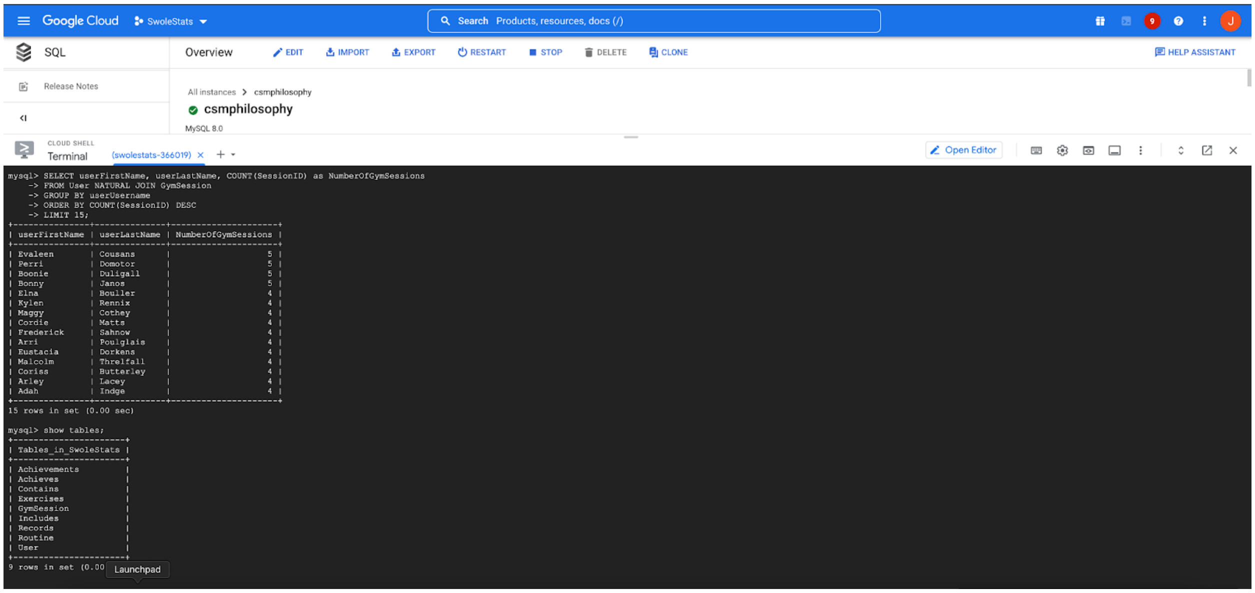
NetIds: dgroves2, jacobdr4, nuox3

GCP Terminal Screenshot



```
mysql> SELECT userFirstName, userLastName, COUNT(SessionID) as NumberOfGymSessions
    -> FROM User NATURAL JOIN GymSession
    -> GROUP BY userUsername
    -> ORDER BY COUNT(SessionID) DESC
    -> LIMIT 15;
+---------------+--------------+---------------------+
| userFirstName | userLastName | NumberOfGymSessions |
+---------------+--------------+---------------------+
| Evaleen       | Cousans      |                   5 |
| Perri         | Domotor      |                   5 |
| Boonie        | Duligall     |                   5 |
| Bonny         | Janos        |                   5 |
| Elna          | Bouller      |                   4 |
| Kylen         | Rennix       |                   4 |
| Maggy         | Cothey       |                   4 |
| Cordie        | Matts        |                   4 |
| Frederick     | Sahnow       |                   4 |
| Arri          | Poulglais    |                   4 |
| Eustacia      | Dorkens      |                   4 |
| Malcolm       | Threlfall    |                   4 |
| Coriss        | Butterley    |                   4 |
| Arley         | Lacey        |                   4 |
| Adah          | Indge        |                   4 |
+---------------+--------------+---------------------+
15 rows in set (0.00 sec)

mysql> show tables;
+----------------------+
| Tables_in_SwoleStats |
+----------------------+
| Achievements         |
| Achieves             |
| Contains             |
| Exercises            |
| GymSession           |
| Includes             |
| Records              |
| Routine              |
| User                 |
+----------------------+
9 rows in set (0.00
```

NetIds: dgroves2, jacobdr4, nuox3

DDL Commands

```
CREATE TABLE User (
        userUsername VARCHAR(20),
        userPassword VARCHAR(20),
        userFirstName VARCHAR(20),
        userLastName VARCHAR(20),
        userGender VARCHAR(10),
        userAge INT,
        userWeight FLOAT,
        userHeight FLOAT,
        PRIMARY KEY(userUsername)
);


CREATE TABLE GymSession (
        sessionID INT NOT NULL AUTO_INCREMENT,
        userUsername VARCHAR(20),
        routineID INT,
        sessionDate DATE,
        sessionStartTime TIME,
        sessionEndTime TIME,
        currentWeight FLOAT,
        PRIMARY KEY (sessionID),
        FOREIGN KEY (userUsername)
                REFERENCES User(userUsername),
        FOREIGN KEY (routineID)
                REFERENCES Routine(routineID)
);

CREATE TABLE Achievements (
        achievementTitle VARCHAR(20),
        achievementDescription VARCHAR(100),
        PRIMARY KEY(achievementTitle)
);

CREATE TABLE Contains (
        sessionID INT,
        exerciseID INT,
        sessionExerciseReps INT,
        sessionExerciseSets INT,
        sessionExerciseWeight INT,
        PRIMARY KEY(sessionID, exerciseID),
        FOREIGN KEY(sessionID)
                REFERENCES GymSession(sessionID),
        FOREIGN KEY(exerciseID)
```

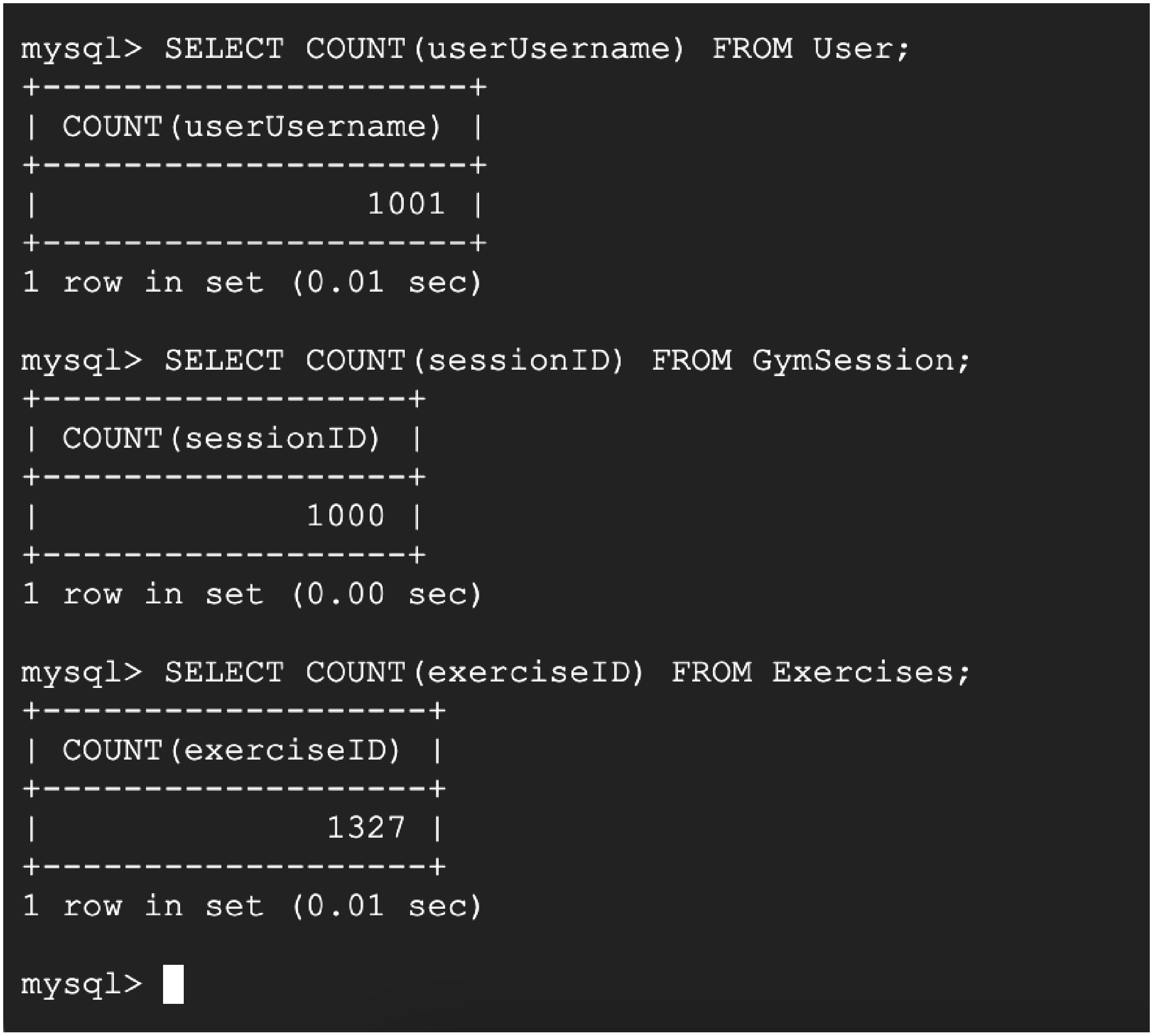
```
                REFERENCES Exercises(exerciseID)
);


CREATE TABLE Exercises (
        exerciseID INT NOT NULL AUTO_INCREMENT,
        exerciseName VARCHAR(100),
        exerciseBodyPart VARCHAR(20),
        exerciseEquipment VARCHAR(20),
        exerciseGIFURL VARCHAR(100),
        PRIMARY KEY (exerciseID)
);

CREATE TABLE Routine (
        routineID INT,
        routineName VARCHAR(20),
        PRIMARY KEY (routineID)
);


CREATE TABLE Includes (
        routineID INT,
        exerciseID INT,
        routineExerciseSets INT,
        routineExerciseReps INT,
        PRIMARY KEY (routineID, exerciseID),
        FOREIGN KEY (routineID)
                REFERENCES Routine(routineID),
        FOREIGN KEY (exerciseID)
                REFERENCES Exercises(exerciseID)
);

CREATE TABLE Achieves (
        userUsername VARCHAR(20),
        achievementTitle VARCHAR(20),
        userAchievementDate DATE,
        PRIMARY KEY (userUsername, achievementTitle),
        FOREIGN KEY (userUsername)
                REFERENCES User(userUsername),
        FOREIGN KEY (achievementTitle)
                REFERENCES Achievements(achievementTitle)
);
```

NetIds: dgroves2, jacobdr4, nuox3

```
CREATE TABLE Records (
        userUsername VARCHAR(20),
        exerciseID INT,
        prWeight INT,
        PRIMARY KEY (userUsername, exerciseID),
        FOREIGN KEY (userUsername)
                REFERENCES User(userUsername),
        FOREIGN KEY (exerciseID)
                REFERENCES Exercises(exerciseID)
);
```

NetIds: dgroves2, jacobdr4, nuox3

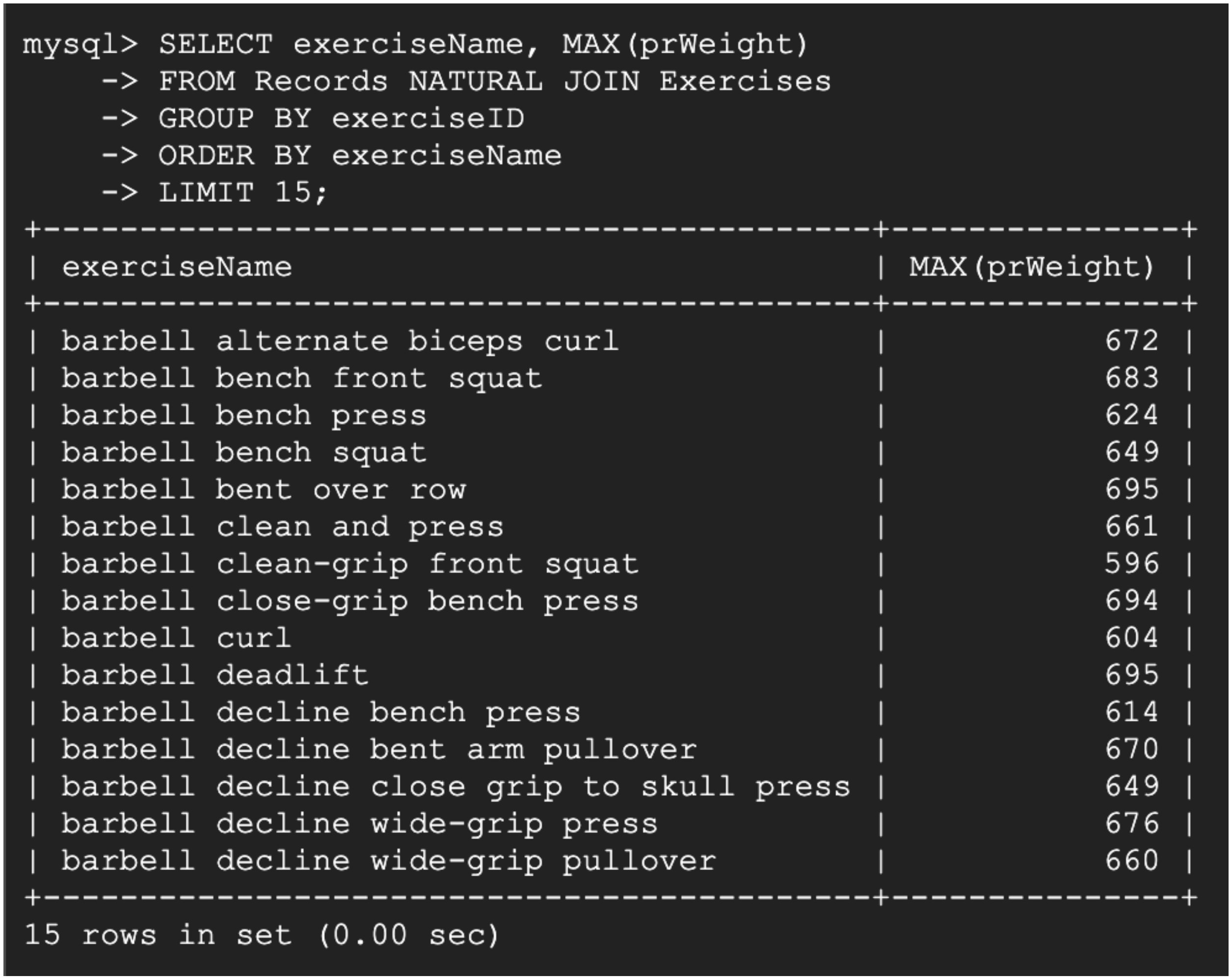
User, GymSession, Exercises Tables Have 1000+ Rows



```
mysql> SELECT COUNT(userUsername) FROM User;
+---------------------+
| COUNT(userUsername) |
+---------------------+
|                1001 |
+---------------------+
1 row in set (0.01 sec)

mysql> SELECT COUNT(sessionID) FROM GymSession;
+------------------+
| COUNT(sessionID) |
+------------------+
|             1000 |
+------------------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(exerciseID) FROM Exercises;
+-------------------+
| COUNT(exerciseID) |
+-------------------+
|              1327 |
+-------------------+
1 row in set (0.01 sec)

mysql> 
```

NetIds: dgroves2, jacobdr4, nuox3

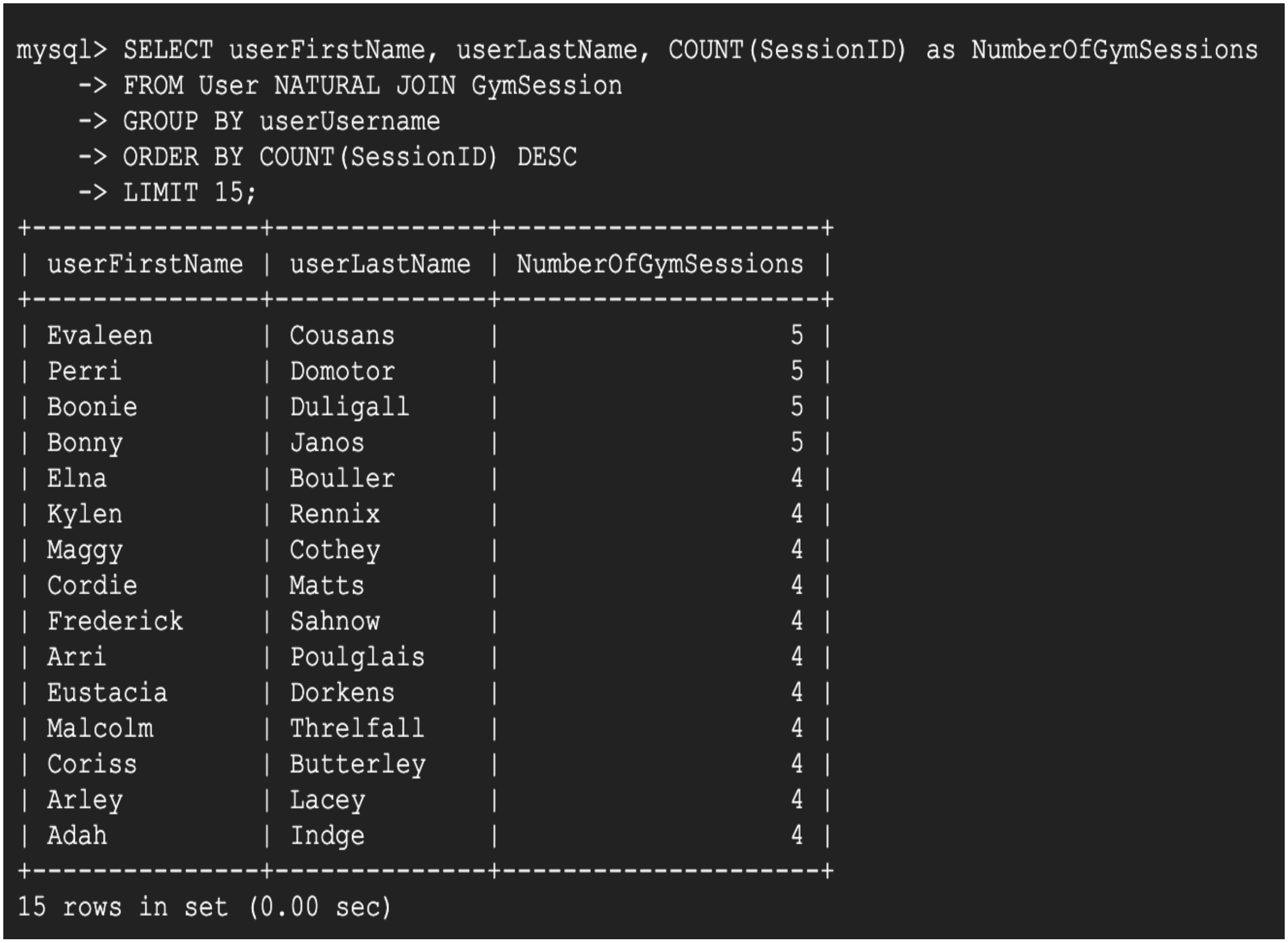
First Advanced Query + Results:
Find the highest PR for each workout



```
mysql> SELECT exerciseName, MAX(prWeight)
    -> FROM Records NATURAL JOIN Exercises
    -> GROUP BY exerciseID
    -> ORDER BY exerciseName
    -> LIMIT 15;
+--------------------------------------+---------------+
| exerciseName                         | MAX(prWeight) |
+--------------------------------------+---------------+
| barbell alternate biceps curl        |           672 |
| barbell bench front squat            |           683 |
| barbell bench press                  |           624 |
| barbell bench squat                  |           649 |
| barbell bent over row                |           695 |
| barbell clean and press              |           661 |
| barbell clean-grip front squat       |           596 |
| barbell close-grip bench press       |           694 |
| barbell curl                         |           604 |
| barbell deadlift                     |           695 |
| barbell decline bench press          |           614 |
| barbell decline bent arm pullover    |           670 |
| barbell decline close grip to skull press |      649 |
| barbell decline wide-grip press      |           676 |
| barbell decline wide-grip pullover   |           660 |
+--------------------------------------+---------------+
15 rows in set (0.00 sec)
```

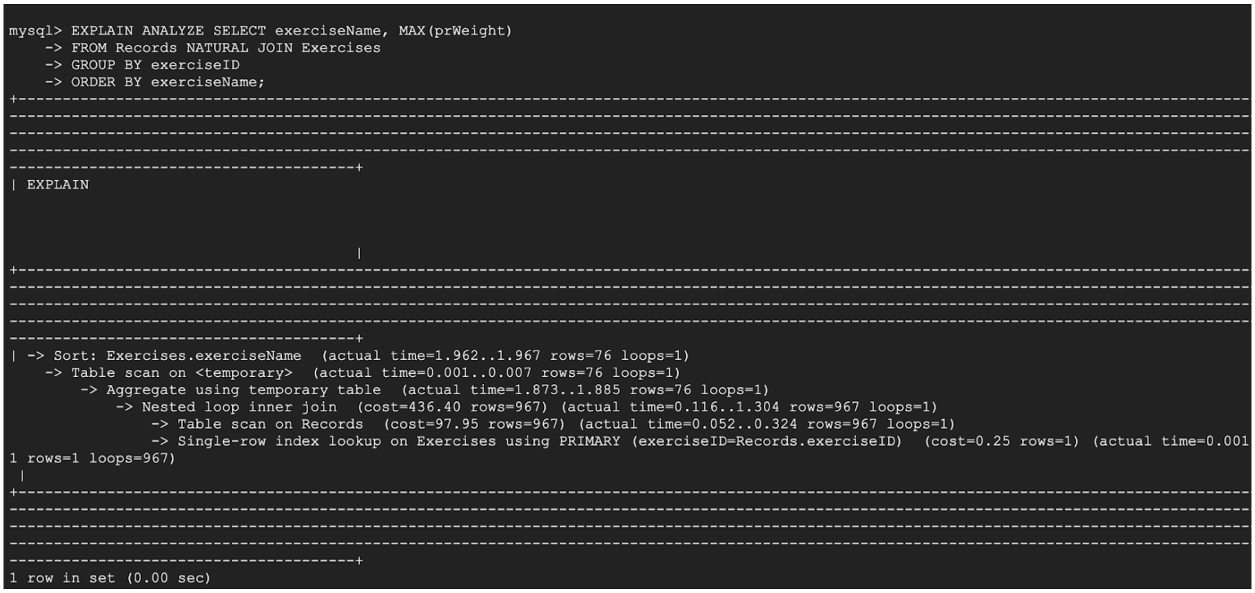
NetIds: dgroves2, jacobdr4, nuox3

Second Advanced Queries + Results:
Find the users who’ve gone to the gym the most

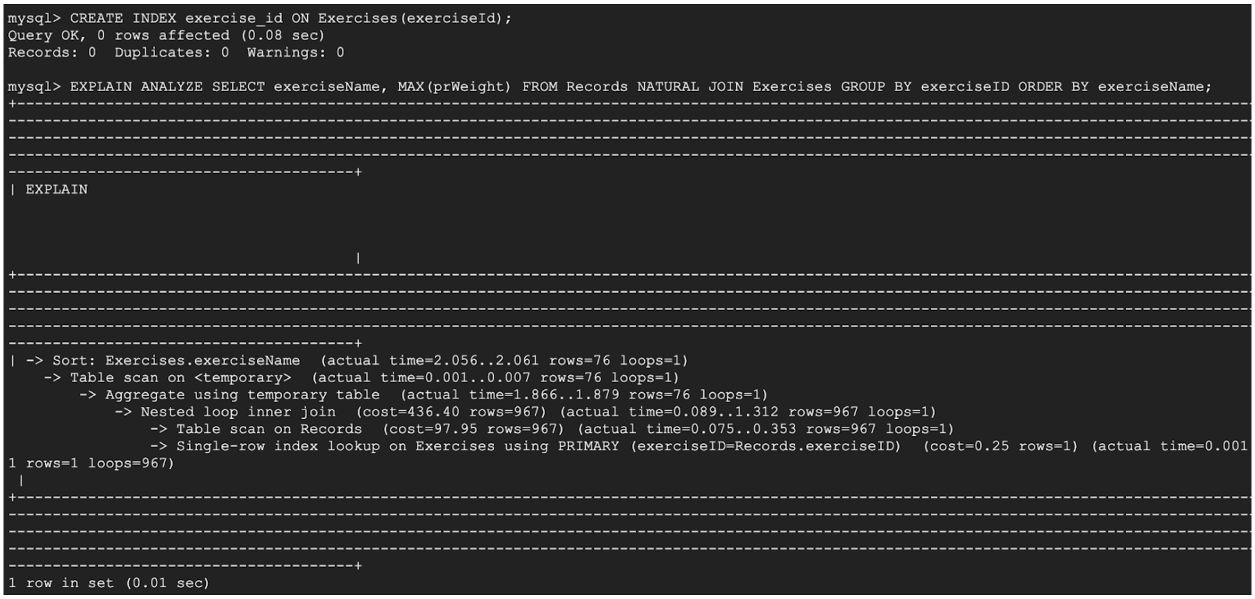


```
mysql> SELECT userFirstName, userLastName, COUNT(SessionID) as NumberOfGymSessions
    -> FROM User NATURAL JOIN GymSession
    -> GROUP BY userUsername
    -> ORDER BY COUNT(SessionID) DESC
    -> LIMIT 15;
+---------------+--------------+---------------------+
| userFirstName | userLastName | NumberOfGymSessions |
+---------------+--------------+---------------------+
| Evaleen       | Cousans      |                   5 |
| Perri         | Domotor      |                   5 |
| Boonie        | Duligall     |                   5 |
| Bonny         | Janos        |                   5 |
| Elna          | Bouller      |                   4 |
| Kylen         | Rennix       |                   4 |
| Maggy         | Cothey       |                   4 |
| Cordie        | Matts        |                   4 |
| Frederick     | Sahnow       |                   4 |
| Arri          | Poulglais    |                   4 |
| Eustacia      | Dorkens      |                   4 |
| Malcolm       | Threlfall    |                   4 |
| Coriss        | Butterley    |                   4 |
| Arley         | Lacey        |                   4 |
| Adah          | Indge        |                   4 |
+---------------+--------------+---------------------+
15 rows in set (0.00 sec)
```

NetIds: dgroves2, jacobdr4, nuox3



```
mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight)
    -> FROM Records NATURAL JOIN Exercises
    -> GROUP BY exerciseID
    -> ORDER BY exerciseName;
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| EXPLAIN



                                       |
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=1.962..1.967 rows=76 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.007 rows=76 loops=1)
        -> Aggregate using temporary table  (actual time=1.873..1.885 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.116..1.304 rows=967 loops=1)
                -> Table scan on Records  (cost=97.95 rows=967) (actual time=0.052..0.324 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.001
1 rows=1 loops=967)
 |
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
1 row in set (0.00 sec)
```

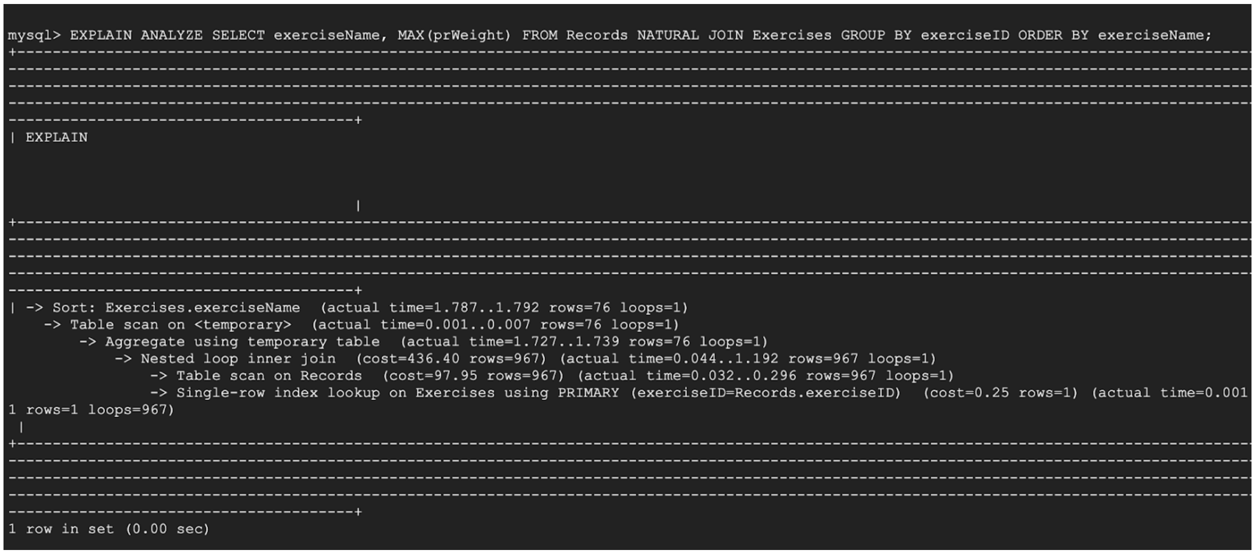


```
mysql> CREATE INDEX exercise_id ON Exercises(exerciseId);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight) FROM Records NATURAL JOIN Exercises GROUP BY exerciseID ORDER BY exerciseName;
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| EXPLAIN



                                       |
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=2.056..2.061 rows=76 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.007 rows=76 loops=1)
        -> Aggregate using temporary table  (actual time=1.866..1.879 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.089..1.312 rows=967 loops=1)
                -> Table scan on Records  (cost=97.95 rows=967) (actual time=0.075..0.353 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.001
1 rows=1 loops=967)
 |
+------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
1 row in set (0.01 sec)
```

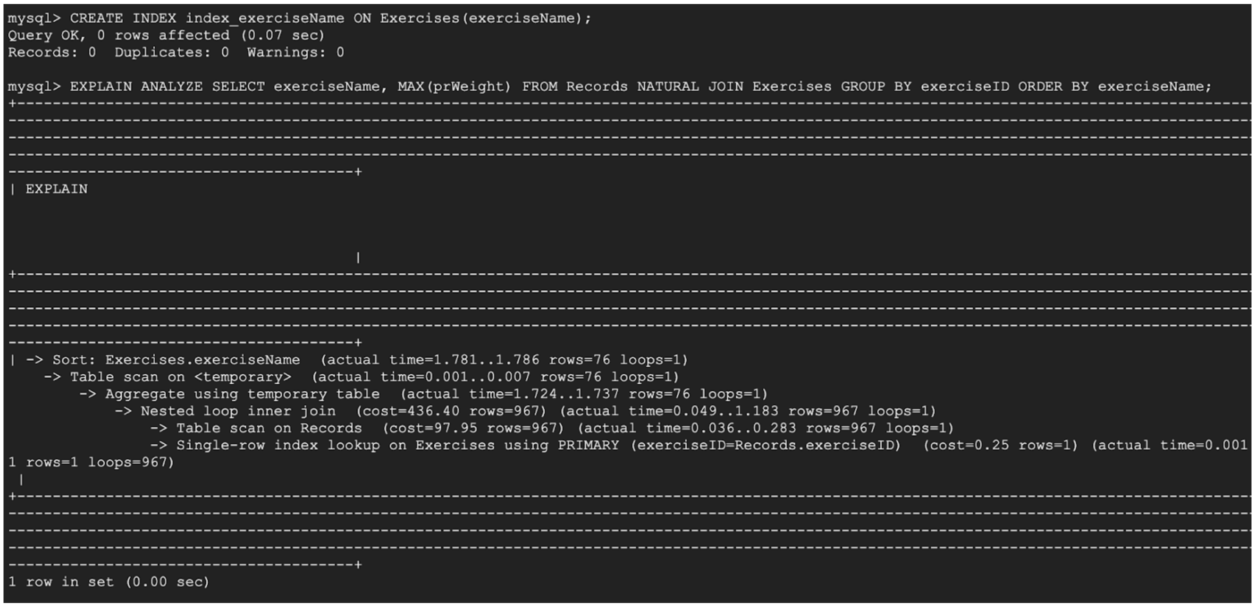
Top image: Baseline without manually adding an index
Bottom image: Adding index to exercise_id

For our first indexing design that we performed on our database was indexing on our exercise_id. We thought this would be most effective for our query as when we are joining tables we would be joining on this identification number. In addition, when we group we are using the identification number so we predicted it would help that query as well. To our surprise (which maybe came from our lack of knowledge) this resulted in the same costs as we ran our baseline without any indexing. Though the time seemed to differentiate slightly we felt as though this could be due to network traffic, and believe by default the databases use primary keys to index.

NetIds: dgroves2, jacobdr4, nuox3



```
mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight) FROM Records NATURAL JOIN Exercises GROUP BY exerciseID ORDER BY exerciseName;
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN



                                        |
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=1.787..1.792 rows=76 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.007 rows=76 loops=1)
        -> Aggregate using temporary table  (actual time=1.727..1.739 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.044..1.192 rows=967 loops=1)
                -> Table scan on Records  (cost=97.95 rows=967) (actual time=0.032..0.296 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.001
1 rows=1 loops=967)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.00 sec)
```

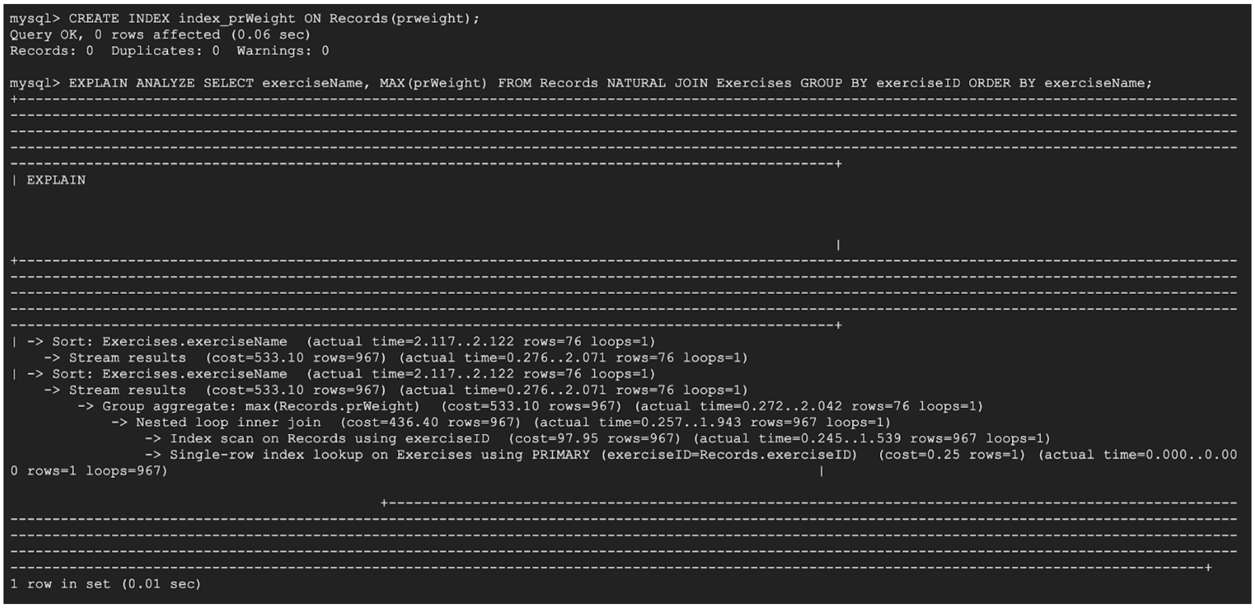


```
mysql> CREATE INDEX index_exerciseName ON Exercises(exerciseName);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight) FROM Records NATURAL JOIN Exercises GROUP BY exerciseID ORDER BY exerciseName;
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN



                                        |
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=1.781..1.786 rows=76 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.007 rows=76 loops=1)
        -> Aggregate using temporary table  (actual time=1.724..1.737 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.049..1.183 rows=967 loops=1)
                -> Table scan on Records  (cost=97.95 rows=967) (actual time=0.036..0.283 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.001
1 rows=1 loops=967)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.00 sec)
```

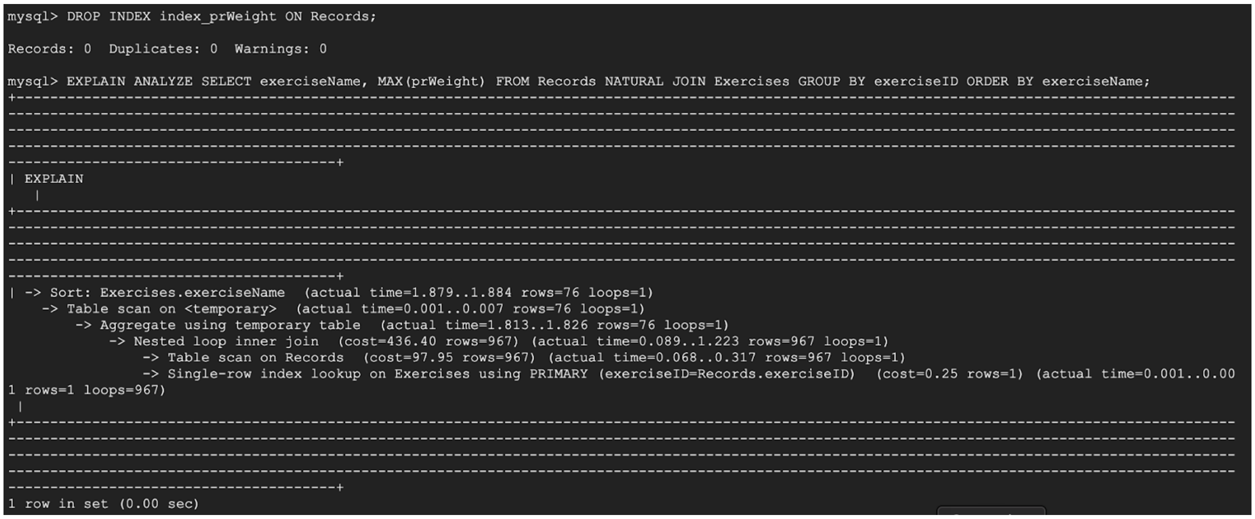
Top image: Baseline without manually adding an index
Bottom image: Adding index to exercise_name

For our second indexing design that we performed on our database was indexing on our exerciseName. Though we did not predict this to be as effective as exerciseId, as we are not joining on this property, we thought it would still be useful as we are ordering by this attribute. To our surprise this indexing method actually performed the exact same as the baseline and did not increase performance. This leads us to believe that ordering by an index does not alter efficiency.

NetIds: dgroves2, jacobdr4, nuox3



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

mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight) FROM Records NATURAL JOIN Exercises GROUP BY exerciseID ORDER BY exerciseName;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| EXPLAIN



                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=2.117..2.122 rows=76 loops=1)
    -> Stream results  (cost=533.10 rows=967) (actual time=0.276..2.071 rows=76 loops=1)
| -> Sort: Exercises.exerciseName  (actual time=2.117..2.122 rows=76 loops=1)
    -> Stream results  (cost=533.10 rows=967) (actual time=0.276..2.071 rows=76 loops=1)
        -> Group aggregate: max(Records.prWeight)  (cost=533.10 rows=967) (actual time=0.272..2.042 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.257..1.943 rows=967 loops=1)
                -> Index scan on Records using exerciseID  (cost=97.95 rows=967) (actual time=0.245..1.539 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.000..0.00
0 rows=1 loops=967)                                                                                 |

                                             +--------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```



```
mysql> DROP INDEX index_prWeight ON Records;

Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT exerciseName, MAX(prWeight) FROM Records NATURAL JOIN Exercises GROUP BY exerciseID ORDER BY exerciseName;
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| EXPLAIN
   |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
| -> Sort: Exercises.exerciseName  (actual time=1.879..1.884 rows=76 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.007 rows=76 loops=1)
        -> Aggregate using temporary table  (actual time=1.813..1.826 rows=76 loops=1)
            -> Nested loop inner join  (cost=436.40 rows=967) (actual time=0.089..1.223 rows=967 loops=1)
                -> Table scan on Records  (cost=97.95 rows=967) (actual time=0.068..0.317 rows=967 loops=1)
                -> Single-row index lookup on Exercises using PRIMARY (exerciseID=Records.exerciseID)  (cost=0.25 rows=1) (actual time=0.001..0.00
1 rows=1 loops=967)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------+
1 row in set (0.00 sec)
```

Our third indexing design was prRecords. We didn't expect it to make it much faster as PR's were only being used as aggregates. But we did not expect indexing to decrease performance this drastically, not only creating a whole other branch but also creating a branch with a larger time than the old one. We do not know why this happened, but it's clear this is a poor index design.

Looking at all three designs, we believe that the first design of indexing on exerciseId is most impactful because it is what we are performing our joins on and we are not filtering upon anything else. However, if our query was more complicated, having an index be an attribute in the where clause or in a join would make those better.