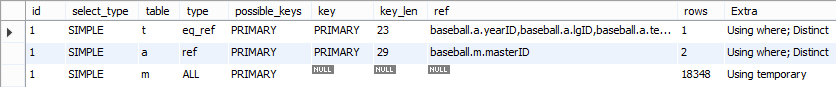
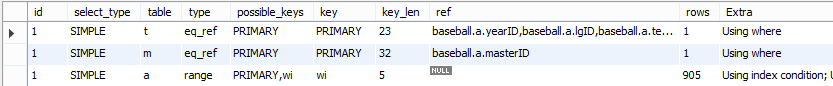
SQL Query Optimization

1)

a. The Cost of evaluating the query: Here the query is processing 18k+ rows



b. What you did to optimize the query: Created indexes for “wins” column in pitching table and reduced the number of rows.



c. Changes made to query to optimize:

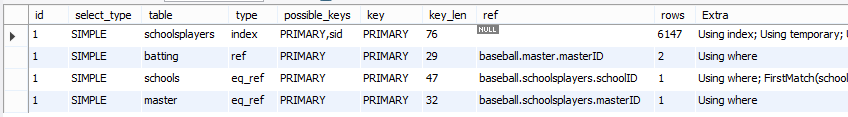
Created index using query: alter table pitching add index wi(w);

No changes are made to the query but created indexes to reduce number of rows as shown in above picture.

2)

a. The Cost of evaluating the query: the provided query is using 6000fvb c

+ rows to process.



b. What you did to optimize the query: I tried creating index for schoolName column of schools and schoolID of schoolPlayers but didn’t find any change.

c. Changes made to query to optimize: I created index using the below query but the results didn’t change.

alter table schools add index sname(schoolName)

alter table schoolsPlayers add index sid(schoolID)

I also tried using Cartesian product of two queries but found that is leading to processing of more rows.

3)

a. The Cost of evaluating the query: this is processing 97,000 rows.

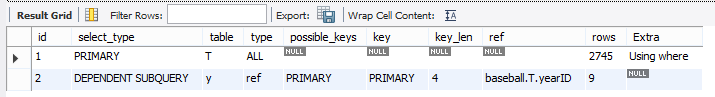
b. What you did to optimize the query: Created index for firstName and lastName of master table to optimize the query. Index created using below query:

Alter table master add index fn(nameFirst)

Alter table master add index ln(nameLast)

c. Changes made to query to optimize: no changes made to the query but after creating indices, the processing time was reduced when compared with the earlier query.

4)

a. The Cost of evaluating the query: 

b. What you did to optimize the query: I created index for yearID, wins and name as below

alter table teams add index yid(yearID)

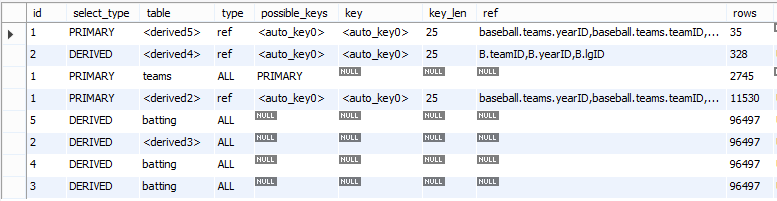
alter table teams add index wi(w)

alter table teams add index tname(name)

But didn’t find any changes in the results

c. Changes made to query to optimize: I created index and tried executing query but found the same results.

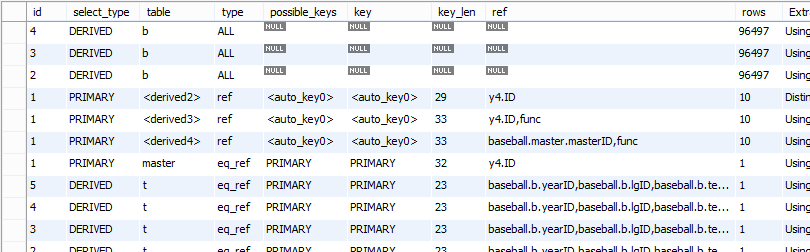
5)

a. The Cost of evaluating the query: 

b. What you did to optimize the query: I tried using join, Cartesian product and also tried using index but, I am not finding any change in the optimization.

c. Changes made to query to optimize: Tried using join and Cartesian product but no changes discovered.

6)

a. The Cost of evaluating the query: 

b. What you did to optimize the query:

c. Changes made to query to optimize:

7)

a. The Cost of evaluating the query:

b. What you did to optimize the query:

b. What you did to optimize the query: I used a different query for optimizing.

c. Changes made to query to optimize: I used below query to optimize.

SELECT DISTINCT

    M.nameFirst AS `First Name`, M.nameLast AS `Last Name`

FROM

    batting B

        INNER JOIN

    master M ON M.masterID = B.masterID

        INNER JOIN

    teams T ON B.teamID = T.teamID

        AND B.yearID = T.yearID

WHERE

    B.masterID IN (SELECT DISTINCT

            B1.masterID

        FROM

            batting B1

                INNER JOIN

            batting B2 ON B1.masterID = B2.masterID

                AND B1.yearID = B2.yearID - 1

                AND B1.teamID = B2.teamID

                INNER JOIN

            batting B3 ON B2.masterID = B3.masterID

                AND B2.yearID = B3.yearID - 1

                AND B2.teamID = B3.teamID

                INNER JOIN

            batting B4 ON B3.masterID = B4.masterID

                AND B3.yearID = B4.yearID - 1

                AND B3.teamID = B4.teamID

        WHERE

            B1.teamID IN (select teamID from teams where name LIKE '%New York Yankees%'))

        AND B.teamID IN (select teamID from teams where name LIKE '%New York Yankees%')

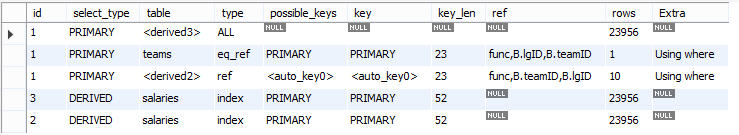
        AND T.name LIKE '%New York Yankees%'

GROUP BY B.masterID

HAVING COUNT(B.yearID) >= 4

ORDER BY M.nameLast , M.nameFirst;

7.

a. The Cost of evaluating the query: the cost resulted in almost 24000 rows

b. What you did to optimize the query: I tried using join, Cartesian product but I didn’t find any changes.

c. Changes made to query to optimize: No changes made to the query but, no results found after using join/Cartesian product.