ECE 356 Lab2 Baseball

Lab section 206
Group 10

Victor Yan
Lun Jing

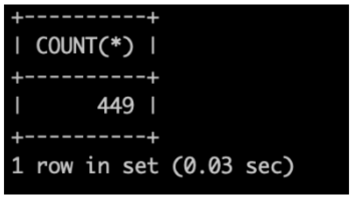
Note: all the results before adding indexes, are recorded when there is no indexes added for all related tables, except primary keys and foreign keys.

(a) How many players have an unknown birthdate?

```
EXPLAIN SELECT COUNT(*) FROM Master AS M
WHERE M.`birthDay` = 0 OR M.`birthMonth` = 0 OR M.`birthYear` =
0;
```

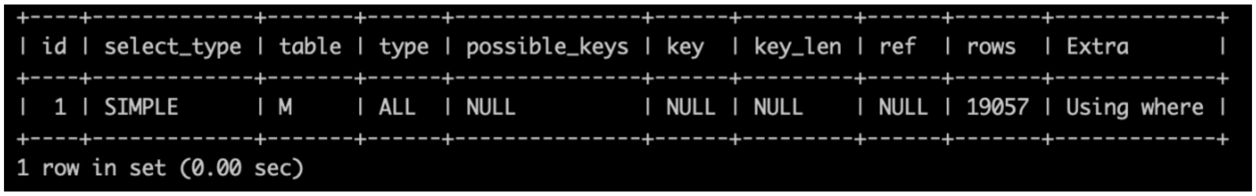
Before adding indexes:

Running time:



```
+----------+
| COUNT(*) |
+----------+
|      449 |
+----------+
1 row in set (0.03 sec)
```

Explain:

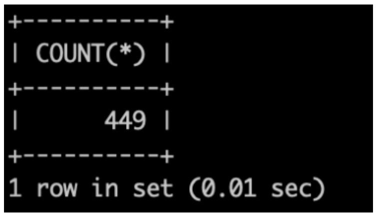


```
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows  | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
|  1 | SIMPLE      | M     | ALL  | NULL          | NULL | NULL    | NULL | 19057 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+-------+-------------+
1 row in set (0.00 sec)
```

Since it trying to find the entries where ‘birthDay’ or ‘birthMonth’ or ‘birthYear’ is empty, null, or zero (basically they mean the same thing). Therefore, the ideal proposal is to adding indexes to the “Master” table by these three attributes, ‘birthDay’, ‘birthMonth’ or ‘birthYear’, then when we go through the table, every entry with unknown birthdays have already been gathered.

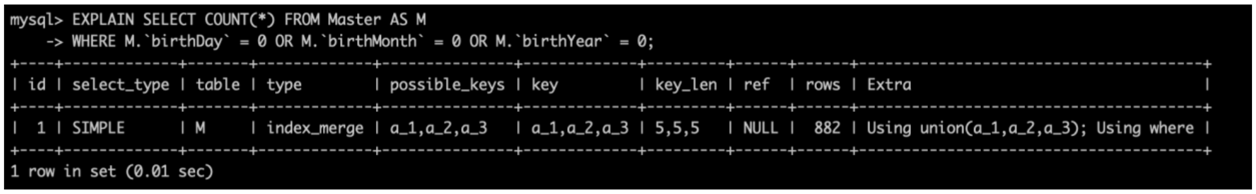
```
CREATE INDEX a_1 on Master (birthDay);
CREATE INDEX a_2 on Master (birthMonth);
CREATE INDEX a_3 on Master (birthYear);
```

After adding indexes:
Running time:



```
+----------+
| COUNT(*) |
+----------+
|      449 |
+----------+
1 row in set (0.01 sec)
```

Explain:



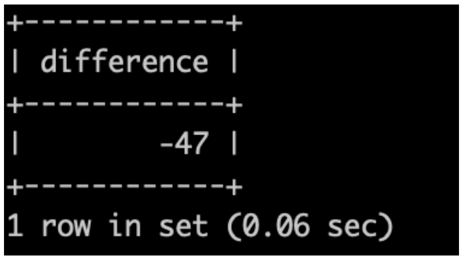
```
mysql> EXPLAIN SELECT COUNT(*) FROM Master AS M
    -> WHERE M.`birthDay` = 0 OR M.`birthMonth` = 0 OR M.`birthYear` = 0;
+----+-------------+-------+-------------+---------------+-------------+---------+------+------+-------------------------------------+
| id | select_type | table | type        | possible_keys | key         | key_len | ref  | rows | Extra                               |
+----+-------------+-------+-------------+---------------+-------------+---------+------+------+-------------------------------------+
|  1 | SIMPLE      | M     | index_merge | a_1,a_2,a_3   | a_1,a_2,a_3 | 5,5,5   | NULL |  882 | Using union(a_1,a_2,a_3); Using where |
+----+-------------+-------+-------------+---------------+-------------+---------+------+------+-------------------------------------+
1 row in set (0.01 sec)
```

As we can see the rows are reduced after indexing.

(b) Are more players in the Hall of Fame dead or alive? (Output the number alive minus the number dead)

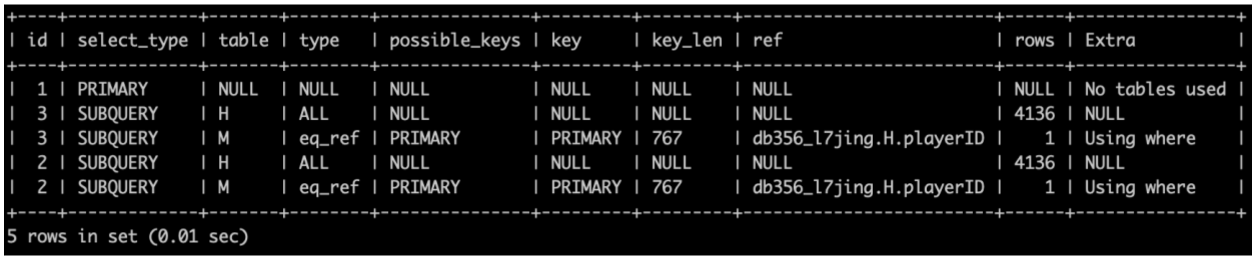
```
EXPLAIN SELECT
(SELECT count(DISTINCT M.playerID) FROM HallOfFame AS H LEFT
outer join Master AS M
ON H.playerID = M.playerID
WHERE M.deathYear = '' AND M.deathMonth = '' AND  M.deathDay = ''
AND  M.deathCountry = '' AND  M.deathState = '' AND  M.deathCity
= '')
-
(SELECT count(DISTINCT M.playerID) FROM HallOfFame AS H LEFT
outer join Master AS M
ON H.playerID = M.playerID
WHERE M.deathYear <> '' OR  M.deathMonth <> '' OR  M.deathDay <>
'' OR  M.deathCountry <> '' OR  M.deathState <> '' OR
M.deathCity <> '') as difference;
```

Before adding indexes:
Running time:



```
+------------+
| difference |
+------------+
|        -47 |
+------------+
1 row in set (0.06 sec)
```

Explain:



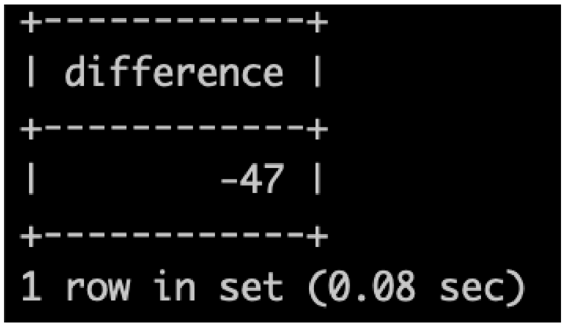
```
+----+-------------+-------+--------+---------------+---------+---------+------------------------+------+----------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref                    | rows | Extra          |
+----+-------------+-------+--------+---------------+---------+---------+------------------------+------+----------------+
|  1 | PRIMARY     | NULL  | NULL   | NULL          | NULL    | NULL    | NULL                   | NULL | No tables used |
|  3 | SUBQUERY    | H     | ALL    | NULL          | NULL    | NULL    | NULL                   | 4136 | NULL           |
|  3 | SUBQUERY    | M     | eq_ref | PRIMARY       | PRIMARY | 767     | db356_l7jing.H.playerID |    1 | Using where    |
|  2 | SUBQUERY    | H     | ALL    | NULL          | NULL    | NULL    | NULL                   | 4136 | NULL           |
|  2 | SUBQUERY    | M     | eq_ref | PRIMARY       | PRIMARY | 767     | db356_l7jing.H.playerID |    1 | Using where    |
+----+-------------+-------+--------+---------------+---------+---------+------------------------+------+----------------+
5 rows in set (0.01 sec)
```

Similar to (a), since we are going to search all the player with or without the specific birthdate. So eventually all entries will be accessed at least once in the master table, so we try to adding indexes for birthdate and playerID to compare their performances.

```
CREATE INDEX Master_1 on Master (birthDay);
CREATE INDEX Master_2 on Master (birthMonth);
CREATE INDEX Master_3 on Master (birthYear);
CREATE INDEX Master_4 on Master (playerID);
```

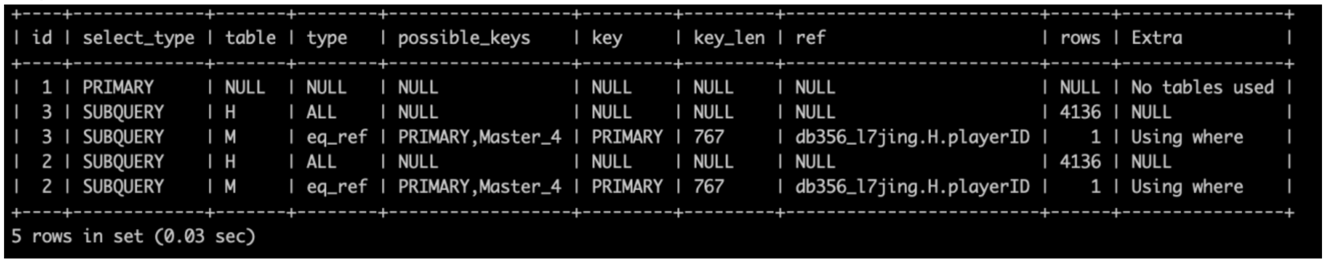
After adding indexes:

Running time:



```
+------------+
| difference |
+------------+
|        -47 |
+------------+
1 row in set (0.08 sec)
```

Explain:



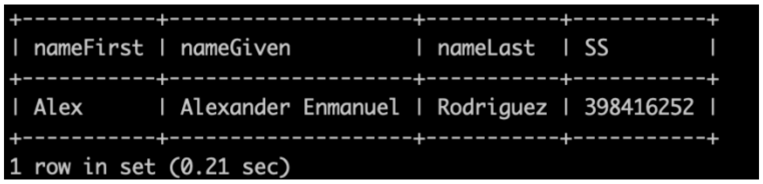
```
+----+-------------+-------+--------+------------------+---------+---------+------------------------+------+----------------+
| id | select_type | table | type   | possible_keys    | key     | key_len | ref                    | rows | Extra          |
+----+-------------+-------+--------+------------------+---------+---------+------------------------+------+----------------+
|  1 | PRIMARY     | NULL  | NULL   | NULL             | NULL    | NULL    | NULL                   | NULL | No tables used |
|  3 | SUBQUERY    | H     | ALL    | NULL             | NULL    | NULL    | NULL                   | 4136 | NULL           |
|  3 | SUBQUERY    | M     | eq_ref | PRIMARY,Master_4 | PRIMARY | 767     | db356_l7jing.H.playerID |    1 | Using where    |
|  2 | SUBQUERY    | H     | ALL    | NULL             | NULL    | NULL    | NULL                   | 4136 | NULL           |
|  2 | SUBQUERY    | M     | eq_ref | PRIMARY,Master_4 | PRIMARY | 767     | db356_l7jing.H.playerID |    1 | Using where    |
+----+-------------+-------+--------+------------------+---------+---------+------------------------+------+----------------+
5 rows in set (0.03 sec)
```

There is no obvious rows reduce after indexing, since the primary key has already provided a index table in Master table. Therefore, this table do not need other indexing tables.

(c) What is the name and total pay of the player with the largest total salary?

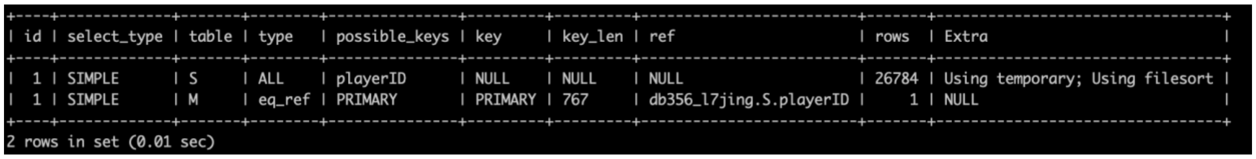
```
EXPLAIN SELECT M.nameFirst, M.nameGiven, M.nameLast,
SUM(S.salary) as SS FROM Salaries AS S LEFT OUTER JOIN Master AS
M
USING (playerID)
GROUP BY playerID
ORDER BY SS DESC
LIMIT 1;
```

Before adding indexes:
Running time:



```
+-----------+-------------------+-----------+-----------+
| nameFirst | nameGiven         | nameLast  | SS        |
+-----------+-------------------+-----------+-----------+
| Alex      | Alexander Enmanuel | Rodriguez | 398416252 |
+-----------+-------------------+-----------+-----------+
1 row in set (0.21 sec)
```

Explain:

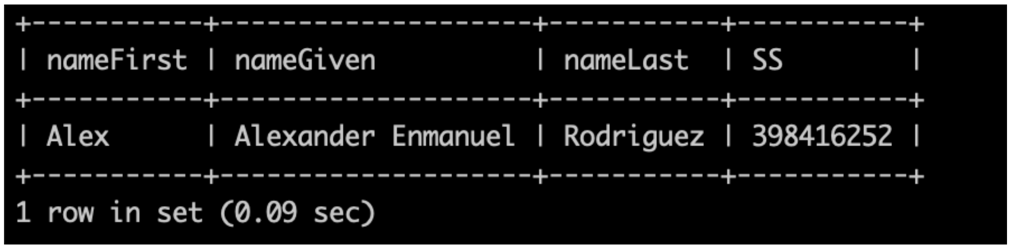


```
+----+-------------+-------+--------+---------------+---------+---------+-----------------------+-------+---------------------------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref                   | rows  | Extra                           |
+----+-------------+-------+--------+---------------+---------+---------+-----------------------+-------+---------------------------------+
|  1 | SIMPLE      | S     | ALL    | playerID      | NULL    | NULL    | NULL                  | 26784 | Using temporary; Using filesort |
|  1 | SIMPLE      | M     | eq_ref | PRIMARY       | PRIMARY | 767     | db356_l7jing.S.playerID |     1 | NULL                            |
+----+-------------+-------+--------+---------------+---------+---------+-----------------------+-------+---------------------------------+
2 rows in set (0.01 sec)
```

Since we are using “Salaries” left outer join “Master” and group by ‘playerID’, we can add index playerID in “Salaries” table first, since playerID will have duplications in “Salaries” table.

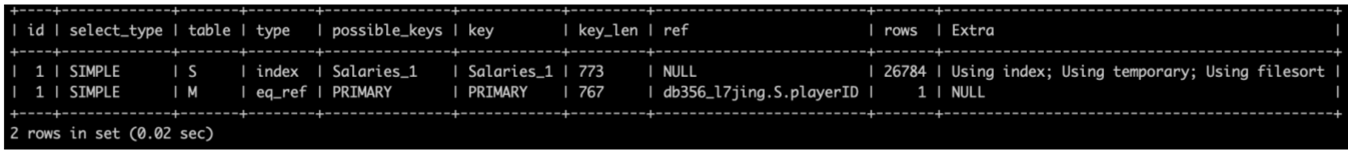
```
CREATE INDEX Salaries_1 on Salaries (playerID);
```

After adding indexes:
Running time:



```
+-----------+-------------------+-----------+-----------+
| nameFirst | nameGiven         | nameLast  | SS        |
+-----------+-------------------+-----------+-----------+
| Alex      | Alexander Enmanuel | Rodriguez | 398416252 |
+-----------+-------------------+-----------+-----------+
1 row in set (0.09 sec)
```

Explain:



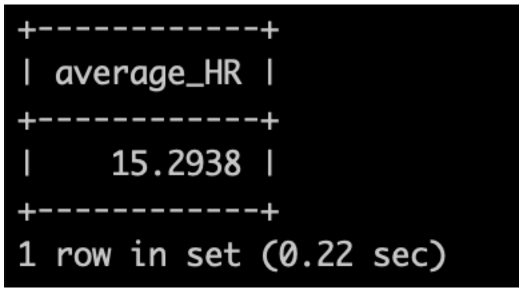
```
+----+-------------+-------+--------+---------------+-----------+---------+------------------------+-------+------------------------------------------+
| id | select_type | table | type   | possible_keys | key       | key_len | ref                    | rows  | Extra                                    |
+----+-------------+-------+--------+---------------+-----------+---------+------------------------+-------+------------------------------------------+
|  1 | SIMPLE      | S     | index  | Salaries_1    | Salaries_1 | 773    | NULL                   | 26784 | Using index; Using temporary; Using filesort |
|  1 | SIMPLE      | M     | eq_ref | PRIMARY       | PRIMARY   | 767     | db356_l7jing.S.playerID |     1 | NULL                                     |
+----+-------------+-------+--------+---------------+-----------+---------+------------------------+-------+------------------------------------------+
2 rows in set (0.02 sec)
```

We see the performance is actually getting better, though the rows do not change after adding the indexes.

(d) What is the average number of Home Runs a player has?

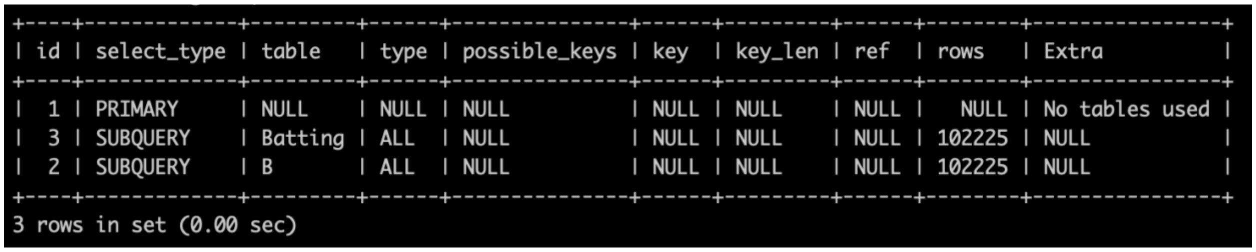
```
EXPLAIN
SELECT
(SELECT SUM(HR) FROM Batting as B)
/
(SELECT COUNT(DISTINCT playerID) FROM Batting)
AS average_HR;
```

Before adding indexes:
Running time:



```
+------------+
| average_HR |
+------------+
|    15.2938 |
+------------+
1 row in set (0.22 sec)
```

Explain:

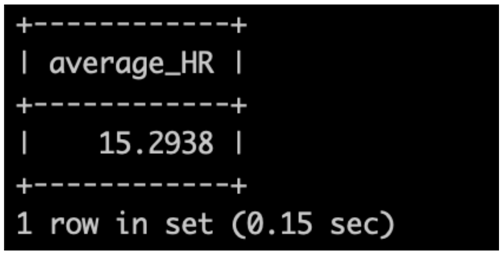


```
+----+-------------+---------+------+---------------+------+---------+------+--------+----------------+
| id | select_type | table   | type | possible_keys | key  | key_len | ref  | rows   | Extra          |
+----+-------------+---------+------+---------------+------+---------+------+--------+----------------+
|  1 | PRIMARY     | NULL    | NULL | NULL          | NULL | NULL    | NULL |   NULL | No tables used |
|  3 | SUBQUERY    | Batting | ALL  | NULL          | NULL | NULL    | NULL | 102225 | NULL           |
|  2 | SUBQUERY    | B       | ALL  | NULL          | NULL | NULL    | NULL | 102225 | NULL           |
+----+-------------+---------+------+---------------+------+---------+------+--------+----------------+
3 rows in set (0.00 sec)
```

In this case, we need to go through all entries in Batting to sum up HR anyways, but we only need distinct playerID, so we may add index on playerIDs in Batting table. For SUM() function, it is a aggregation function so indexing will not help to optimize this.

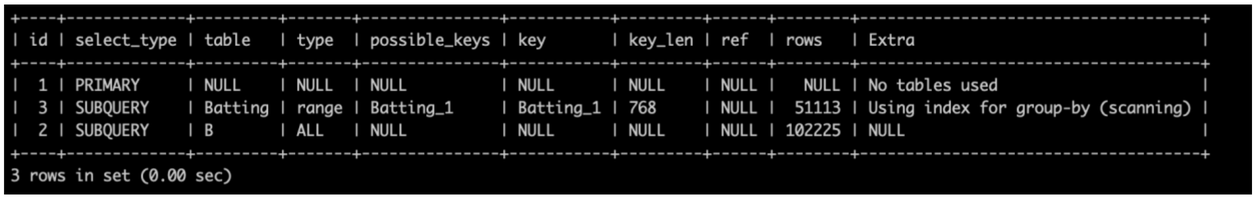
```
CREATE INDEX Batting_1 on Batting (playerID);
```

After adding index:
Running time:



```
+------------+
| average_HR |
+------------+
|    15.2938 |
+------------+
1 row in set (0.15 sec)
```

Explain:



```
+----+-------------+---------+-------+---------------+-----------+---------+------+--------+-----------------------------------+
| id | select_type | table   | type  | possible_keys | key       | key_len | ref  | rows   | Extra                             |
+----+-------------+---------+-------+---------------+-----------+---------+------+--------+-----------------------------------+
|  1 | PRIMARY     | NULL    | NULL  | NULL          | NULL      | NULL    | NULL |   NULL | No tables used                    |
|  3 | SUBQUERY    | Batting | range | Batting_1     | Batting_1 | 768     | NULL |  51113 | Using index for group-by (scanning) |
|  2 | SUBQUERY    | B       | ALL   | NULL          | NULL      | NULL    | NULL | 102225 | NULL                              |
+----+-------------+---------+-------+---------------+-----------+---------+------+--------+-----------------------------------+
3 rows in set (0.00 sec)
```

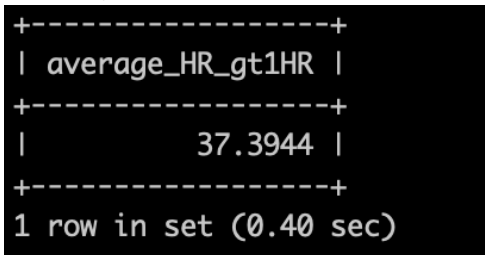
As we can see, adding index may not help to improve the aggregation queries, but can optimize the searching queries for indexings.

(e) If we only count players who got at least 1 Home Run, what is the average number of Home Runs a player has?

```
EXPLAIN
SELECT
(SELECT SUM(HR) FROM Batting as B)
/
(SELECT COUNT(DISTINCT T.playerID) FROM
(SELECT playerID FROM Batting as B
GROUP BY playerID
HAVING Sum(B.HR) > 0) AS T) as average_HR_gt1HR;
```

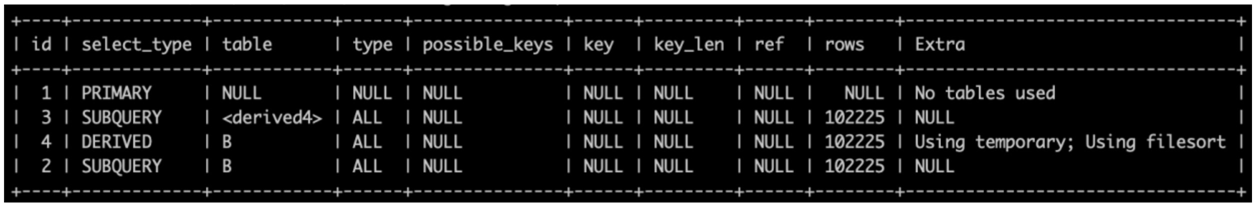
Before adding indexes:

Running time:



```
+------------------+
| average_HR_gt1HR |
+------------------+
|          37.3944 |
+------------------+
1 row in set (0.40 sec)
```

Explain:

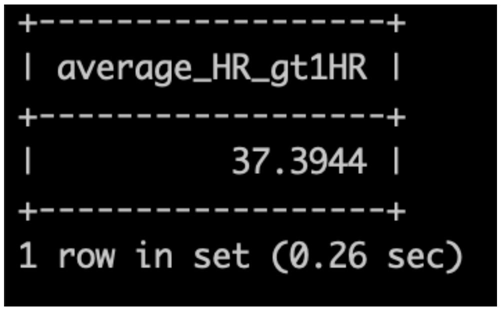


```
+----+-------------+------------+------+---------------+------+---------+------+--------+------------------------------+
| id | select_type | table      | type | possible_keys | key  | key_len | ref  | rows   | Extra                        |
+----+-------------+------------+------+---------------+------+---------+------+--------+------------------------------+
|  1 | PRIMARY     | NULL       | NULL | NULL          | NULL | NULL    | NULL |   NULL | No tables used               |
|  3 | SUBQUERY    | <derived4> | ALL  | NULL          | NULL | NULL    | NULL | 102225 | NULL                         |
|  4 | DERIVED     | B          | ALL  | NULL          | NULL | NULL    | NULL | 102225 | Using temporary; Using filesort |
|  2 | SUBQUERY    | B          | ALL  | NULL          | NULL | NULL    | NULL | 102225 | NULL                         |
+----+-------------+------------+------+---------------+------+---------+------+--------+------------------------------+
```

In this case, we still need to go through all entries in Batting at least once to sum up the HR, but for the players, we only consider the distinct player who at least has one HR. so we can create an index for playerID, and also an index for HR

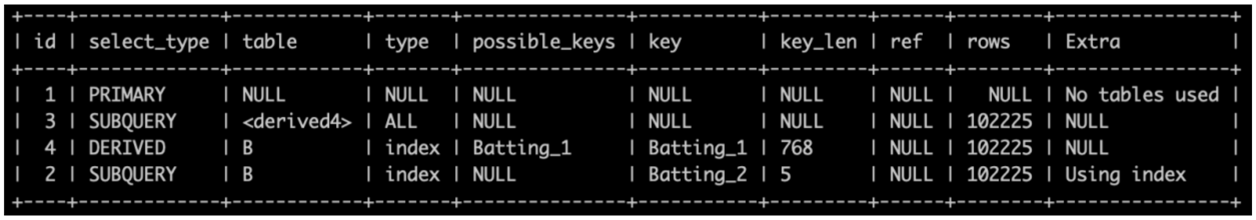
```
CREATE INDEX Batting_3 on Batting (playerID, HR);
```

After adding index:
Running time:



```
+-----------------+
| average_HR_gt1HR |
+-----------------+
|         37.3944 |
+-----------------+
1 row in set (0.26 sec)
```

Explain:



```
+----+-------------+-----------+-------+---------------+-----------+---------+------+--------+----------------+
| id | select_type | table     | type  | possible_keys | key       | key_len | ref  | rows   | Extra          |
+----+-------------+-----------+-------+---------------+-----------+---------+------+--------+----------------+
|  1 | PRIMARY     | NULL      | NULL  | NULL          | NULL      | NULL    | NULL |   NULL | No tables used |
|  3 | SUBQUERY    | <derived4> | ALL  | NULL          | NULL      | NULL    | NULL | 102225 | NULL           |
|  4 | DERIVED     | B         | index | Batting_1     | Batting_1 | 768     | NULL | 102225 | NULL           |
|  2 | SUBQUERY    | B         | index | NULL          | Batting_2 | 5       | NULL | 102225 | Using index    |
+----+-------------+-----------+-------+---------------+-----------+---------+------+--------+----------------+
```

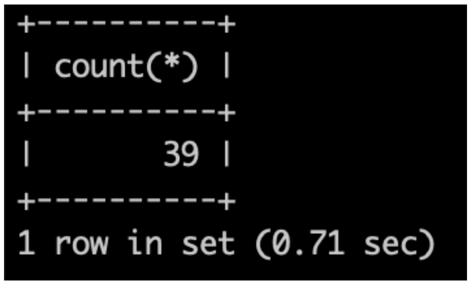
After using index, there is no changes on rows but there are some keys were used, and in that case the performance is getting better.

(f) If we define a player as a good batter if they have more than the average number of Home Runs, and a player is a good Pitcher if they have more than the average number of ShutOut games, then how many players are both good batters and good pitchers?

```
EXPLAIN
SELECT count(*) FROM

(SELECT * FROM
(SELECT playerID, SUM(HR) as sumHR FROM Batting as B
GROUP BY playerID) as T
WHERE T.sumHR > (
SELECT
(SELECT SUM(HR) FROM Batting as B)
/
(SELECT COUNT(DISTINCT playerID) FROM Batting)
AS average_HR
)) AS T_goodbatter
```

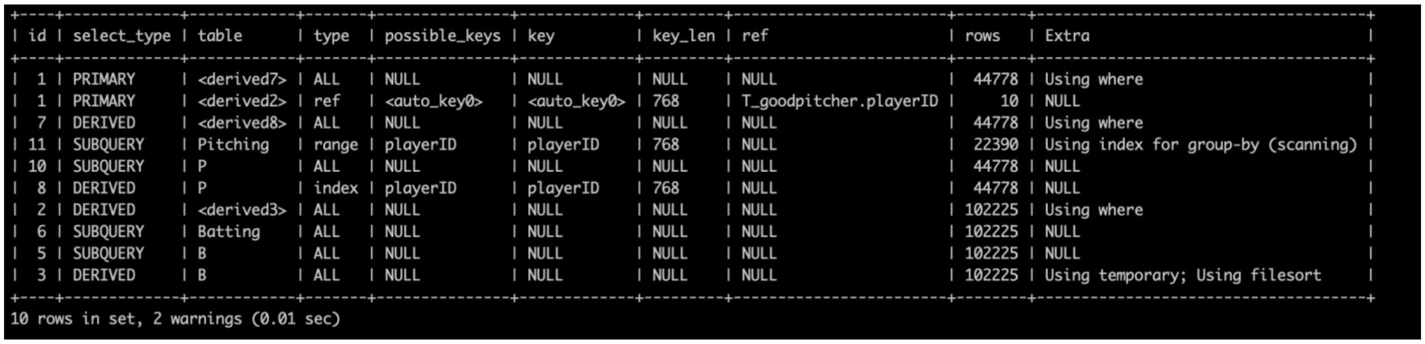
```
INNER join

(SELECT * FROM
(SELECT playerID, SUM(SHO) as sumSHO FROM Pitching as P
GROUP BY playerID) as T
WHERE T.sumSHO > (
SELECT
(SELECT SUM(SHO) FROM Pitching as P)
/
(SELECT COUNT(DISTINCT playerID) FROM Pitching)
AS average_SHO
)) AS T_goodpitcher

using (playerID);
```

Before adding indexes:
Running time:



```
+----------+
| count(*) |
+----------+
|       39 |
+----------+
1 row in set (0.71 sec)
```

Explain:



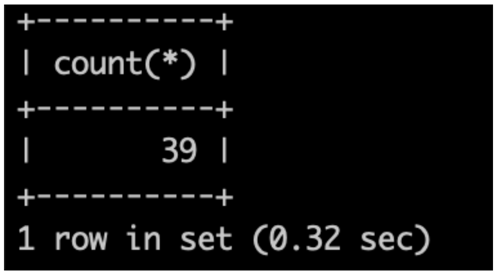
```
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
| id | select_type | table      | type  | possible_keys | key        | key_len | ref                   | rows   | Extra                             |
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
|  1 | PRIMARY     | <derived7> | ALL   | NULL          | NULL       | NULL    | NULL                  |  44778 | Using where                       |
|  1 | PRIMARY     | <derived2> | ref   | <auto_key0>   | <auto_key0> | 768    | T_goodpitcher.playerID |     10 | NULL                             |
|  7 | DERIVED     | <derived8> | ALL   | NULL          | NULL       | NULL    | NULL                  |  44778 | Using where                       |
| 11 | SUBQUERY    | Pitching   | range | playerID      | playerID   | 768     | NULL                  |  22390 | Using index for group-by (scanning) |
| 10 | SUBQUERY    | P          | ALL   | NULL          | NULL       | NULL    | NULL                  |  44778 | NULL                              |
|  8 | DERIVED     | P          | index | playerID      | playerID   | 768     | NULL                  |  44778 | NULL                              |
|  2 | DERIVED     | <derived3> | ALL   | NULL          | NULL       | NULL    | NULL                  | 102225 | Using where                       |
|  6 | SUBQUERY    | Batting    | ALL   | NULL          | NULL       | NULL    | NULL                  | 102225 | NULL                              |
|  5 | SUBQUERY    | B          | ALL   | NULL          | NULL       | NULL    | NULL                  | 102225 | NULL                              |
|  3 | DERIVED     | B          | ALL   | NULL          | NULL       | NULL    | NULL                  | 102225 | Using temporary; Using filesort   |
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
10 rows in set, 2 warnings (0.01 sec)
```

The attribute playerID for Batting and Pitching are obvious indexes as they are respectively grouped in their subqueries and subsequently joined using this attribute, which indexing can help improve in efficiency. The HR and SHO attributes are also indexed as a precaution in case it can assist in the predicate for comparing sums of HRs and SHOs to be higher than a certain amount (in this case, the average), which indexing may also help in.

```
CREATE INDEX Batting_1 on Batting (playerID, HR);
CREATE INDEX Batting_2 on Batting (HR);
CREATE INDEX Pitching_1 on Pitching (playerID, SHO);
CREATE INDEX Pitching_2 on Pitching (SHO);
```

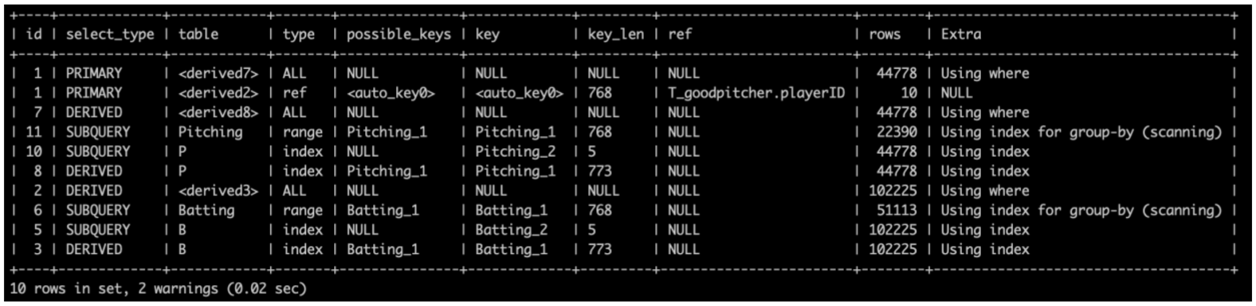
After adding indexes:

Running time:



```
+----------+
| count(*) |
+----------+
|       39 |
+----------+
1 row in set (0.32 sec)
```

Explain:



```
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
| id | select_type | table      | type  | possible_keys | key        | key_len | ref                   | rows   | Extra                             |
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
|  1 | PRIMARY     | <derived7> | ALL   | NULL          | NULL       | NULL    | NULL                  |  44778 | Using where                       |
|  1 | PRIMARY     | <derived2> | ref   | <auto_key0>   | <auto_key0> | 768    | T_goodpitcher.playerID |     10 | NULL                              |
|  7 | DERIVED     | <derived8> | ALL   | NULL          | NULL       | NULL    | NULL                  |  44778 | Using where                       |
| 11 | SUBQUERY    | Pitching   | range | Pitching_1    | Pitching_1 | 768     | NULL                  |  22390 | Using index for group-by (scanning) |
| 10 | SUBQUERY    | P          | index | NULL          | Pitching_2 | 5       | NULL                  |  44778 | Using index                       |
|  8 | DERIVED     | P          | index | Pitching_1    | Pitching_1 | 773     | NULL                  |  44778 | Using index                       |
|  2 | DERIVED     | <derived3> | ALL   | NULL          | NULL       | NULL    | NULL                  | 102225 | Using where                       |
|  6 | SUBQUERY    | Batting    | range | Batting_1     | Batting_1  | 768     | NULL                  |  51113 | Using index for group-by (scanning) |
|  5 | SUBQUERY    | B          | index | NULL          | Batting_2  | 5       | NULL                  | 102225 | Using index                       |
|  3 | DERIVED     | B          | index | Batting_1     | Batting_1  | 773     | NULL                  | 102225 | Using index                       |
+----+-------------+------------+-------+---------------+------------+---------+-----------------------+--------+-----------------------------------+
10 rows in set, 2 warnings (0.02 sec)
```