Query Optimizations

Identify 5 queries that you think requires optimization: provide SQL, query plan and query execution time: Before & After

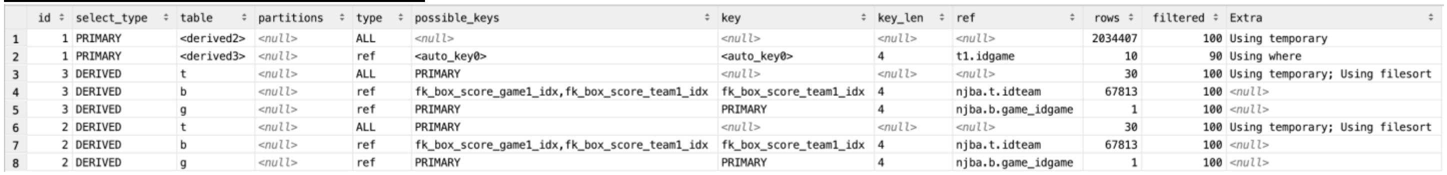
Query #1

3. As a *Fan*, I want to view standings (leaderboard) so that easily know each team's ranking

```
-- #1 Before Optimization Execution Time (Story #3 Sub-task 1)
-- ================================================================
-- 15 s 237 ms
-- 15 s 121 ms
-- 15 s 140 ms

EXPLAIN EXTENDED
(
  SELECT DISTINCT t1.idgame,
          t1.season_idseason,
          t1.home_team_id,
          t1.away_team_id,
          CASE
            WHEN t1.team_points > t2.team_points
              THEN t1.idteam
            WHEN t1.team_points < t2.team_points
              THEN t2.idteam
            END AS win_team_id,
          CASE
            WHEN t1.team_points < t2.team_points
              THEN t1.idteam
            WHEN t1.team_points > t2.team_points
              THEN t2.idteam
            END AS loss_team_id

  FROM (
       SELECT idgame,
          team_name,
          idteam,
          home_team_id,
          away_team_id,
          season_idseason,
          SUM(pts) AS team_points
```

```
        FROM game g
            JOIN box_score b
                ON g.idgame = b.game_idgame
            JOIN team t ON b.team_idteam = t.idteam
        GROUP BY 1, 2
        ORDER BY 1 ASC
      ) AS t1
        INNER JOIN
      (
        SELECT idgame,
            team_name,
            idteam,
            home_team_id,
            away_team_id,
            season_idseason,
            SUM(pts) AS team_points
        FROM game g
            JOIN box_score b
                ON g.idgame = b.game_idgame
            JOIN team t
                ON b.team_idteam = t.idteam

        GROUP BY 1, 2
        ORDER BY 1 ASC
      ) AS t2
      ON t1.idgame = t2.idgame
    WHERE t1.team_name <> t2.team_name
    HAVING win_team_id IS NOT NULL
      AND loss_team_id IS NOT NULL
);
```

optimization-query-plan1-before.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	<derived2>	<null>	ALL	<null>	<null>	<null>	<null>	2034407	100	Using temporary
2	1	PRIMARY	<derived3>	<null>	ref	<auto_key0>	<auto_key0>	4	t1.idgame	10	90	Using where
3	3	DERIVED	t	<null>	ALL	PRIMARY	<null>	<null>	<null>	30	100	Using temporary; Using filesort
4	3	DERIVED	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
5	3	DERIVED	g	<null>	ref	PRIMARY	PRIMARY	4	njba.b.game_idgame	1	100	<null>
6	2	DERIVED	t	<null>	ALL	PRIMARY	<null>	<null>	<null>	30	100	Using temporary; Using filesort
7	2	DERIVED	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
8	2	DERIVED	g	<null>	ref	PRIMARY	PRIMARY	4	njba.b.game_idgame	1	100	<null>

Optimizations

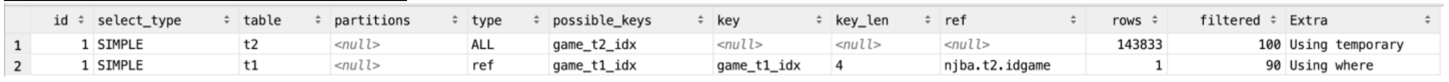
- **Avoid Subqueries**
  Since MySQL 5.7 does not have Common Table Expressions (CTE), subqueries were used but they are not optimized

well by the optimizer. Instead, to optimize this, I created a temporary table that holds the data, which also includes relevant search indices.

```
-- #1 After Optimization Results (Story #3 Sub-task 1)
-- ================================================================
-- 1 s 122 ms
-- 1 s 163 ms
-- 1 s 351 ms
DROP TABLE IF EXISTS t1_temp;

CREATE TEMPORARY TABLE t1_temp AS
(
   SELECT g.idgame,
      t.team_name,
      t.idteam,
      g.home_team_id,
      g.away_team_id,
      g.season_idseason,
      SUM(b.pts) AS team_points
    FROM game g
          JOIN
       box_score b
       ON g.idgame = b.game_idgame
          JOIN
       team t
       ON b.team_idteam = t.idteam
    GROUP BY 1,
          2
    ORDER BY 1 ASC
);

CREATE INDEX season_t1_idx
  ON t1_temp (season_idseason);

CREATE INDEX team_t1_idx
  ON t1_temp (idteam);

CREATE INDEX game_t1_idx
  ON t1_temp (idgame);
```

```
DROP TABLE IF EXISTS t2_temp;

CREATE TEMPORARY TABLE t2_temp AS
(
  SELECT
    g.idgame,
    t.team_name,
    t.idteam,
    g.home_team_id,
    g.away_team_id,
    g.season_idseason,
    SUM(b.pts) AS team_points
  FROM
    game g
  JOIN
    box_score b
      ON g.idgame = b.game_idgame
  JOIN
    team t
      ON b.team_idteam = t.idteam
  GROUP BY
    1,
    2
  ORDER BY
    1 ASC
);

CREATE INDEX season_t2_idx
  ON t2_temp (season_idseason);

CREATE INDEX team_t2_idx
  ON t2_temp (idteam);

CREATE INDEX game_t2_idx
  ON t2_temp (idgame);

EXPLAIN EXTENDED
(
  SELECT DISTINCT t1.idgame,
            t1.season_idseason,
            t1.home_team_id,
            t1.away_team_id,
            CASE
              WHEN t1.team_points > t2.team_points THEN t1.idteam
              WHEN t1.team_points < t2.team_points THEN t2.idteam
              END AS win_team_id,
            CASE
              WHEN t1.team_points < t2.team_points THEN t1.idteam
              WHEN t1.team_points > t2.team_points THEN t2.idteam
              END AS loss_team_id
    FROM t1_temp t1
         INNER JOIN
        t2_temp t2
        ON t1.idgame = t2.idgame
    WHERE t1.team_name <> t2.team_name
    HAVING win_team_id IS NOT NULL
      AND loss_team_id IS NOT NULL
);
```

optimization-query-plan1-after.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	t2	<null>	ALL	game_t2_idx	<null>	<null>	<null>	143833	100	Using temporary
2	1	SIMPLE	t1	<null>	ref	game_t1_idx	game_t1_idx	4	njba.t2.idgame	1	90	Using where

Query #2

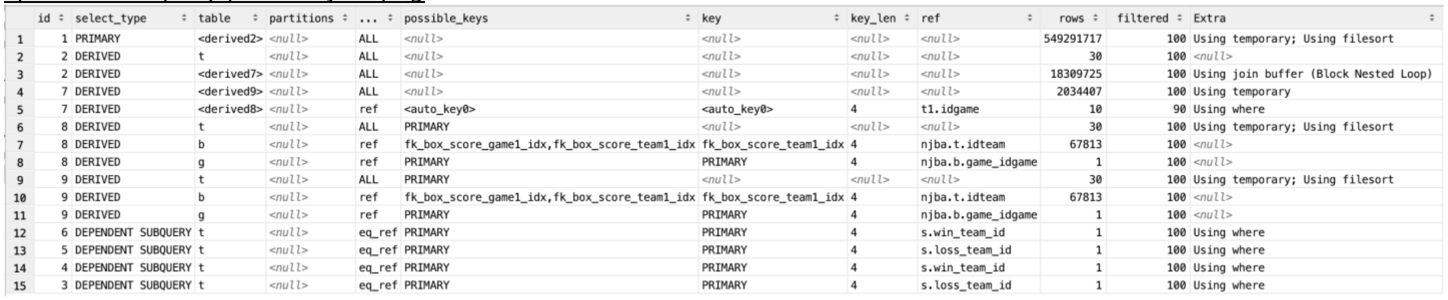
3. As a *Fan*, I want to view standings (leaderboard) so that easily know each team's ranking

```
-- #2 Before Optimization Results (Story #3 Sub-task 2)
-- ===============================================================
-- 24 s 920 ms
-- 25 s 249 ms
-- 25 s 396 ms
EXPLAIN EXTENDED
(
    SELECT season_idseason,
           idteam,
           SUM(game_win)                    AS wins,
           SUM(game_loss)                   AS loses,
           ROUND((SUM(game_win) / SUM(game_win + game_loss)) * 100, 1)
```

```
                                    AS win_pct,
        CONCAT(CAST(SUM(conf_win) AS CHAR(2)), ' - ',
            CAST(SUM(cONf_loss) AS CHAR(2))) AS conf_record,
        CONCAT(CAST(SUM(div_win) AS CHAR(2)), ' - ',
            CAST(SUM(div_loss) AS CHAR(2)))  AS div_record,
        CONCAT(CAST(SUM(home_win) AS CHAR(2)), ' - ',
            CAST(SUM(home_loss) AS CHAR(2))) AS home_record,
        CONCAT(CAST(SUM(away_win) AS CHAR(2)), ' - ',
            CAST(SUM(away_loss) AS CHAR(2))) AS away_record
    FROM (
         SELECT season_idseason,
             idteam,
             conference,
             division,
             s.win_team_id,
             s.loss_team_id,
             CASE
               WHEN s.win_team_id = t.idteam THEN 1
               ELSE 0 END AS game_win,
             CASE
               WHEN s.loss_team_id = t.idteam THEN 1
               ELSE 0 END AS game_loss,
             -- Conference Record
             CASE
               WHEN (s.win_team_id = t.idteam) AND
                  conference = (
                    SELECT t.conference
                    FROM team t
                    WHERE s.loss_team_id = t.idteam
                  )
                 THEN 1
               ELSE 0 END AS conf_win,
             CASE
               WHEN (s.loss_team_id = t.idteam) AND
                  conference = (
                    SELECT t.conference
                    FROM team t
                    WHERE s.win_team_id = t.idteam
                  )
        THEN 1
    ELSE 0 END AS cONf_loss,
-- Division Record
CASE
    WHEN (s.win_team_id = t.idteam) AND
         division = (
            SELECT t.division
            FROM team t
            WHERE s.loss_team_id = t.idteam
         )
        THEN 1
    ELSE 0 END AS div_win,
CASE
    WHEN (s.loss_team_id = t.idteam) AND
         division = (
            SELECT t.division
            FROM team t
            WHERE s.win_team_id = t.idteam
         )
        THEN 1
    ELSE 0 END AS div_loss,
-- Home Record
CASE
    WHEN (s.win_team_id = t.idteam) AND home_team_id = win_team_id
        THEN 1
    ELSE 0 END AS home_win,
CASE
    WHEN (s.loss_team_id = t.idteam) AND home_team_id = loss_team_id
        THEN 1
    ELSE 0 END AS home_loss,
-- Away Record
CASE
    WHEN (s.win_team_id = t.idteam) AND away_team_id = win_team_id
        THEN 1
    ELSE 0 END AS away_win,
CASE
    WHEN (s.loss_team_id = t.idteam) AND away_team_id = loss_team_id
        THEN 1
    ELSE 0 END AS away_loss
```

```
            FROM totals s
                    JOIN team t
        ) AS standings
    GROUP BY idteam
);
```

optimization-query-plan2-before.png



	id	select_type	table	partitions	...	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	<derived2>	<null>	ALL	<null>	<null>	<null>	<null>	549291717	100	Using temporary; Using filesort
2	2	DERIVED	t	<null>	ALL	<null>	<null>	<null>	<null>	30	100	<null>
3	2	DERIVED	<derived7>	<null>	ALL	<null>	<null>	<null>	<null>	18309725	100	Using join buffer (Block Nested Loop)
4	7	DERIVED	<derived9>	<null>	ALL	<null>	<null>	<null>	<null>	2034407	100	Using temporary
5	7	DERIVED	<derived8>	<null>	ref	<auto_key0>	<auto_key0>	4	t1.idgame	10	90	Using where
6	8	DERIVED	t	<null>	ALL	PRIMARY	<null>	<null>	<null>	30	100	Using temporary; Using filesort
7	8	DERIVED	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
8	8	DERIVED	g	<null>	ref	PRIMARY	PRIMARY	4	njba.b.game_idgame	1	100	<null>
9	9	DERIVED	t	<null>	ALL	PRIMARY	<null>	<null>	<null>	30	100	Using temporary; Using filesort
10	9	DERIVED	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
11	9	DERIVED	g	<null>	ref	PRIMARY	PRIMARY	4	njba.b.game_idgame	1	100	<null>
12	6	DEPENDENT SUBQUERY	t	<null>	eq_ref	PRIMARY	PRIMARY	4	s.win_team_id	1	100	Using where
13	5	DEPENDENT SUBQUERY	t	<null>	eq_ref	PRIMARY	PRIMARY	4	s.loss_team_id	1	100	Using where
14	4	DEPENDENT SUBQUERY	t	<null>	eq_ref	PRIMARY	PRIMARY	4	s.win_team_id	1	100	Using where
15	3	DEPENDENT SUBQUERY	t	<null>	eq_ref	PRIMARY	PRIMARY	4	s.loss_team_id	1	100	Using where

Optimizations

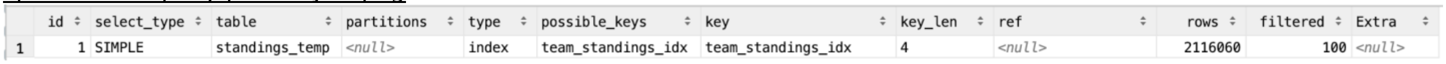
- **Explicitly ORDER BY After GROUP BY**
  By default, the database sorts all 'GROUP BY col1, col2, ...' queries as if you specified 'ORDER BY col1, col2, ...' in the query as well. If a query includes a GROUP BY clause but you want to avoid the overhead of sorting the result, you can suppress sorting by specifying 'ORDER BY NULL'.

- **Avoid Subqueries**
  Since MySQL 5.7 does not have Common Table Expressions (CTE), subqueries were used but they are not optimized well by the optimizer. Instead, to optimize this, I created a temporary table that holds the data, which also includes relevant search indices.

```
-- #2 After Optimization Results (Story #3 Sub-task 2)
-- ================================================================


-- 5 s 835 ms
-- 4 s 973 ms
-- 5 s 198 ms
DROP TABLE IF EXISTS standings_temp;


CREATE TEMPORARY TABLE standings_temp AS
(
  SELECT season_idseason,
    t.idteam,
    t.conference,
    t.division,
    s.win_team_id,
    s.loss_team_id,
CASE
  WHEN s.win_team_id = t.idteam THEN 1
  ELSE 0
  END AS game_win,
CASE
  WHEN s.loss_team_id = t.idteam THEN 1
  ELSE 0
  END AS game_loss,
CASE
  WHEN (s.win_team_id = t.idteam)
    AND conference = (SELECT t.conference
                FROM team t
                WHERE s.loss_team_id = t.idteam) THEN 1
  ELSE 0
  END AS conf_win,
CASE
  WHEN (s.loss_team_id = t.idteam)
    AND conference = (SELECT t.conference
                FROM team t
                WHERE s.win_team_id = t.idteam) THEN 1
  ELSE 0
  END AS conf_loss,
CASE
  WHEN (s.win_team_id = t.idteam)
    AND division = (SELECT t.division
               FROM team t
               WHERE s.loss_team_id = t.idteam) THEN 1
  ELSE 0
  END AS div_win,
CASE
  WHEN (s.loss_team_id = t.idteam)
    AND division = (SELECT t.division
               FROM team t
               WHERE s.win_team_id = t.idteam) THEN 1
  ELSE 0
  END AS div_loss,
CASE
  WHEN (s.win_team_id = t.idteam)
    AND home_team_id = win_team_id THEN 1
```

```
          ELSE 0
          END AS home_win,
        CASE
          WHEN (s.loss_team_id = t.idteam)
            AND home_team_id = loss_team_id THEN 1
          ELSE 0
          END AS home_loss,
        CASE
          WHEN (s.win_team_id = t.idteam)
            AND away_team_id = win_team_id THEN 1
          ELSE 0
          END AS away_win,
        CASE
          WHEN (s.loss_team_id = t.idteam)
            AND away_team_id = loss_team_id THEN 1
          ELSE 0
          END AS away_loss
 FROM totals s
         JOIN
      team t
);

CREATE INDEX season_standings_idx
  ON standings_temp (season_idseason);

CREATE INDEX team_standings_idx
  ON standings_temp (idteam);

EXPLAIN EXTENDED
(
    SELECT standings_temp.season_idseason,
           standings_temp.idteam,
           SUM(standings_temp.game_win)                  AS wins,
           SUM(standings_temp.game_loss)                 AS loses,
           ROUND((SUM(standings_temp.game_win) /
                 SUM(standings_temp.game_win +
                    standings_temp.game_loss)) * 100, 1)     AS win_pct,
           CONCAT(CAST(SUM(standings_temp.conf_win) AS CHAR(2)),
                 ' - ',
            CAST(SUM(standings_temp.conf_loss) AS CHAR(2))) AS conf_record,
        CONCAT(CAST(SUM(standings_temp.div_win) AS CHAR(2)),
            '-',
            CAST(SUM(standings_temp.div_loss) AS CHAR(2)))  AS div_record,
        CONCAT(CAST(SUM(standings_temp.home_win) AS CHAR(2)),
            '-',
            CAST(SUM(standings_temp.home_loss) AS CHAR(2))) AS home_record,
        CONCAT(CAST(SUM(standings_temp.away_win) AS CHAR(2)),
            '-',
            CAST(SUM(standings_temp.away_loss) AS CHAR(2))) AS away_record
  FROM standings_temp
  GROUP BY standings_temp.idteam
  ORDER BY NULL
);
```

optimization-query-plan2-after.png

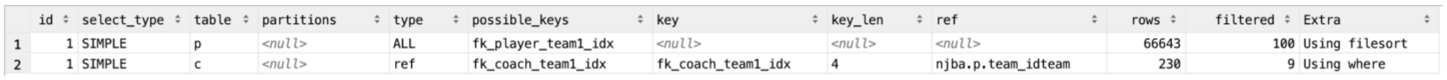


	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	standings_temp	<null>	index	team_standings_idx	team_standings_idx	4	<null>	2116060	100	<null>

Query #3

```
-- #3 Before Optimiziation Results (No Story)
--       Find all the Assistant coaches full names that have the
--       same full name AS players on their team's roster
-- ===============================================================
-- 16 s 551 ms
-- 16 s 441 ms
-- 17 s 52 ms
EXPLAIN EXTENDED
(
  SELECT c.full_name, p.team_idteam
  FROM coach c
       JOIN player p
        ON c.full_name = p.full_name AND
          p.team_idteam = c.team_idteam
  WHERE title <> 'Head Coach'
  ORDER BY team_idteam
);
```

optimization-query-plan3-before.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	p	<null>	ALL	fk_player_team1_idx	<null>	<null>	<null>	66643	100	Using filesort
2	1	SIMPLE	c	<null>	ref	fk_coach_team1_idx	fk_coach_team1_idx	4	njba.p.team_idteam	230	9	Using where

Optimizations

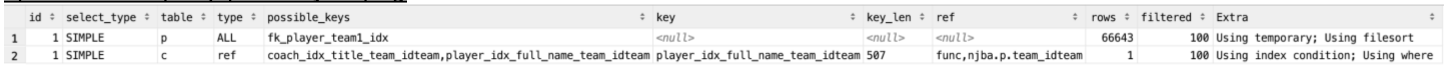
- **Create Optimal Indexes**
  Indexing recommendations are pending in the indexing tab above. These indexes are an integral part of this optimization effort and should be created beforetesting the execution duration of the optimized query.

- **Avoid Comparing Columns From Different Types**
  Joining or filtering using columns of different types in the same condition may cause performance degradation. The database will have to perform a cast foreach of these values before performing the comparison. Make sure to alter the column types so that common comparisons will be done between two columns of the same type.

```
-- #3 After Optimization Results (No Story)
--      Find all the Assistant coaches full names that have the
--      same full name AS players on their team's roster
-- ================================================================
-- 316 ms
-- 410 ms
-- 602 ms

DROP INDEX coach_idx_title_team_idteam ON name_type_merge;
DROP INDEX player_idx_full_name_team_idteam ON name_type_merge;
DROP TABLE IF EXISTS name_type_merge;

CREATE TEMPORARY TABLE name_type_merge AS (
   SELECT idcoach,
          team_idteam,
          first_name,
          last_name,
          CAST(full_name AS CHAR(125)) AS full_name,
          title
   FROM coach
);

CREATE INDEX coach_idx_title_team_idteam
  ON name_type_merge (`title`,`team_idteam`);

CREATE INDEX player_idx_full_name_team_idteam
  ON name_type_merge (`full_name`,`team_idteam`);
```

```
SELECT c.full_name,
       p.team_idteam
FROM
    name_type_merge c
JOIN
    player p
        ON
            c.full_name = p.full_name

        AND
            p.team_idteam = c.team_idteam
where c.title <> 'Head Coach'
ORDER BY
    c.team_idteam;
```

optimization-query-plan3-after.png



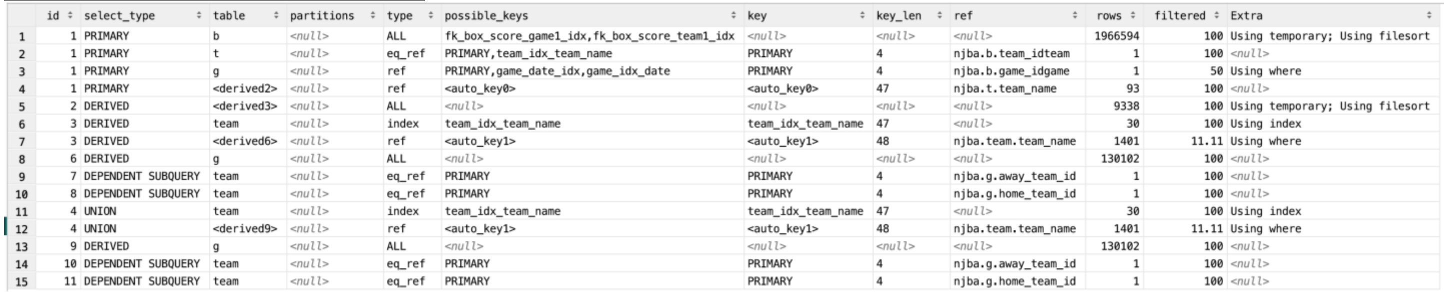
	id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	p	ALL	fk_player_team1_idx	<null>	<null>	<null>	66643	100	Using temporary; Using filesort
2	1	SIMPLE	c	ref	coach_idx_title_team_idteam,player_idx_full_name_team_idteam	player_idx_full_name_team_idteam	507	func,njba.p.team_idteam	1	100	Using index condition; Using where

Query #4

9. As a *Fan*, I want to view team stats so that I can see a team's averages over various time periods (e.g. month-to-month stats, season averages, etc.)

```
-- #4 Before Optimization Results (Story #9 Team Averages)
-- ==================================================================
-- 21 s 556 ms
-- 21 s 729 ms
-- 22 s 110 ms
EXPLAIN EXTENDED
(
    SELECT t.team_name,
        ROUND(SUM(mins / gp.games_played), 0)                AS mins,
        ROUND(SUM(fgm) / gp.games_played, 0)                 AS team_fgm,
        ROUND(SUM(fga) / gp.games_played, 0)                 AS team_fga,
        ROUND((ROUND(SUM(fgm) / gp.games_played, 0) /
            ROUND(SUM(fga) / gp.games_played, 0)) * 100, 1)  AS team_fg_pct,
        ROUND(SUM(fg3) / gp.games_played, 0)                 AS team_fg3,
        ROUND(SUM(fg3a) / gp.games_played, 0)                AS team_fg3a,
        ROUND((ROUND(SUM(fg3) / gp.games_played, 0) /
            ROUND(SUM(fg3a) / gp.games_played, 0)) * 100, 1) AS team_fg3_pct,
        ROUND(SUM(ft) / gp.games_played, 0)                 AS team_ft,
        ROUND(SUM(fta) / gp.games_played, 0)                AS team_fta,
        ROUND((ROUND(SUM(ft) / gp.games_played, 0) /
            ROUND(SUM(fta) / gp.games_played, 0)) * 100, 1)  AS team_ft_pct,
        ROUND(SUM(orb) / gp.games_played, 0)                AS team_orb,
        ROUND(SUM(drb) / gp.games_played, 0)                AS team_drb,
        ROUND(SUM(trb) / gp.games_played, 0)                AS team_trb,
        ROUND(SUM(ast) / gp.games_played, 0)                AS team_assists,
        ROUND(SUM(blk) / gp.games_played, 0)                AS team_blocks,
        ROUND(SUM(tov) / gp.games_played, 0)                AS team_turnovers,
        ROUND(SUM(fouls) / gp.games_played, 0)              AS team_fouls,
        ROUND(SUM(pts) / gp.games_played, 0)                AS team_points
    FROM game g
         JOIN box_score b
            ON idgame = game_idgame
         JOIN team t
            ON team_idteam = idteam
         JOIN games_played gp
            ON t.team_name = gp.team_name
    WHERE g.date BETWEEN start_date() AND end_date()
    GROUP BY 1
    ORDER BY 1 ASC
);
```

optimization-query-plan4-before.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	b	<null>	ALL	fk_box_score_game1_idx,fk_box_score_team1_idx	<null>	<null>	<null>	1966594	100	Using temporary; Using filesort
2	1	PRIMARY	t	<null>	eq_ref	PRIMARY,team_idx_team_name	PRIMARY	4	njba.b.team_idteam	1	100	<null>
3	1	PRIMARY	g	<null>	ref	PRIMARY,game_date_idx,game_idx_date	PRIMARY	4	njba.b.game_idgame	1	50	Using where
4	1	PRIMARY	<derived2>	<null>	ref	<auto_key0>	<auto_key0>	47	njba.t.team_name	93	100	<null>
5	2	DERIVED	<derived3>	<null>	ALL	<null>	<null>	<null>	<null>	9338	100	Using temporary; Using filesort
6	3	DERIVED	team	<null>	index	team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
7	3	DERIVED	<derived6>	<null>	ref	<auto_key1>	<auto_key1>	48	njba.team.team_name	1401	11.11	Using where
8	6	DERIVED	g	<null>	ALL	<null>	<null>	<null>	<null>	130102	100	<null>
9	7	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.away_team_id	1	100	<null>
10	8	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.home_team_id	1	100	<null>
11	4	UNION	team	<null>	index	team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
12	4	UNION	<derived9>	<null>	ref	<auto_key1>	<auto_key1>	48	njba.team.team_name	1401	11.11	Using where
13	9	DERIVED	g	<null>	ALL	<null>	<null>	<null>	<null>	130102	100	<null>
14	10	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.away_team_id	1	100	<null>
15	11	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.home_team_id	1	100	<null>

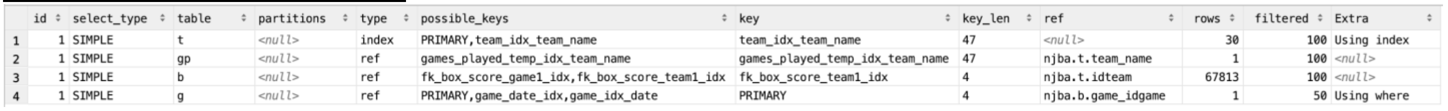
Optimizations

- **Create Optimal Indexes**
  Indexing recommendations are pending in the indexing tab above. These indexes are an integral part of this optimization effort and should be created beforetesting the execution duration of the optimized query.

-- #4 After Optimization Results (Story #9 Team Averages)

```
-- ================================================================
-- 10 s 444 ms
-- 10 s 396 ms
-- 10 s 383 ms

DROP INDEX games_played_temp_idx_team_name ON games_played_temp;
DROP TABLE IF EXISTS games_played_temp;

CREATE TEMPORARY TABLE games_played_temp AS
(
  SELECT team_name, sum(games_played) games_played
  FROM
  (
    SELECT count(*) AS games_played, team_name
    FROM schedule s
         JOIN team
           ON team_name = home
    WHERE date BETWEEN start_date() AND end_date()
    GROUP BY team_name

    UNION ALL

    SELECT count(*) AS games_played, team_name
    FROM schedule s
         JOIN team ON team_name = away
    WHERE date BETWEEN start_date() AND end_date()
    GROUP BY team_name
  ) AS game_total
  GROUP BY team_name
);

ALTER TABLE `game` ADD INDEX `game_idx_date` (`date`);
ALTER TABLE `games_played_temp` ADD INDEX `games_played_temp_idx_team_name` (`team_name`);

EXPLAIN EXTENDED
(
  SELECT t.team_name,
      ROUND(SUM(mins / gp.games_played), 0)          AS mins,
      ROUND(SUM(fgm) / gp.games_played, 0)           AS team_fgm,
      ROUND(SUM(fga) / gp.games_played, 0)                    AS team_fga,
      ROUND((ROUND(SUM(fgm) / gp.games_played, 0) /
          ROUND(SUM(fga) / gp.games_played, 0)) * 100, 1)  AS team_fg_pct,
      ROUND(SUM(fg3) / gp.games_played, 0)                    AS team_fg3,
      ROUND(SUM(fg3a) / gp.games_played, 0)                   AS team_fg3a,
      ROUND((ROUND(SUM(fg3) / gp.games_played, 0) /
          ROUND(SUM(fg3a) / gp.games_played, 0)) * 100, 1) AS team_fg3_pct,
      ROUND(SUM(ft) / gp.games_played, 0)                     AS team_ft,
      ROUND(SUM(fta) / gp.games_played, 0)                    AS team_fta,
      ROUND((ROUND(SUM(ft) / gp.games_played, 0) /
          ROUND(SUM(fta) / gp.games_played, 0)) * 100, 1)  AS team_ft_pct,
      ROUND(SUM(orb) / gp.games_played, 0)                    AS team_orb,
      ROUND(SUM(drb) / gp.games_played, 0)                    AS team_drb,
      ROUND(SUM(trb) / gp.games_played, 0)                    AS team_trb,
      ROUND(SUM(ast) / gp.games_played, 0)                    AS team_assists,
      ROUND(SUM(blk) / gp.games_played, 0)                    AS team_blocks,
      ROUND(SUM(tov) / gp.games_played, 0)                    AS team_turnovers,
      ROUND(SUM(fouls) / gp.games_played, 0)                  AS team_fouls,
      ROUND(SUM(pts) / gp.games_played, 0)                    AS team_points
  FROM game g
       JOIN box_score b
           ON idgame = game_idgame
       JOIN team t
           ON team_idteam = idteam
       JOIN games_played_temp gp
           ON t.team_name = gp.team_name
  WHERE g.date BETWEEN start_date() AND end_date()
  GROUP BY 1
  ORDER BY 1 ASC
);
```

<u>*optimization-query-plan4-after.png*</u>



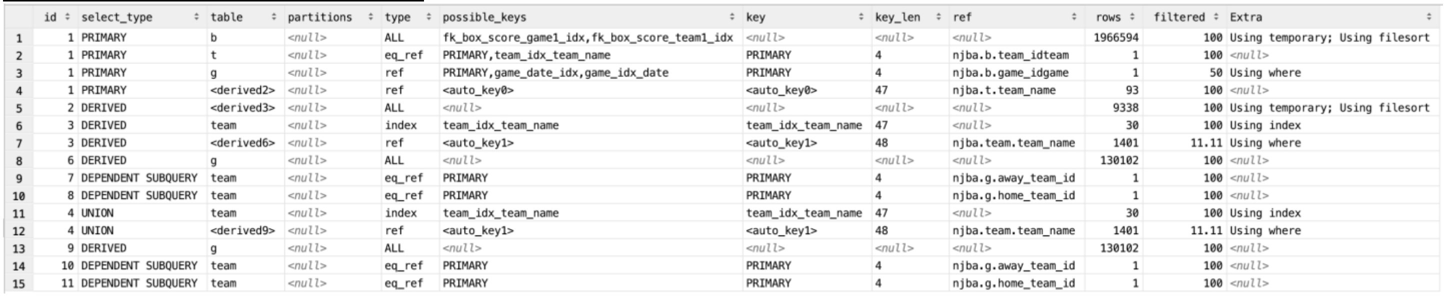
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	t	<null>	index	PRIMARY,team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
2	1	SIMPLE	gp	<null>	ref	games_played_temp_idx_team_name	games_played_temp_idx_team_name	47	njba.t.team_name	1	100	<null>
3	1	SIMPLE	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
4	1	SIMPLE	g	<null>	ref	PRIMARY,game_date_idx,game_idx_date	PRIMARY	4	njba.b.game_idgame	1	50	Using where

Query #5

9. As a *Fan*, I want to view team stats so that I can see a team's averages over various time periods (e.g. month-to-month stats, season averages, etc.)

```
-- #5 Before Optimiziation (Story #9 Team Raw)
-- ================================================================
-- 16 s 529 ms
-- 16 s 538 ms
-- 17 s 255 ms

EXPLAIN EXTENDED
(
    SELECT t.team_name,
           ROUND(SUM(mins), 0)                  AS mins,
           SUM(fgm)                             AS team_fgm,
           SUM(fga)                             AS team_fga,
           ROUND((SUM(fgm) / SUM(fga)) * 100, 1)  AS team_fg_pct,
           SUM(fg3)                             AS team_fg3,
           SUM(fg3a)                            AS team_fg3a,
           ROUND((SUM(fg3) / SUM(fg3a)) * 100, 1) AS team_fg3_pct,
           SUM(ft)                              AS team_ft,
           SUM(fta)                             AS team_fta,
           ROUND((SUM(ft) / SUM(fta)) * 100, 1)   AS team_ft_pct,
           SUM(orb)                             AS team_orb,
           SUM(drb)                             AS team_drb,
           SUM(trb)                             AS team_trb,
           SUM(ASt)                             AS team_assists,
           SUM(blk)                             AS team_blocks,
           SUM(tov)                             AS team_turnovers,
           SUM(fouls)                           AS team_fouls,
           SUM(pts)                             AS team_points
    FROM game g
             JOIN box_score b
                  ON idgame = game_idgame
             JOIN team t
                  ON team_idteam = idteam
             JOIN games_played gp
                  ON t.team_name = gp.team_name
    WHERE g.date BETWEEN start_date() AND end_date()
    GROUP BY 1
    ORDER BY 1 ASC
);
```

optimization-query-plan5-before.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	PRIMARY	b	<null>	ALL	fk_box_score_game1_idx,fk_box_score_team1_idx	<null>	<null>	<null>	1966594	100	Using temporary; Using filesort
2	1	PRIMARY	t	<null>	eq_ref	PRIMARY,team_idx_team_name	PRIMARY	4	njba.b.team_idteam	1	100	<null>
3	1	PRIMARY	g	<null>	ref	PRIMARY,game_date_idx,game_idx_date	PRIMARY	4	njba.b.game_idgame	1	50	Using where
4	1	PRIMARY	<derived2>	<null>	ref	<auto_key0>	<auto_key0>	47	njba.t.team_name	93	100	<null>
5	2	DERIVED	<derived3>	<null>	ALL	<null>	<null>	<null>	<null>	9338	100	Using temporary; Using filesort
6	3	DERIVED	team	<null>	index	team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
7	3	DERIVED	<derived6>	<null>	ref	<auto_key1>	<auto_key1>	48	njba.team.team_name	1401	11.11	Using where
8	6	DERIVED	g	<null>	ALL	<null>	<null>	<null>	<null>	130102	100	<null>
9	7	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.away_team_id	1	100	<null>
10	8	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.home_team_id	1	100	<null>
11	4	UNION	team	<null>	index	team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
12	4	UNION	<derived9>	<null>	ref	<auto_key1>	<auto_key1>	48	njba.team.team_name	1401	11.11	Using where
13	9	DERIVED	g	<null>	ALL	<null>	<null>	<null>	<null>	130102	100	<null>
14	10	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.away_team_id	1	100	<null>
15	11	DEPENDENT SUBQUERY	team	<null>	eq_ref	PRIMARY	PRIMARY	4	njba.g.home_team_id	1	100	<null>

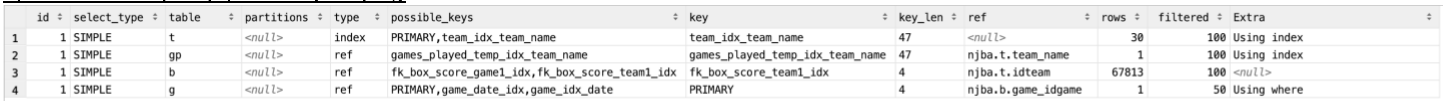
Optimizations

- **Create Optimal Indexes**
  Indexing recommendations are pending in the indexing tab above. These indexes are an integral part of this optimization effort and should be created beforetesting the execution duration of the optimized query.

```
-- #5 After Optimiziation (Story #9 Team Raw)

-- ============================================================

--  9 s 753 ms

-- 10 s 122 ms

--  9 s 756 ms


DROP INDEX games_played_temp_idx_team_name ON games_played_temp;

DROP TABLE IF EXISTS games_played_temp;


CREATE TEMPORARY TABLE games_played_temp AS

(

   SELECT team_name, sum(games_played) games_played

   FROM

   (

      SELECT count(*) AS games_played, team_name

      FROM schedule s

            JOIN team

               ON team_name = home

      WHERE date BETWEEN start_date() AND end_date()

      GROUP BY team_name


      UNION ALL


      SELECT count(*) AS games_played, team_name

      FROM schedule s

            JOIN team ON team_name = away
```

```
        WHERE date BETWEEN start_date() AND end_date()
        GROUP BY team_name
    ) AS game_total
    GROUP BY team_name
);

ALTER TABLE `game` ADD INDEX `game_idx_date` (`date`);
ALTER TABLE `games_played_temp` ADD INDEX `games_played_temp_idx_team_name` (`team_name`);

EXPLAIN EXTENDED
(
    SELECT t.team_name,
           ROUND(SUM(mins), 0)                  AS mins,
           SUM(fgm)                             AS team_fgm,
           SUM(fga)                             AS team_fga,
           ROUND((SUM(fgm) / SUM(fga)) * 100, 1)  AS team_fg_pct,
           SUM(fg3)                             AS team_fg3,
           SUM(fg3a)                            AS team_fg3a,
           ROUND((SUM(fg3) / SUM(fg3a)) * 100, 1) AS team_fg3_pct,
           SUM(ft)                              AS team_ft,
           SUM(fta)                             AS team_fta,
           ROUND((SUM(ft) / SUM(fta)) * 100, 1)   AS team_ft_pct,
           SUM(orb)                             AS team_orb,
           SUM(drb)                             AS team_drb,
           SUM(trb)                             AS team_trb,
           SUM(ASt)                             AS team_assists,
           SUM(blk)                             AS team_blocks,
           SUM(tov)                             AS team_turnovers,
           SUM(fouls)                           AS team_fouls,
           SUM(pts)                             AS team_points
    FROM game g
            JOIN box_score b
                ON idgame = game_idgame
            JOIN team t
                ON team_idteam = idteam
            JOIN games_played_temp gp
                ON t.team_name = gp.team_name
    WHERE g.date BETWEEN start_date() AND end_date()
    GROUP BY 1
    ORDER BY 1 ASC
);
```

optimization-query-plan5-after.png



	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	t	<null>	index	PRIMARY,team_idx_team_name	team_idx_team_name	47	<null>	30	100	Using index
2	1	SIMPLE	gp	<null>	ref	games_played_temp_idx_team_name	games_played_temp_idx_team_name	47	njba.t.team_name	1	100	Using index
3	1	SIMPLE	b	<null>	ref	fk_box_score_game1_idx,fk_box_score_team1_idx	fk_box_score_team1_idx	4	njba.t.idteam	67813	100	<null>
4	1	SIMPLE	g	<null>	ref	PRIMARY,game_date_idx,game_idx_date	PRIMARY	4	njba.b.game_idgame	1	50	Using where