# ECE 356 Winter 2019: Lab2 Part2

## Baseball

The general approach of how we add indexes and determine whether if those indexes are helpful in solving those problems:

1) Look at the output from the keyword “explain”, determine whether if the query involves table scan.
2) Look at the query, if it has the keywords: where, join on, order by, having, group by. The columns that involve those keywords are very likely to be the index.
3) After adding indexes, look at the output from the keyword “explain” again.
4) If the rows number reduced, the added indexes are useful for the problem solving.
5) If the rows number didn’t change, the added indexes are NOT useful for the problem solving.

### Question 1

```
EXPLAIN SELECT count(DISTINCT playerID) AS player_missing_birthday
FROM Master
WHERE birthMonth='0'
    OR birthDay='0'
    OR birthYear='0';
```

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

We can see that it is use type “all” for searching in the Master table and it is “Using where”. So clearly, birthMonth, birthday, birthYear needs index because database will go through a table scan to find out it they are “0”.

```
CREATE INDEX INDEX_birthMonth ON Master(birthMonth) USING BTREE;
CREATE INDEX INDEX_birthDay ON Master(birthDay) USING BTREE;
CREATE INDEX INDEX_birthYear ON Master(birthYear) USING BTREE;
```

```
+----+-------------+--------+-------------+-------------------------------------------------------------+----------------------------------------------------+---------+------+------+-----------------------------------------------------------------------------+
| id | select_type | table  | type        | possible_keys                                               | key                                                | key_len | ref  | rows | Extra                                                                       |
+----+-------------+--------+-------------+-------------------------------------------------------------+----------------------------------------------------+---------+------+------+-----------------------------------------------------------------------------+
|  1 | SIMPLE      | Master | index_merge | PRIMARY,INDEX_birthMonth,INDEX_birthDay,INDEX_birthYear | INDEX_birthMonth,INDEX_birthDay,INDEX_birthYear | 5,5,5   | NULL |  882 | Using union(INDEX_birthMonth,INDEX_birthDay,INDEX_birthYear); Using where |
+----+-------------+--------+-------------+-------------------------------------------------------------+----------------------------------------------------+---------+------+------+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Now, we can see that after the index adding, the number of rows scanned reduced from 19057 to 882. This means that less rows went through the table scan. This means that the added indexes are useful for the question solving as it improves the performance. So our method for determine the indexes are correct.

## Question 2

```
EXPLAIN SELECT
  (SELECT count(*) AS alive
   FROM
   (SELECT DISTINCT HallOfFame.playerID,
                    deathCountry
    FROM HallOfFame
    INNER JOIN Master ON HallOfFame.playerID = Master.playerID) AS temp1
  WHERE deathCountry = '') -
  (SELECT count(*) AS dead
   FROM
   (SELECT DISTINCT HallOfFame.playerID,
                    deathCountry
    FROM HallOfFame
    INNER JOIN Master ON HallOfFame.playerID = Master.playerID) AS temp2
  WHERE deathCountry != '') AS alive_minus_dead;
```

```
+----+-------------+------------+--------+---------------+------------+---------+-----------------------------------+------+------------------------------+
| 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               |
|  4 | SUBQUERY    | <derived5> | ALL    | NULL          | NULL       | NULL    | NULL                              | 4156 | Using where                  |
|  5 | DERIVED     | HallOfFame | index  | PRIMARY       | PRIMARY    | 1538    | NULL                              | 4156 | Using index; Using temporary |
|  5 | DERIVED     | Master     | eq_ref | PRIMARY       | PRIMARY    | 767     | db356_jm3zhang.HallOfFame.playerID |    1 | Distinct                     |
|  2 | SUBQUERY    | <derived3> | ref    | <auto_key0>   | <auto_key0> | 768    | const                             |   10 | Using where                  |
|  3 | DERIVED     | HallOfFame | index  | PRIMARY       | PRIMARY    | 1538    | NULL                              | 4156 | Using index; Using temporary |
|  3 | DERIVED     | Master     | eq_ref | PRIMARY       | PRIMARY    | 767     | db356_jm3zhang.HallOfFame.playerID |    1 | Distinct                     |
+----+-------------+------------+--------+---------------+------------+---------+-----------------------------------+------+------------------------------+
7 rows in set (0.01 sec)
```

We can see that, two of the scans are already using indexes to scan. The first line of the input said that the table name is null. We will ignore that since it is scanning nowhere. The derived tables cannot be indexed so they will also be ignored for indexing. The two scan from the Master is under type eq_ref, which is good since they will straight reference to Master. There are 2 “Using where” for “deathCountry”, so they need to be indexed. Also there are 2 join on “HallOfFame.playerID”, so they need to be indexed.

```
CREATE INDEX INDEX_deathCountry ON HallOfFame(deathCountry) USING BTREE;
CREATE INDEX INDEX_HOFplayerID ON HallOfFame(playerID) USING BTREE;
```

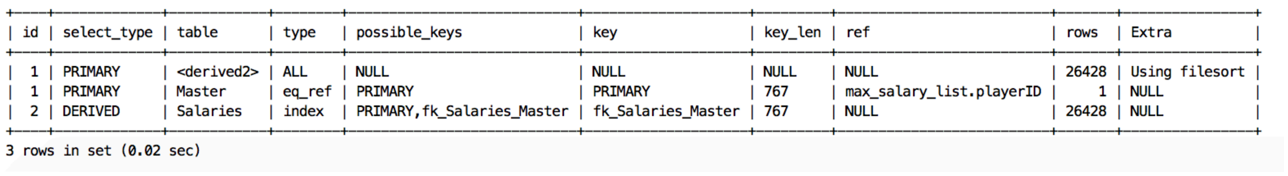
```
+----+-------------+------------+--------+---------------------------+-------------------+---------+-----------------------------------+------+------------------------------+
| id | select_type | table      | type   | possible_keys             | key               | key_len | ref                               | rows | Extra                        |
+----+-------------+------------+--------+---------------------------+-------------------+---------+-----------------------------------+------+------------------------------+
|  1 | PRIMARY     | NULL       | NULL   | NULL                      | NULL              | NULL    | NULL                              | NULL | No tables used               |
|  4 | SUBQUERY    | <derived5> | ALL    | NULL                      | NULL              | NULL    | NULL                              | 4156 | Using where                  |
|  5 | DERIVED     | HallOfFame | index  | PRIMARY,INDEX_HOFplayerID | INDEX_HOFplayerID | 767     | NULL                              | 4156 | Using index; Using temporary |
|  5 | DERIVED     | Master     | eq_ref | PRIMARY                   | PRIMARY           | 767     | db356_jm3zhang.HallOfFame.playerID |    1 | Distinct                     |
|  2 | SUBQUERY    | <derived3> | ref    | <auto_key0>               | <auto_key0>       | 768     | const                             |   10 | Using where                  |
|  3 | DERIVED     | HallOfFame | index  | PRIMARY,INDEX_HOFplayerID | INDEX_HOFplayerID | 767     | NULL                              | 4156 | Using index; Using temporary |
|  3 | DERIVED     | Master     | eq_ref | PRIMARY                   | PRIMARY           | 767     | db356_jm3zhang.HallOfFame.playerID |    1 | Distinct                     |
+----+-------------+------------+--------+---------------------------+-------------------+---------+-----------------------------------+------+------------------------------+
7 rows in set (0.01 sec)
```

Now, we can see that after the index adding, the key length reduced from 1538 to 767. This can show that the add indexes provides a little bit improvement on the problem solving.

## Question 3

```
EXPLAIN SELECT nameFirst,
       nameLast,
       nameGiven,
       salary_sum AS largest_total_salary
FROM Master
INNER JOIN
    (SELECT playerID,
            SUM(salary) AS salary_sum
     FROM Salaries
     GROUP BY playerID) AS max_salary_list ON max_salary_list.playerID =
Master.playerID
ORDER BY salary_sum DESC
LIMIT 1;
```

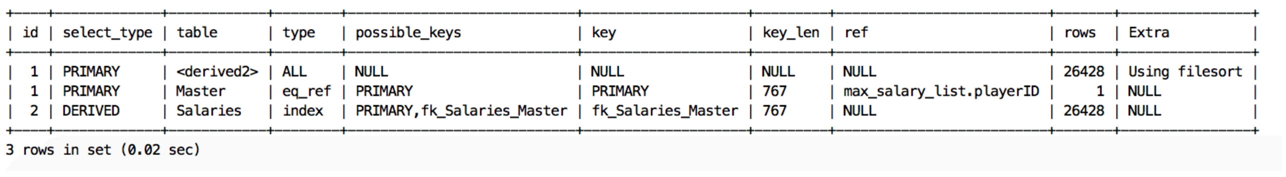
Output from keyword "explain":



```
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
| id | select_type | table      | type   | possible_keys               | key               | key_len | ref                      | rows  | Extra          |
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
|  1 | PRIMARY     | <derived2> | ALL    | NULL                        | NULL              | NULL    | NULL                     | 26428 | Using filesort |
|  1 | PRIMARY     | Master     | eq_ref | PRIMARY                     | PRIMARY           | 767     | max_salary_list.playerID |     1 | NULL           |
|  2 | DERIVED     | Salaries   | index  | PRIMARY,fk_Salaries_Master  | fk_Salaries_Master | 767    | NULL                     | 26428 | NULL           |
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
3 rows in set (0.02 sec)
```

We can see that the second and third line of output is scanning with the type eq_ref and index, so they are already good. The first row is using derived table so it cannot be indexed. Indexing won't help for this question.

```
CREATE INDEX INDEX_salary ON Salaries(salary) USING BTREE;
CREATE INDEX INDEX_Salariespid ON Salaries(playerID) USING BTREE;
```

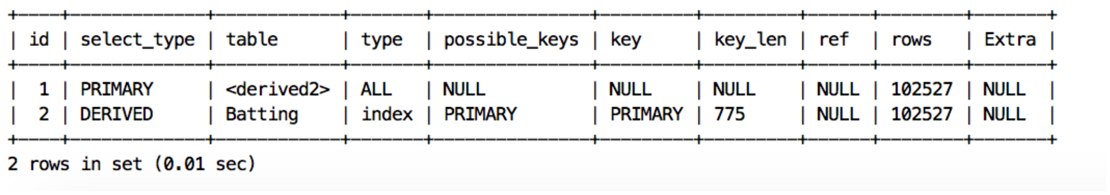


```
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
| id | select_type | table      | type   | possible_keys               | key               | key_len | ref                      | rows  | Extra          |
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
|  1 | PRIMARY     | <derived2> | ALL    | NULL                        | NULL              | NULL    | NULL                     | 26428 | Using filesort |
|  1 | PRIMARY     | Master     | eq_ref | PRIMARY                     | PRIMARY           | 767     | max_salary_list.playerID |     1 | NULL           |
|  2 | DERIVED     | Salaries   | index  | PRIMARY,fk_Salaries_Master  | fk_Salaries_Master | 767    | NULL                     | 26428 | NULL           |
+----+-------------+------------+--------+-----------------------------+-------------------+---------+--------------------------+-------+----------------+
3 rows in set (0.02 sec)
```

This show that the indexing is NOT useful for this problem solving. Since there is a "SUM(salary)" (Aggregation), the whole table will be scanned no matter what, so index will NOT help in this case.

## Question 4

```
EXPLAIN SELECT AVG(individual_hr_sum) AS average_hr
FROM
  (SELECT playerID,
          Sum(HR) AS individual_hr_sum
   FROM Batting
   GROUP BY playerID) AS sum_list;
```

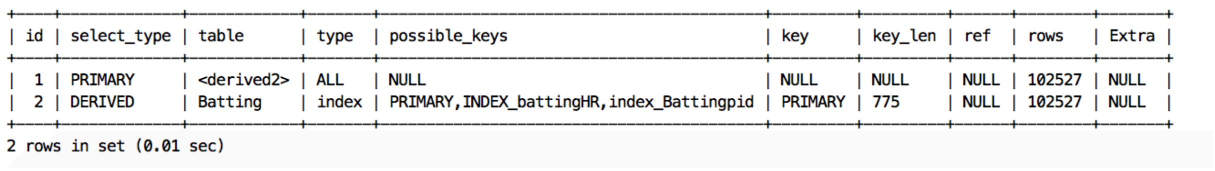


```
+----+-------------+------------+-------+---------------+---------+---------+------+--------+-------+
| id | select_type | table      | type  | possible_keys | key     | key_len | ref  | rows   | Extra |
+----+-------------+------------+-------+---------------+---------+---------+------+--------+-------+
|  1 | PRIMARY     | <derived2> | ALL   | NULL          | NULL    | NULL    | NULL | 102527 | NULL  |
|  2 | DERIVED     | Batting    | index | PRIMARY       | PRIMARY | 775     | NULL | 102527 | NULL  |
+----+-------------+------------+-------+---------------+---------+---------+------+--------+-------+
2 rows in set (0.01 sec)
```

Again, derived table cannot be indexed and the second one is already indexed, so no indexing is needed for them. Form the query, since there is a "Sum(HR)" (Aggregation), the whole table will be scanned no matter what, so index will NOT help in this case.

## Question 5

```
EXPLAIN SELECT AVG(individual_hr_sum_without_zero) AS
average_hr_without_zero
FROM
  (SELECT playerID,
          Sum(HR) AS individual_hr_sum_without_zero
   FROM Batting
   GROUP BY playerID
   HAVING avg(HR) > 0) AS sum_list;
```

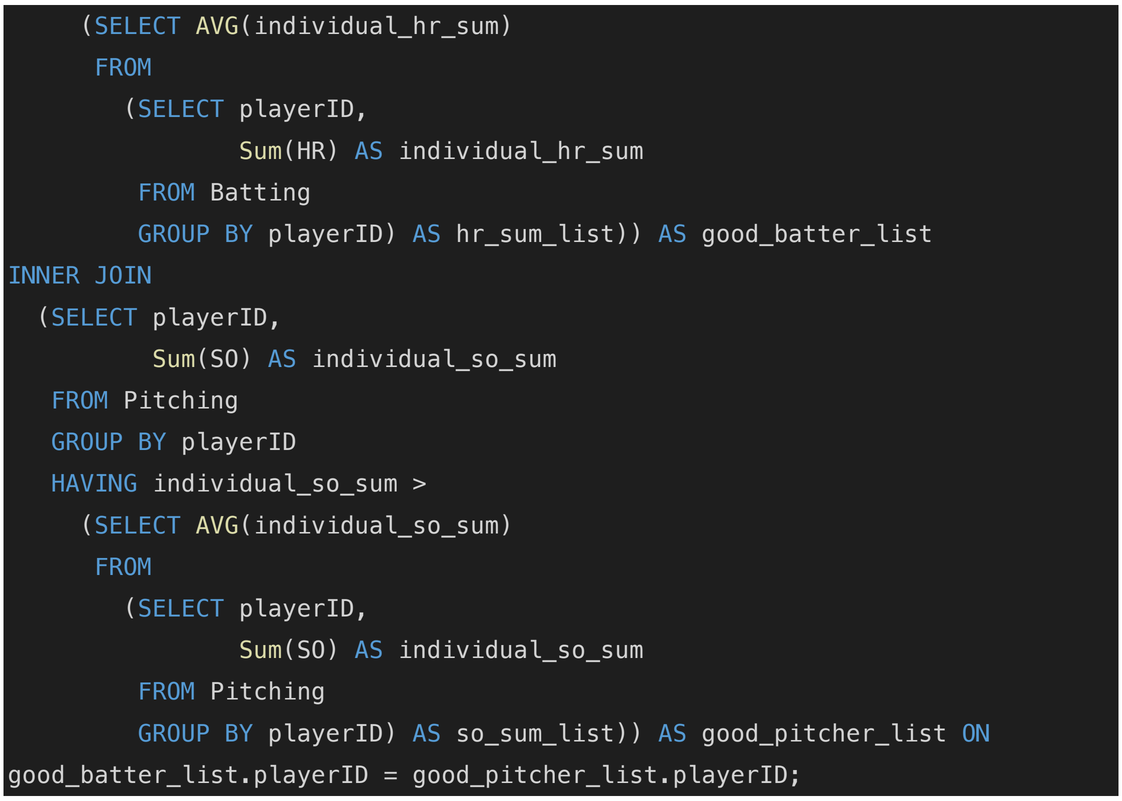


```
+----+-------------+------------+-------+----------------------------------------------+---------+---------+------+--------+-------+
| id | select_type | table      | type  | possible_keys                                | key     | key_len | ref  | rows   | Extra |
+----+-------------+------------+-------+----------------------------------------------+---------+---------+------+--------+-------+
|  1 | PRIMARY     | <derived2> | ALL   | NULL                                         | NULL    | NULL    | NULL | 102527 | NULL  |
|  2 | DERIVED     | Batting    | index | PRIMARY,INDEX_battingHR,index_Battingpid     | PRIMARY | 775     | NULL | 102527 | NULL  |
+----+-------------+------------+-------+----------------------------------------------+---------+---------+------+--------+-------+
2 rows in set (0.01 sec)
```

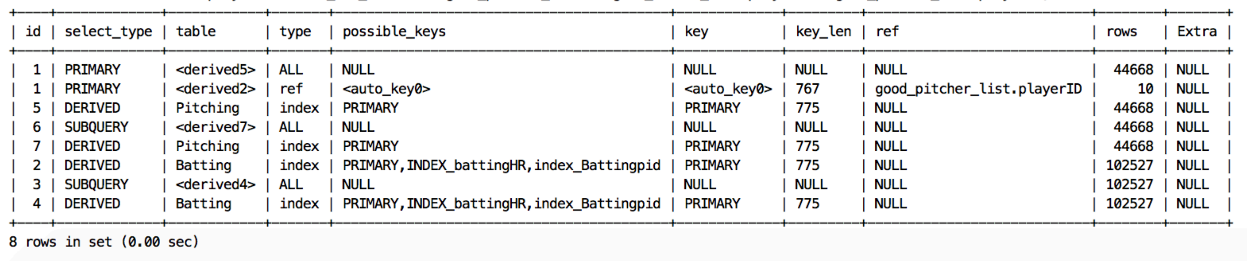
Again, derived table cannot be indexed and the second one is already indexed, so no indexing is needed for them. Form the query, since there is a "Sum(HR)" (Aggregation), the whole table will be scanned no matter what, so index will NOT help in this case.

## Question 6

```
EXPLAIN SELECT count(*) AS good_player
FROM
  (SELECT playerID,
          Sum(HR) AS individual_hr_sum
   FROM Batting
   GROUP BY playerID
   HAVING individual_hr_sum >
```



```
      (SELECT AVG(individual_hr_sum)
       FROM
         (SELECT playerID,
                 Sum(HR) AS individual_hr_sum
          FROM Batting
          GROUP BY playerID) AS hr_sum_list)) AS good_batter_list
INNER JOIN
  (SELECT playerID,
          Sum(SO) AS individual_so_sum
   FROM Pitching
   GROUP BY playerID
   HAVING individual_so_sum >
     (SELECT AVG(individual_so_sum)
      FROM
        (SELECT playerID,
                Sum(SO) AS individual_so_sum
         FROM Pitching
         GROUP BY playerID) AS so_sum_list)) AS good_pitcher_list ON
good_batter_list.playerID = good_pitcher_list.playerID;
```



```
+----+-------------+------------+-------+----------------------------------------+-------------+---------+----------------------------+--------+-------+
| id | select_type | table      | type  | possible_keys                          | key         | key_len | ref                        | rows   | Extra |
+----+-------------+------------+-------+----------------------------------------+-------------+---------+----------------------------+--------+-------+
|  1 | PRIMARY     | <derived5> | ALL   | NULL                                   | NULL        | NULL    | NULL                       |  44668 | NULL  |
|  1 | PRIMARY     | <derived2> | ref   | <auto_key0>                            | <auto_key0> | 767     | good_pitcher_list.playerID |     10 | NULL  |
|  5 | DERIVED     | Pitching   | index | PRIMARY                                | PRIMARY     | 775     | NULL                       |  44668 | NULL  |
|  6 | SUBQUERY    | <derived7> | ALL   | NULL                                   | NULL        | NULL    | NULL                       |  44668 | NULL  |
|  7 | DERIVED     | Pitching   | index | PRIMARY                                | PRIMARY     | 775     | NULL                       |  44668 | NULL  |
|  2 | DERIVED     | Batting    | index | PRIMARY,INDEX_battingHR,index_Battingpid | PRIMARY   | 775     | NULL                       | 102527 | NULL  |
|  3 | SUBQUERY    | <derived4> | ALL   | NULL                                   | NULL        | NULL    | NULL                       | 102527 | NULL  |
|  4 | DERIVED     | Batting    | index | PRIMARY,INDEX_battingHR,index_Battingpid | PRIMARY   | 775     | NULL                       | 102527 | NULL  |
+----+-------------+------------+-------+----------------------------------------+-------------+---------+----------------------------+--------+-------+
8 rows in set (0.00 sec)
```

Again, derived table cannot be indexed and the third, fifth, sixth, eighth one is already indexed, so no indexing is needed for them. Form the query, since there is a "Sum(HR)", "Sum(SO)", count(*), "AVG(individual_hr_sum)" and "AVG(individual_so_sum)" (Aggregation) the whole table will be scanned no matter what, so index will NOT help in this case.