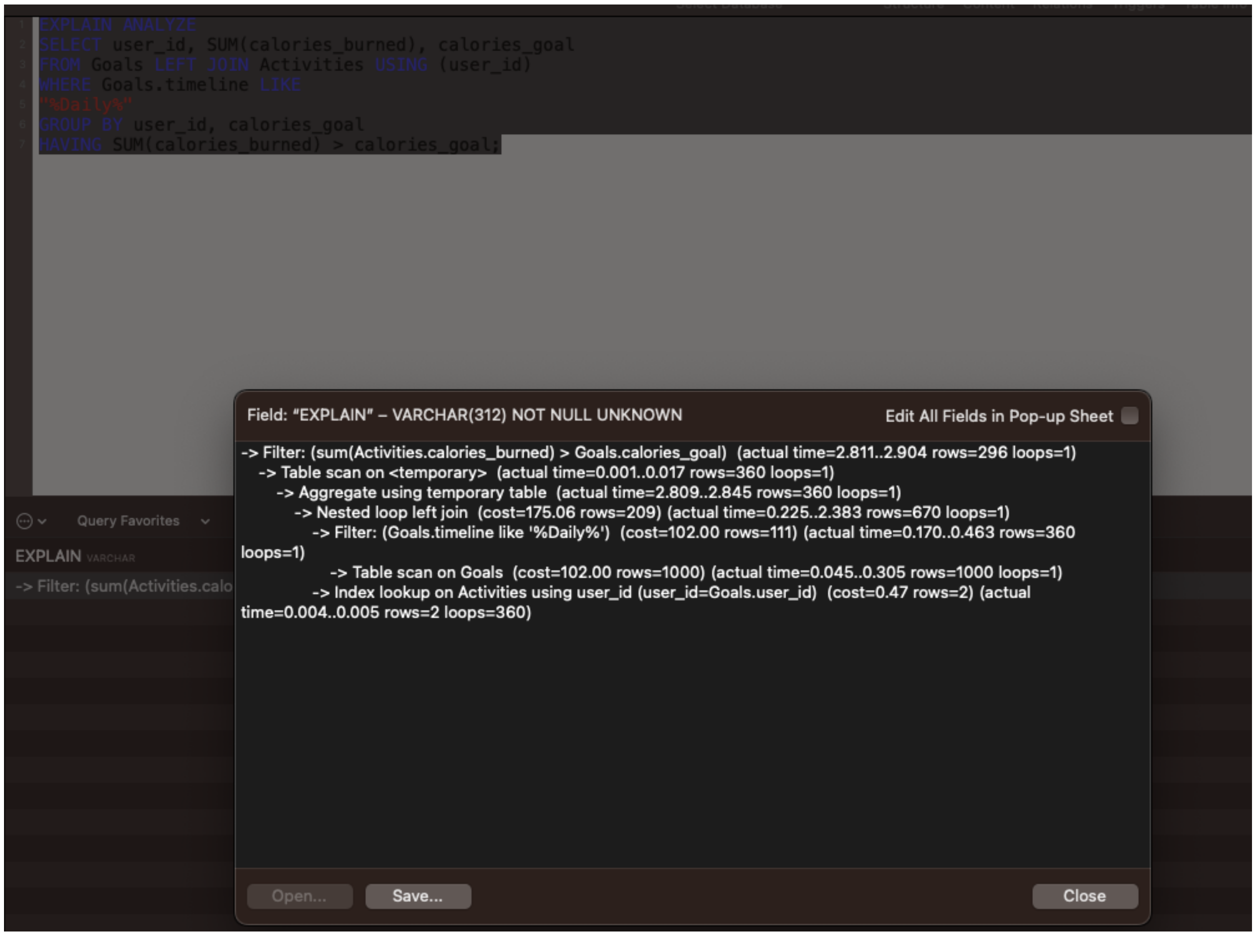
Index Design:

Query 1:

1. Initial run of ‘EXPLAIN ANALYZE’ without creating an index.



```
mysql> EXPLAIN analyze
    -> SELECT user_id, SUM(calories_burned), calories_goal
    -> FROM Goals LEFT JOIN Activities USING (user_id)
    -> WHERE Goals.timeline LIKE "%Daily"
    -> GROUP BY user_id, calories_goal
    -> HAVING SUM(calories_burned) > calories_goal;
```



```
EXPLAIN ANALYZE
SELECT user_id, SUM(calories_burned), calories_goal
FROM Goals LEFT JOIN Activities USING (user_id)
WHERE Goals.timeline LIKE
"%Daily%"
GROUP BY user_id, calories_goal
HAVING SUM(calories_burned) > calories_goal;
```

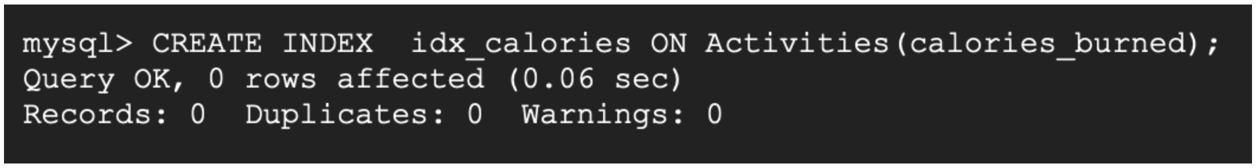
Field: "EXPLAIN" – VARCHAR(312) NOT NULL UNKNOWN

```
-> Filter: (sum(Activities.calories_burned) > Goals.calories_goal)  (actual time=2.811..2.904 rows=296 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.017 rows=360 loops=1)
        -> Aggregate using temporary table  (actual time=2.809..2.845 rows=360 loops=1)
            -> Nested loop left join  (cost=175.06 rows=209) (actual time=0.225..2.383 rows=670 loops=1)
                -> Filter: (Goals.timeline like '%Daily%')  (cost=102.00 rows=111) (actual time=0.170..0.463 rows=360 loops=1)
                    -> Table scan on Goals  (cost=102.00 rows=1000) (actual time=0.045..0.305 rows=1000 loops=1)
                -> Index lookup on Activities using user_id (user_id=Goals.user_id)  (cost=0.47 rows=2) (actual time=0.004..0.005 rows=2 loops=360)
```

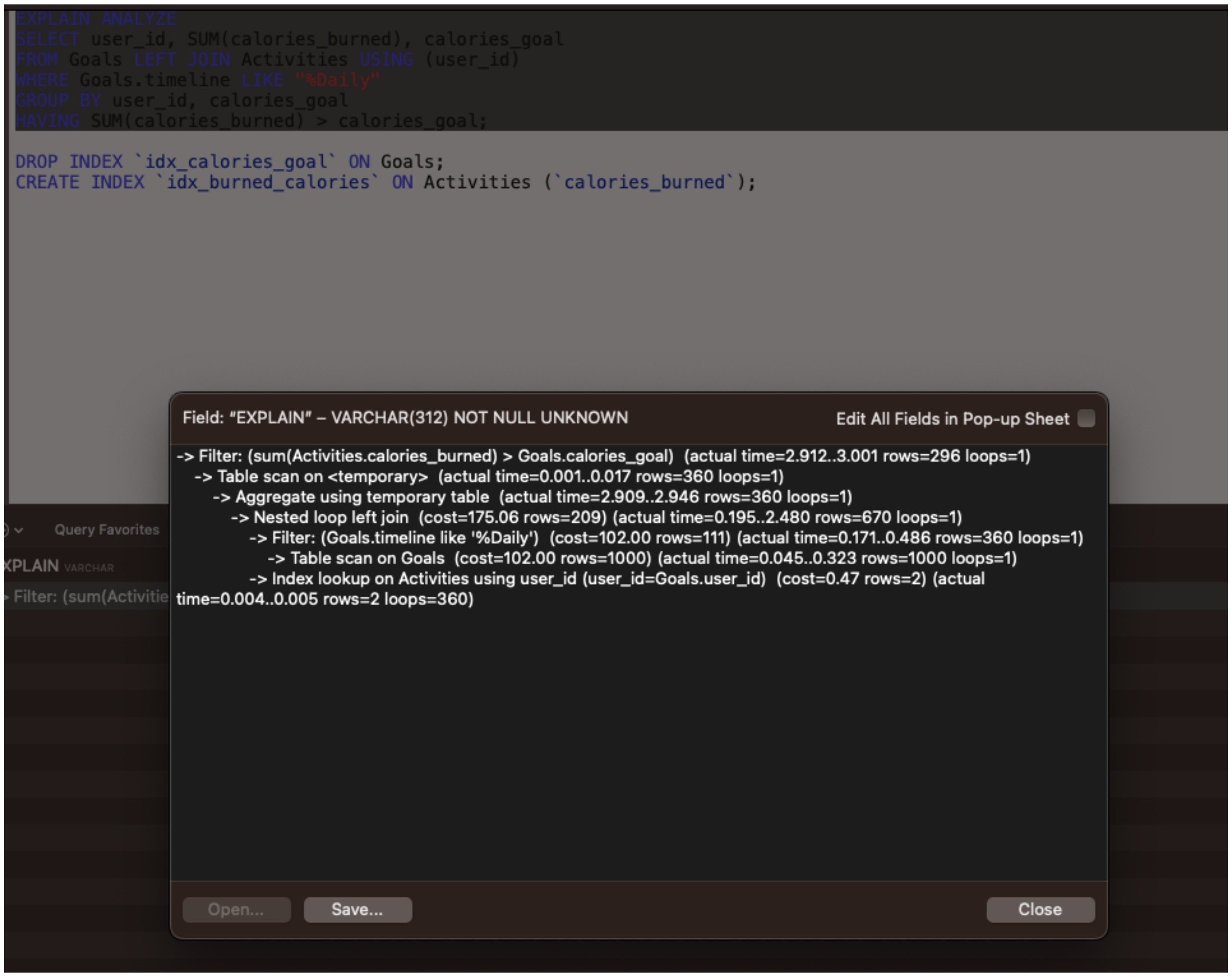
Edit All Fields in Pop-up Sheet
Query Favorites
EXPLAIN VARCHAR
-> Filter: (sum(Activities.calo
Open...
Save...
Close

2. We added an index idx_burned_calories on Activities(calories_burned) to check the performance of the query.
   CREATE INDEX idx_burned_calories ON Activities(calories_burned);

   EXPLAIN ANALYZE
   SELECT user_id, SUM(calories_burned), calories_goal
   FROM Goals LEFT JOIN Activities USING (user_id)
   WHERE Goals.timeline LIKE "%Daily"
   GROUP BY user_id, calories_goal
   HAVING SUM(calories_burned) > calories_goal
   LIMIT 20;



```
mysql> CREATE INDEX  idx_calories ON Activities(calories_burned);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



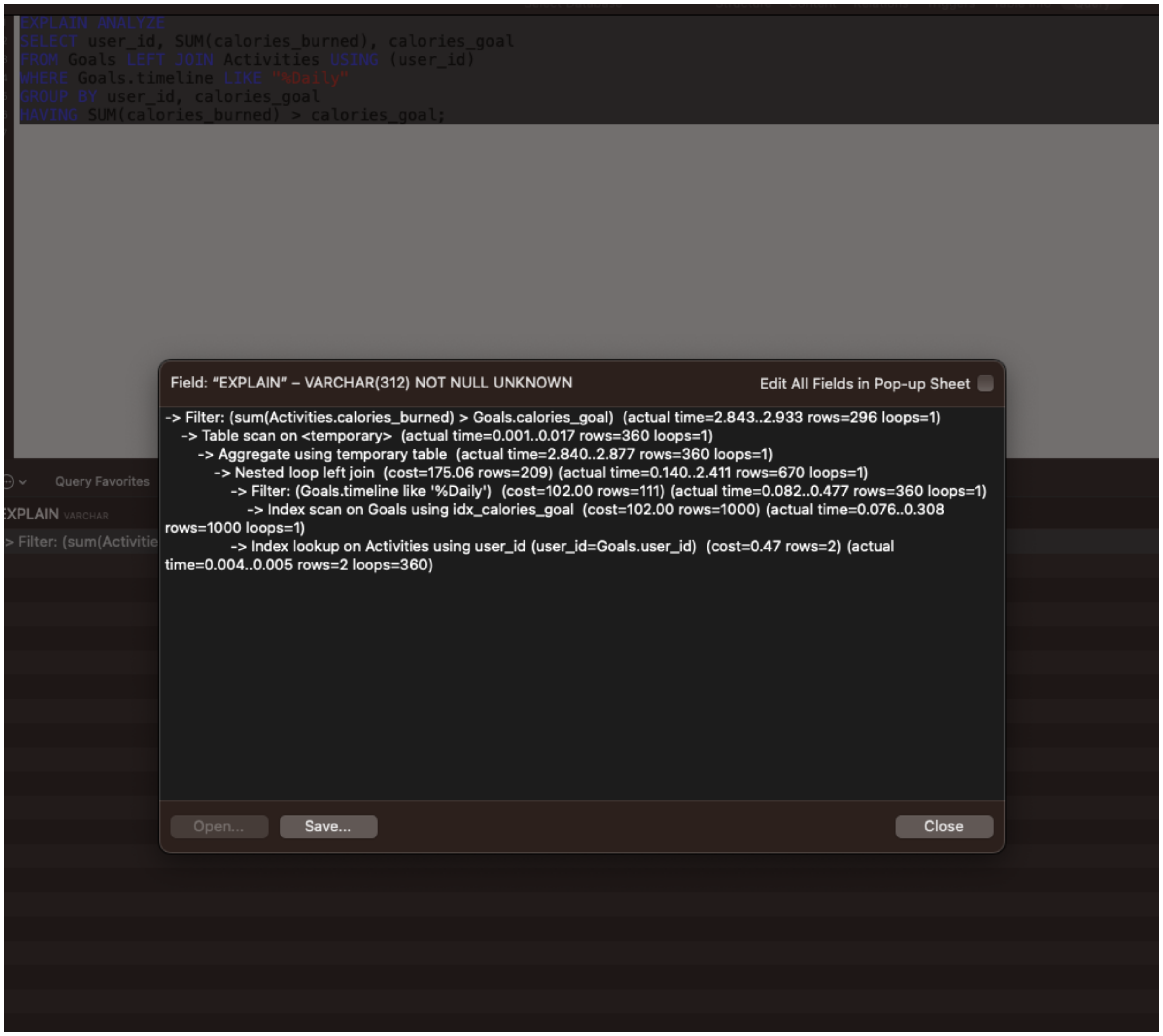
```
EXPLAIN ANALYZE
SELECT user_id, SUM(calories_burned), calories_goal
FROM Goals LEFT JOIN Activities USING (user_id)
WHERE Goals.timeline LIKE "%Daily"
GROUP BY user_id, calories_goal
HAVING SUM(calories_burned) > calories_goal;

DROP INDEX `idx_calories_goal` ON Goals;
CREATE INDEX `idx_burned_calories` ON Activities (`calories_burned`);
```

Field: "EXPLAIN" – VARCHAR(312) NOT NULL UNKNOWN

```
-> Filter: (sum(Activities.calories_burned) > Goals.calories_goal)  (actual time=2.912..3.001 rows=296 loops=1)
   -> Table scan on <temporary>  (actual time=0.001..0.017 rows=360 loops=1)
      -> Aggregate using temporary table  (actual time=2.909..2.946 rows=360 loops=1)
         -> Nested loop left join  (cost=175.06 rows=209) (actual time=0.195..2.480 rows=670 loops=1)
            -> Filter: (Goals.timeline like '%Daily')  (cost=102.00 rows=111) (actual time=0.171..0.486 rows=360 loops=1)
               -> Table scan on Goals  (cost=102.00 rows=1000) (actual time=0.045..0.323 rows=1000 loops=1)
            -> Index lookup on Activities using user_id (user_id=Goals.user_id)  (cost=0.47 rows=2) (actual
time=0.004..0.005 rows=2 loops=360)
```

Result: After analyzing the query with the idx_calories index, we found that adding the index did not significantly improve its performance. We decided to use another index in further development.

3. We added an index idx_calories_goal on Goals(calories_goal) and kept the previous index on calories_burned.
   CREATE INDEX idx_calories_goal on Goals(calories_goal);



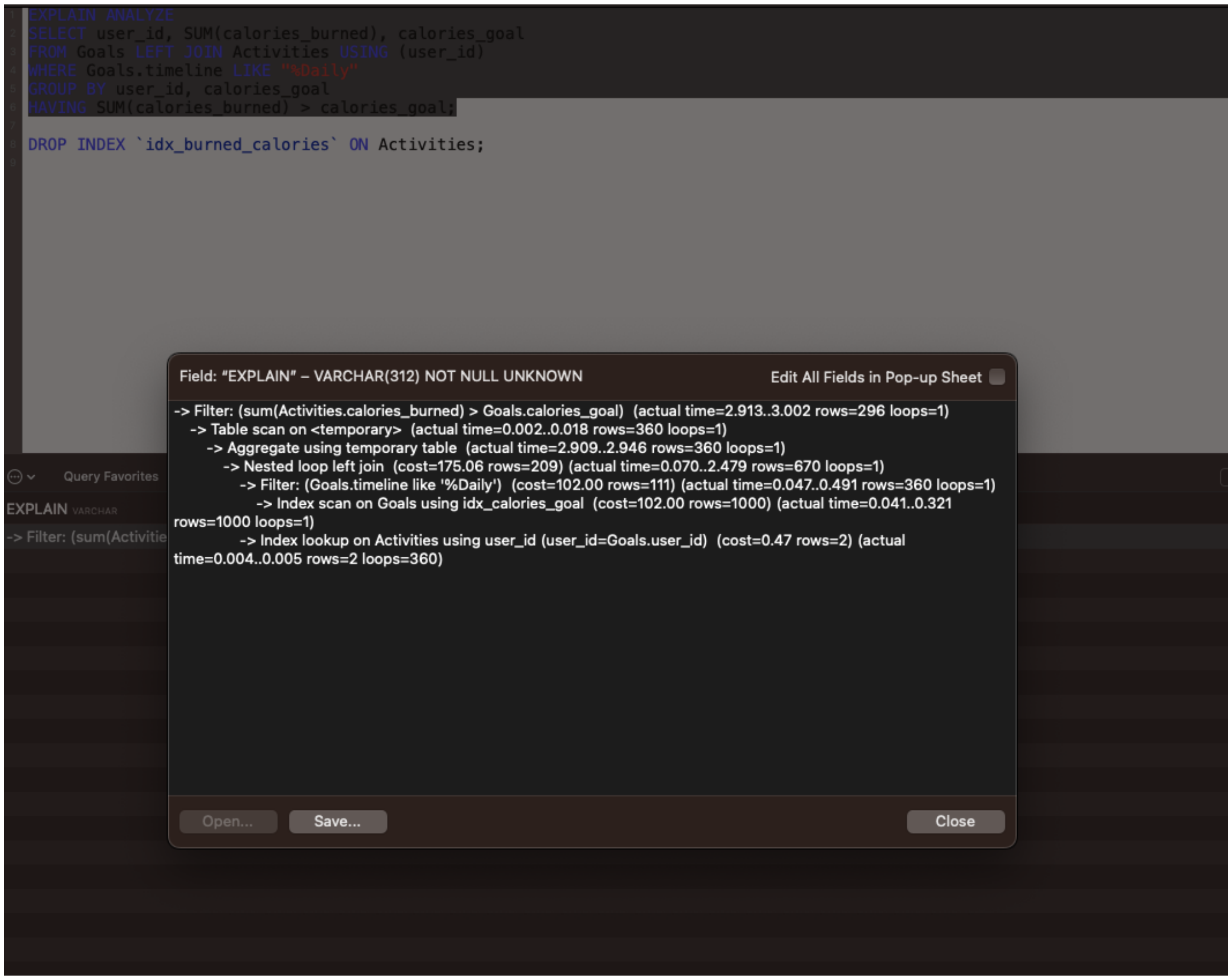
```
EXPLAIN ANALYZE
SELECT user_id, SUM(calories_burned), calories_goal
FROM Goals LEFT JOIN Activities USING (user_id)
WHERE Goals.timeline LIKE "%Daily"
GROUP BY user_id, calories_goal
HAVING SUM(calories_burned) > calories_goal;
```

Field: "EXPLAIN" – VARCHAR(312) NOT NULL UNKNOWN

```
-> Filter: (sum(Activities.calories_burned) > Goals.calories_goal)  (actual time=2.843..2.933 rows=296 loops=1)
   -> Table scan on <temporary>  (actual time=0.001..0.017 rows=360 loops=1)
      -> Aggregate using temporary table  (actual time=2.840..2.877 rows=360 loops=1)
         -> Nested loop left join  (cost=175.06 rows=209) (actual time=0.140..2.411 rows=670 loops=1)
            -> Filter: (Goals.timeline like '%Daily')  (cost=102.00 rows=111) (actual time=0.082..0.477 rows=360 loops=1)
               -> Index scan on Goals using idx_calories_goal  (cost=102.00 rows=1000) (actual time=0.076..0.308
rows=1000 loops=1)
            -> Index lookup on Activities using user_id (user_id=Goals.user_id)  (cost=0.47 rows=2) (actual
time=0.004..0.005 rows=2 loops=360)
```

Result: After analyzing the query, we noticed that having two indices made the query run faster. The speedup was less than a tenth of a second.

4. We added an index idx_calories_goal and removed the calories burned.
   DROP INDEX idx_calories ON Activities;



```sql
EXPLAIN ANALYZE
SELECT user_id, SUM(calories_burned), calories_goal
FROM Goals LEFT JOIN Activities USING (user_id)
WHERE Goals.timeline LIKE "%Daily"
GROUP BY user_id, calories_goal
HAVING SUM(calories_burned) > calories_goal;

DROP INDEX `idx_burned_calories` ON Activities;
```

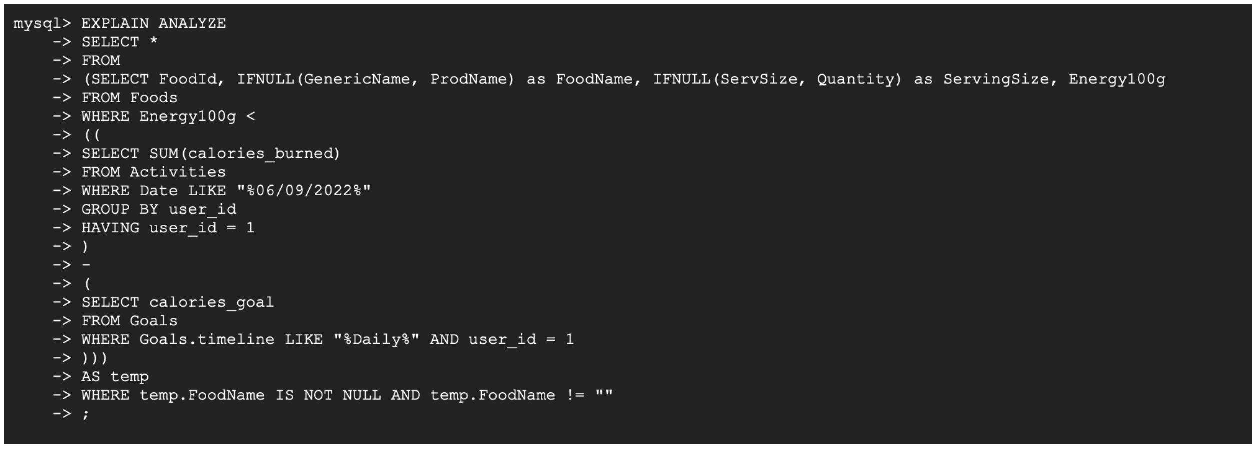
Field: "EXPLAIN" – VARCHAR(312) NOT NULL UNKNOWN

```
-> Filter: (sum(Activities.calories_burned) > Goals.calories_goal)  (actual time=2.913..3.002 rows=296 loops=1)
   -> Table scan on <temporary>  (actual time=0.002..0.018 rows=360 loops=1)
      -> Aggregate using temporary table  (actual time=2.909..2.946 rows=360 loops=1)
         -> Nested loop left join  (cost=175.06 rows=209) (actual time=0.070..2.479 rows=670 loops=1)
            -> Filter: (Goals.timeline like '%Daily')  (cost=102.00 rows=111) (actual time=0.047..0.491 rows=360 loops=1)
               -> Index scan on Goals using idx_calories_goal  (cost=102.00 rows=1000) (actual time=0.041..0.321
rows=1000 loops=1)
            -> Index lookup on Activities using user_id (user_id=Goals.user_id)  (cost=0.47 rows=2) (actual
time=0.004..0.005 rows=2 loops=360)
```

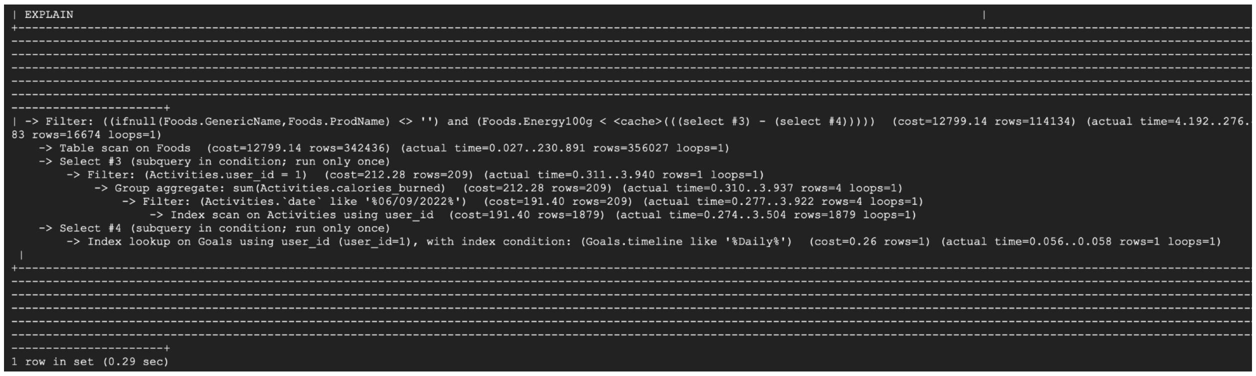
Result: There was insignificant difference between an index on calories_goal and calories_burned.

Query 2:
1st Index:

1. Initial run of 'EXPLAIN ANALYZE' without creating an index.

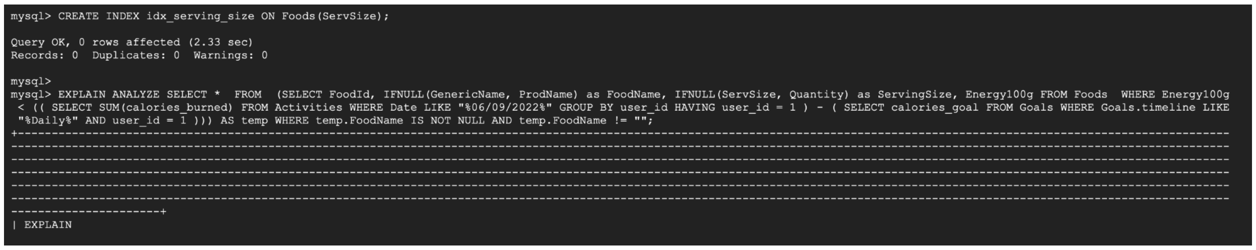


```
mysql> EXPLAIN ANALYZE
    -> SELECT *
    -> FROM
    -> (SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName, IFNULL(ServSize, Quantity) as ServingSize, Energy100g
    -> FROM Foods
    -> WHERE Energy100g <
    -> ((
    -> SELECT SUM(calories_burned)
    -> FROM Activities
    -> WHERE Date LIKE "%06/09/2022%"
    -> GROUP BY user_id
    -> HAVING user_id = 1
    -> )
    -> -
    -> (
    -> SELECT calories_goal
    -> FROM Goals
    -> WHERE Goals.timeline LIKE "%Daily%" AND user_id = 1
    -> )))
    -> AS temp
    -> WHERE temp.FoodName IS NOT NULL AND temp.FoodName != ""
    -> ;
```

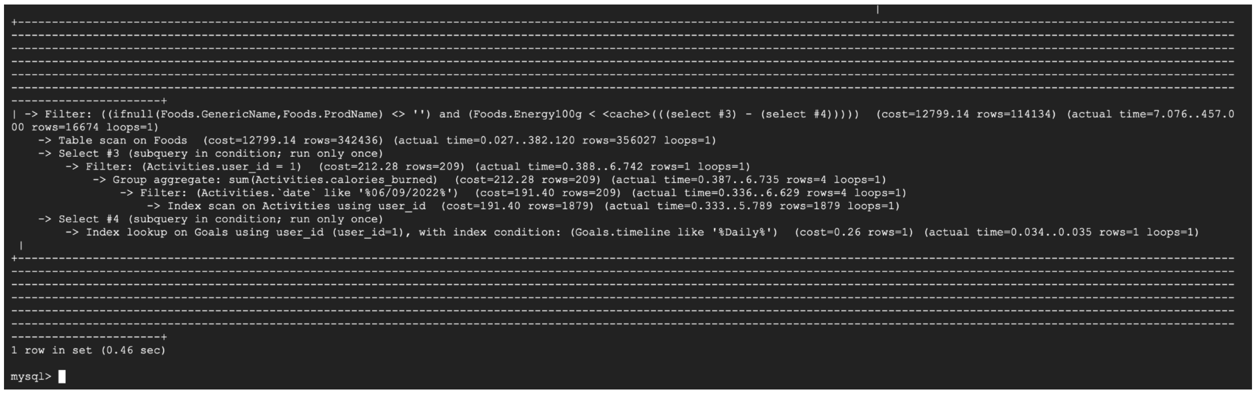


```
| EXPLAIN                                                                                                                          |
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Filter: ((ifnull(Foods.GenericName,Foods.ProdName) <> '') and (Foods.Energy100g < <cache>(((select #3) - (select #4)))))  (cost=12799.14 rows=114134) (actual time=4.192..276.
83 rows=16674 loops=1)
    -> Table scan on Foods  (cost=12799.14 rows=342436) (actual time=0.027..230.891 rows=356027 loops=1)
    -> Select #3 (subquery in condition; run only once)
        -> Filter: (Activities.user_id = 1)  (cost=212.28 rows=209) (actual time=0.311..3.940 rows=1 loops=1)
            -> Group aggregate: sum(Activities.calories_burned)  (cost=212.28 rows=209) (actual time=0.310..3.937 rows=4 loops=1)
                -> Filter: (Activities.`date` like '%06/09/2022%')  (cost=191.40 rows=209) (actual time=0.277..3.922 rows=4 loops=1)
                    -> Index scan on Activities using user_id  (cost=191.40 rows=1879) (actual time=0.274..3.504 rows=1879 loops=1)
    -> Select #4 (subquery in condition; run only once)
        -> Index lookup on Goals using user_id (user_id=1), with index condition: (Goals.timeline like '%Daily%')  (cost=0.26 rows=1) (actual time=0.056..0.058 rows=1 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.29 sec)
```

2. We added an index **idx_serving_size** on **Foods(ServSize)** to check the performance of the query.
   CREATE INDEX idx_serving_size ON Foods(ServSize);



```
mysql> CREATE INDEX idx_serving_size ON Foods(ServSize);

Query OK, 0 rows affected (2.33 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> EXPLAIN ANALYZE SELECT *  FROM  (SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName, IFNULL(ServSize, Quantity) as ServingSize, Energy100g FROM Foods  WHERE Energy100g
 < (( SELECT SUM(calories_burned) FROM Activities WHERE Date LIKE "%06/09/2022%" GROUP BY user_id HAVING user_id = 1 ) - ( SELECT calories_goal FROM Goals WHERE Goals.timeline LIKE
 "%Daily%" AND user_id = 1 ))) AS temp WHERE temp.FoodName IS NOT NULL AND temp.FoodName != "";
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| EXPLAIN
```

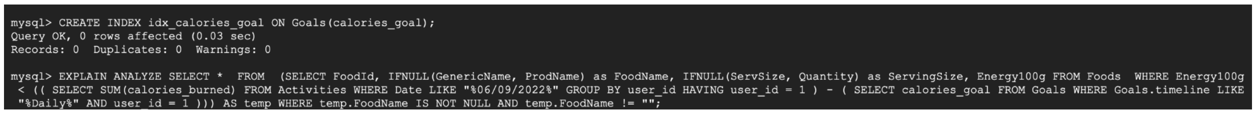


```
                                                                                                                           |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Filter: ((ifnull(Foods.GenericName,Foods.ProdName) <> '') and (Foods.Energy100g < <cache>(((select #3) - (select #4)))))  (cost=12799.14 rows=114134) (actual time=7.076..457.0
00 rows=16674 loops=1)
    -> Table scan on Foods  (cost=12799.14 rows=342436) (actual time=0.027..382.120 rows=356027 loops=1)
    -> Select #3 (subquery in condition; run only once)
        -> Filter: (Activities.user_id = 1)  (cost=212.28 rows=209) (actual time=0.388..6.742 rows=1 loops=1)
            -> Group aggregate: sum(Activities.calories_burned)  (cost=212.28 rows=209) (actual time=0.387..6.735 rows=4 loops=1)
                -> Filter: (Activities.`date` like '%06/09/2022%')  (cost=191.40 rows=209) (actual time=0.336..6.629 rows=4 loops=1)
                    -> Index scan on Activities using user_id  (cost=191.40 rows=1879) (actual time=0.333..5.789 rows=1879 loops=1)
    -> Select #4 (subquery in condition; run only once)
        -> Index lookup on Goals using user_id (user_id=1), with index condition: (Goals.timeline like '%Daily%')  (cost=0.26 rows=1) (actual time=0.034..0.035 rows=1 loops=1)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.46 sec)

mysql> █
```

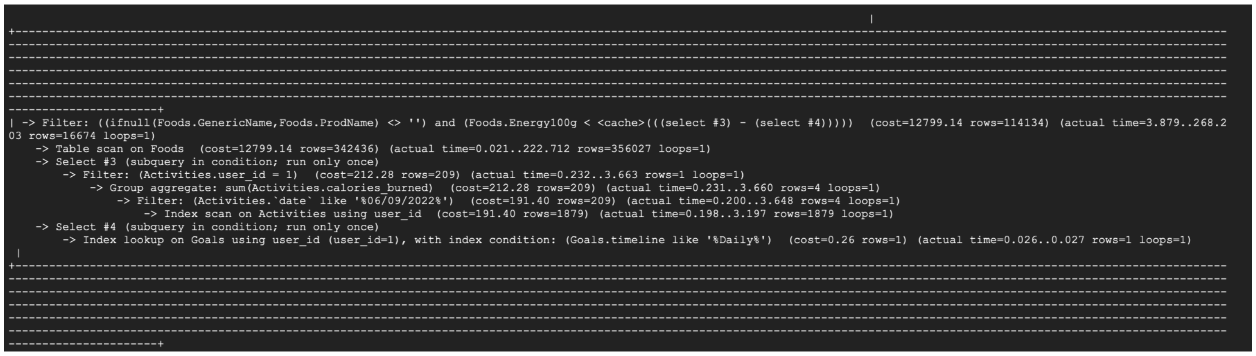
3. Result:There was insignificant difference on idx_serving_size

2nd Index:

1. We added an index on the column idx_calories_goal and removed the previous one.
   CREATE INDEX idx_calories_goal ON Goals(calories_goal);



```
mysql> CREATE INDEX idx_calories_goal ON Goals(calories_goal);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT *  FROM  (SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName, IFNULL(ServSize, Quantity) as ServingSize, Energy100g FROM Foods  WHERE Energy100g
 < (( SELECT SUM(calories_burned) FROM Activities WHERE Date LIKE "%06/09/2022%" GROUP BY user_id HAVING user_id = 1 ) - ( SELECT calories_goal FROM Goals WHERE Goals.timeline LIKE
 "%Daily%" AND user_id = 1 ))) AS temp WHERE temp.FoodName IS NOT NULL AND temp.FoodName != "";
```

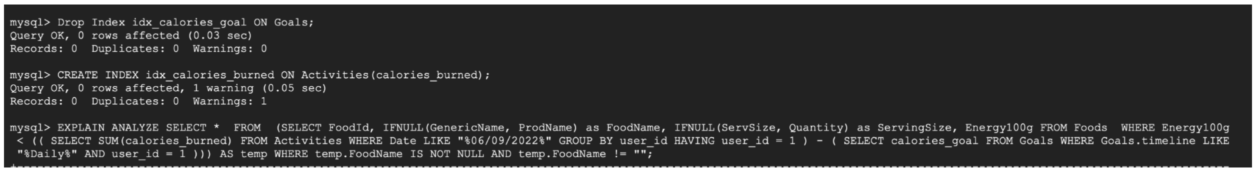


```
                                                                                                                                    |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Filter: ((ifnull(Foods.GenericName,Foods.ProdName) <> '') and (Foods.Energy100g < <cache>(((select #3) - (select #4)))))  (cost=12799.14 rows=114134) (actual time=3.879..268.2
03 rows=16674 loops=1)
    -> Table scan on Foods  (cost=12799.14 rows=342436) (actual time=0.021..222.712 rows=356027 loops=1)
    -> Select #3 (subquery in condition; run only once)
        -> Filter: (Activities.user_id = 1)  (cost=212.28 rows=209) (actual time=0.232..3.663 rows=1 loops=1)
            -> Group aggregate: sum(Activities.calories_burned)  (cost=212.28 rows=209) (actual time=0.231..3.660 rows=4 loops=1)
                -> Filter: (Activities.`date` like '%06/09/2022%')  (cost=191.40 rows=209) (actual time=0.200..3.648 rows=4 loops=1)
                    -> Index scan on Activities using user_id  (cost=191.40 rows=1879) (actual time=0.198..3.197 rows=1879 loops=1)
    -> Select #4 (subquery in condition; run only once)
        -> Index lookup on Goals using user_id (user_id=1), with index condition: (Goals.timeline like '%Daily%')  (cost=0.26 rows=1) (actual time=0.026..0.027 rows=1 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
```

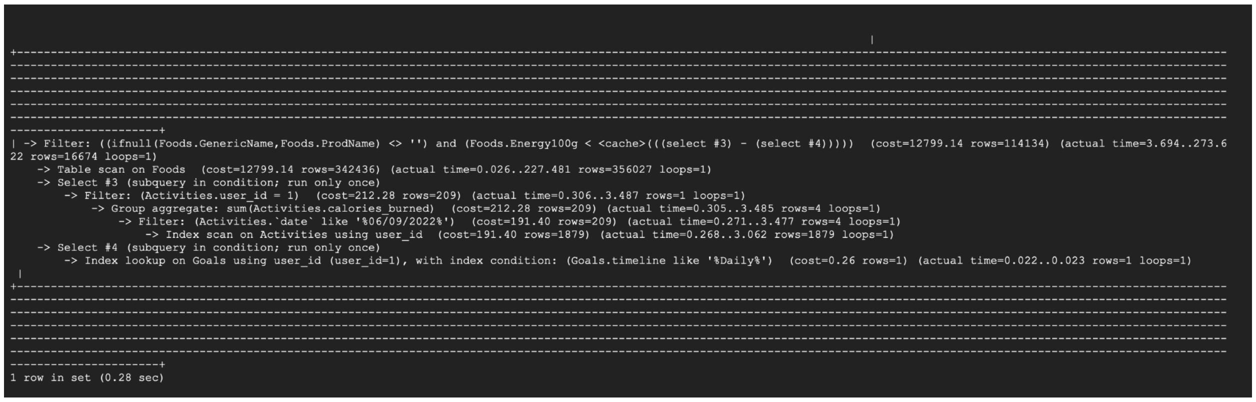
2. Result: There was insignificant difference on calories_goal

3rd Index:

1. We added an index on the column idx_calories_goal and remove the previous one.
   CREATE INDEX idx_calories_burned ON Activities(calories_burned);



```
mysql> Drop Index idx_calories_goal ON Goals;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_calories_burned ON Activities(calories_burned);
Query OK, 0 rows affected, 1 warning (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> EXPLAIN ANALYZE SELECT *  FROM  (SELECT FoodId, IFNULL(GenericName, ProdName) as FoodName, IFNULL(ServSize, Quantity) as ServingSize, Energy100g FROM Foods  WHERE Energy100g
 < (( SELECT SUM(calories_burned) FROM Activities WHERE Date LIKE "%06/09/2022%" GROUP BY user_id HAVING user_id = 1 ) - ( SELECT calories_goal FROM Goals WHERE Goals.timeline LIKE
 "%Daily%" AND user_id = 1 ))) AS temp WHERE temp.FoodName IS NOT NULL AND temp.FoodName != "";
```



```
                                                                                                                                    |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Filter: ((ifnull(Foods.GenericName,Foods.ProdName) <> '') and (Foods.Energy100g < <cache>(((select #3) - (select #4)))))  (cost=12799.14 rows=114134) (actual time=3.694..273.6
22 rows=16674 loops=1)
    -> Table scan on Foods  (cost=12799.14 rows=342436) (actual time=0.026..227.481 rows=356027 loops=1)
    -> Select #3 (subquery in condition; run only once)
        -> Filter: (Activities.user_id = 1)  (cost=212.28 rows=209) (actual time=0.306..3.487 rows=1 loops=1)
            -> Group aggregate: sum(Activities.calories_burned)  (cost=212.28 rows=209) (actual time=0.305..3.485 rows=4 loops=1)
                -> Filter: (Activities.`date` like '%06/09/2022%')  (cost=191.40 rows=209) (actual time=0.271..3.477 rows=4 loops=1)
                    -> Index scan on Activities using user_id  (cost=191.40 rows=1879) (actual time=0.268..3.062 rows=1879 loops=1)
    -> Select #4 (subquery in condition; run only once)
        -> Index lookup on Goals using user_id (user_id=1), with index condition: (Goals.timeline like '%Daily%')  (cost=0.26 rows=1) (actual time=0.022..0.023 rows=1 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set (0.28 sec)
```

2. Result: There was a significant difference on calories_burned

   Based on the analysis, it can be concluded that the utilization of the idx_calories_burned index resulted in a decrease in the overall time taken for the process compared to the scenario where the index was not used. Hence, it can be inferred that idx_calories_burned is a suitable and effective index for our specific use case.