CS 411

Summer 2024

Project Track 1 Stage 3

Database Implementation and Indexing (30%)

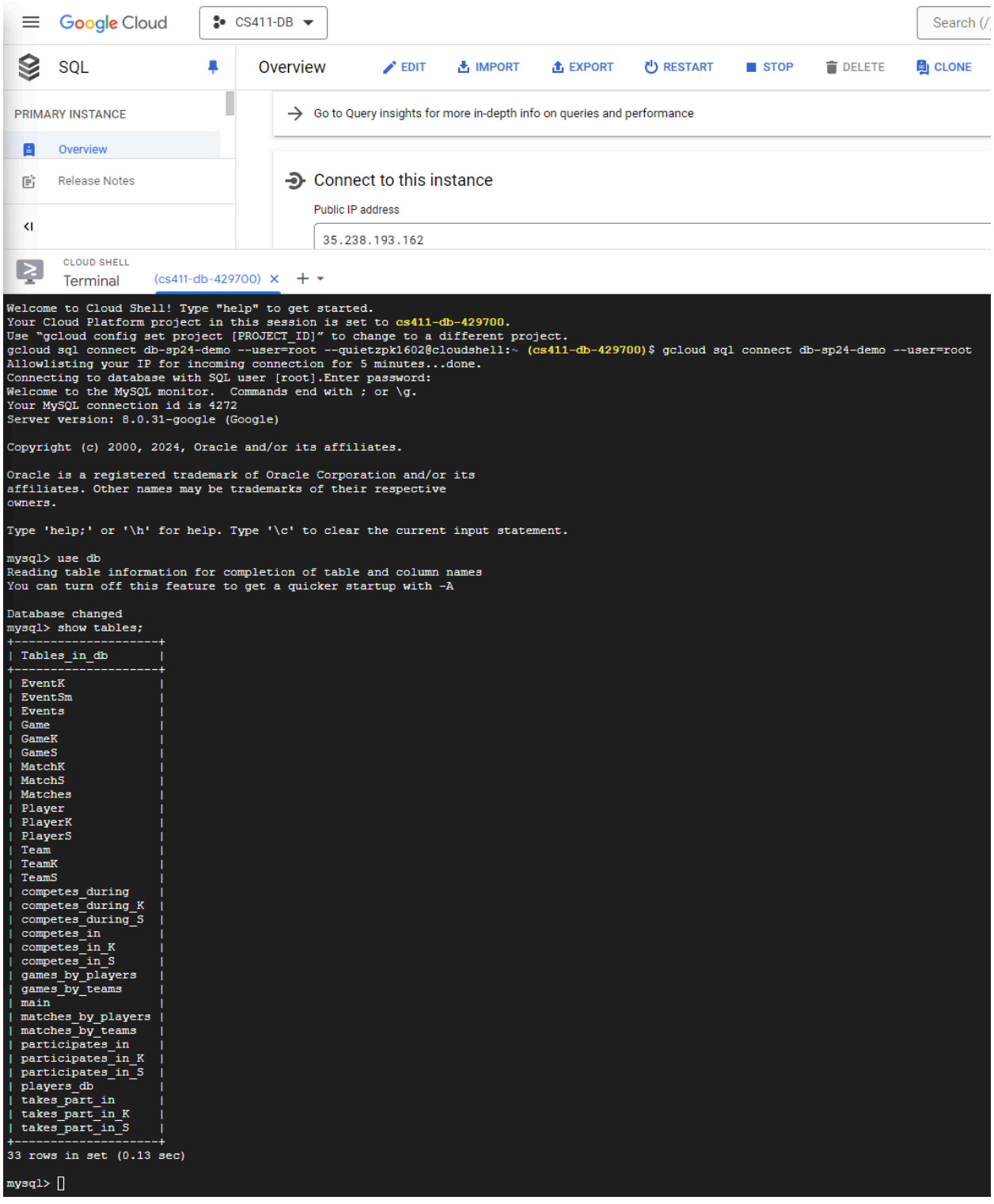
Ethan, Zachary, Ivan, Suyash

July 17, 2024

Stage 1 & Stage 2 Fixes

For Stage 3, we have fixed many of the errors that we previously had on our last 2 stage submissions. Please look at the files “UML Diagram.pdf”, “Project Proposal.pdf”, and “Conceptual Database Design.pdf” in the release.

GCP Database Implementation



- Here is a picture that shows all of the database tables that we currently have implemented. The database tables from our Kaggle source are “games_by_players”,

"game_by_teams", "main", "matches_by_players", "matches_by_teams", and "players_db". Our five entities from our UML diagram are "Player", "Game", "Team", Matches", and "Events". Our main-main relationships are "participates_in", "competes_in", "takes_part_in", and "competes_during". The other database tables present are either resized versions of these previous tables, or they are tables for the queries we have made.

DDL Commands

```
CREATE TABLE Player (
      player_id INT NOT NULL PRIMARY KEY,
      player_slug VARCHAR(255) NOT NULL,
      player_name VARCHAR(255),
      player_tag VARCHAR(255) NOT NULL,
      player_country VARCHAR(2) NOT NULL
);

CREATE TABLE Team (
      team_id INT NOT NULL PRIMARY KEY,
      team_region VARCHAR(10) NOT NULL,
      team_slug VARCHAR(255) NOT NULL,
      team_name VARCHAR(255) NOT NULL
);

CREATE TABLE Game (
      game_id INT NOT NULL,
      player_tag VARCHAR(255) NOT NULL,
      winner BOOL NOT NULL,
      platform VARCHAR(10) NOT NULL,
      PRIMARY KEY (game_id, player_tag),
      FOREIGN KEY (player_tag) REFERENCES Player(player_tag)
);

CREATE TABLE Matches (
      match_id INT NOT NULL PRIMARY KEY,
      score INT NOT NULL,
      winner BOOL NOT NULL,
      color VARCHAR(10) NOT NULL,
      platform VARCHAR(10) NOT NULL,
      team_name VARCHAR(255) NOT NULL,
      player_tag VARCHAR(255) NOT NULL,
      FOREIGN KEY (player_tag) REFERENCES Player(player_tag)
```

```
);

CREATE TABLE Events (
      event_id INT NOT NULL,
      event VARCHAR(25) NOT NULL,
      event_split VARCHAR(255) NOT NULL,
      event_region VARCHAR(25) NOT NULL,
      event_slug VARCHAR(255) NOT NULL,
      event_tier VARCHAR(25) NOT NULL
      PRIMARY KEY (event_id, event_region, event_tier),
);

CREATE TABLE participates_in (
      core_shots INT NOT NULL,
      core_goals INT NOT NULL,
      core_saves INT NOT NULL,
      core_assists INT NOT NULL,
      core_score INT NOT NULL,
      core_shooting_percentage FLOAT NOT NULL,
      boost_bpm FLOAT NOT NULL,
      boost_bcpm FLOAT NOT NULL,
      boost_avg_amount FLOAT NOT NULL,
      boost_amount_collected INT NOT NULL,
      boost_amount_stolen INT NOT NULL,
      boost_amount_collected_big INT NOT NULL,
      boost_amount_stolen_big INT NOT NULL,
      boost_amount_collected_small INT NOT NULL,
      boost_amount_stolen_small INT NOT NULL,
      boost_count_collected_big INT NOT NULL,
      boost_count_stolen_big INT NOT NULL,
      boost_count_collected_small INT NOT NULL,
      boost_count_stolen_small INT NOT NULL,
      boost_amount_overfill INT NOT NULL,
      boost_amount_overfill_stolen INT NOT NULL,
      boost_amount_used_while_supersonic INT NOT NULL,
      boost_time_zero_boost FLOAT NOT NULL,
      boost_percent_zero_boost FLOAT NOT NULL,
      boost_time_full_boost FLOAT NOT NULL,
      boost_percent_full_boost FLOAT NOT NULL,
      boost_time_boost_0_25 FLOAT NOT NULL,
      boost_time_boost_25_50 FLOAT NOT NULL,
      boost_time_boost_50_75 FLOAT NOT NULL,
      boost_time_boost_75_100 FLOAT NOT NULL,
```

```
    boost_percent_boost_0_25 FLOAT NOT NULL,
    boost_percent_boost_25_50 FLOAT NOT NULL,
    boost_percent_boost_50_75 FLOAT NOT NULL,
    boost_percent_boost_75_100 FLOAT NOT NULL,
    movement_avg_speed FLOAT NOT NULL,
    movement_total_distance FLOAT NOT NULL,
    movement_time_supersonic_speed FLOAT NOT NULL,
    movement_time_boost_speed FLOAT NOT NULL,
    movement_time_slow_speed FLOAT NOT NULL,
    movement_time_ground FLOAT NOT NULL,
    movement_time_low_air FLOAT NOT NULL,
    movement_time_high_air FLOAT NOT NULL,
    movement_time_powerslide FLOAT NOT NULL,
    movement_count_powerslide INT NOT NULL,
    movement_avg_powerslide_duration FLOAT NOT NULL,
    movement_avg_speed_percentage FLOAT NOT NULL,
    movement_percent_slow_speed FLOAT NOT NULL,
    movement_percent_boost_speed FLOAT NOT NULL,
    movement_percent_supersonic_speed FLOAT NOT NULL,
    movement_percent_ground FLOAT NOT NULL,
    movement_percent_low_air FLOAT NOT NULL,
    movement_percent_high_air FLOAT NOT NULL,
    positioning_avg_distance_to_ball FLOAT NOT NULL,
    positioning_avg_distance_to_ball_possession FLOAT NOT NULL,
    positioning_avg_distance_to_ball_no_possession FLOAT NOT NULL,
    positioning_avg_distance_to_mates FLOAT NOT NULL,
    positioning_time_defensive_third FLOAT NOT NULL,
    positioning_time_neutral_third FLOAT NOT NULL,
    positioning_time_offensive_third FLOAT NOT NULL,
    positioning_time_defensive_half FLOAT NOT NULL,
    positioning_time_offensive_half FLOAT NOT NULL,
    positioning_time_behind_ball FLOAT NOT NULL,
    positioning_time_in_front_ball FLOAT NOT NULL,
    positioning_time_most_back FLOAT NOT NULL,
    positioning_time_most_forward FLOAT NOT NULL,
    positioning_goals_against_while_last_defender INT NOT NULL,
    positioning_time_closest_to_ball FLOAT NOT NULL,
    positioning_time_farthest_from_ball FLOAT NOT NULL,
    positioning_percent_defensive_third FLOAT NOT NULL,
    positioning_percent_offensive_third FLOAT NOT NULL,
    positioning_percent_neutral_third FLOAT NOT NULL,
    positioning_percent_defensive_half FLOAT NOT NULL,
    positioning_percent_offensive_half FLOAT NOT NULL,
```

```
      positioning_percent_behind_ball FLOAT NOT NULL,
      positioning_percent_in_front_ball FLOAT NOT NULL,
      positioning_percent_most_back FLOAT NOT NULL,
      positioning_percent_most_forward FLOAT NOT NULL,
      positioning_percent_closest_to_ball FLOAT NOT NULL,
      positioning_percent_farthest_from_ball FLOAT NOT NULL,
      demo_inflicted INT NOT NULL,
      demo_taken INT NOT NULL,
      advanced_goal_participation FLOAT NOT NULL,
      advanced_rating FLOAT NOT NULL,
      advanced_mvp INT NOT NULL,
      platform VARCHAR(100) NOT NULL,
      platform_id VARCHAR(255) NOT NULL,
      car_id INT NOT NULL,
      car_name VARCHAR(255) NOT NULL,
      steering_sensitivity FLOAT NOT NULL,
      camera_fov FLOAT NOT NULL,
      camera_height FLOAT NOT NULL,
      camera_pitch FLOAT NOT NULL,
      camera_distance FLOAT NOT NULL,
      camera_stiffness FLOAT NOT NULL,
      camera_swivel_speed FLOAT NOT NULL,
      camera_transition_speed FLOAT NOT NULL,
      player_id VARCHAR(255),
      player_tag VARCHAR(255),
      player_country VARCHAR(2),
      PRIMARY KEY (player_id, game_id, player_tag),
      FOREIGN KEY game_id VARCHAR(255) NOT NULL,
      FOREIGN KEY player_id VARCHAR(255) NOT NULL,
      FOREIGN KEY player_tag VARCHAR(255) NOT NULL
);

CREATE TABLE competes_in (
      game_id VARCHAR(255) NOT NULL,
      team_id VARCHAR(255) NOT NULL,
      team_name VARCHAR(100) NOT NULL,
      ball_possession_time REAL NOT NULL,
      ball_time_in_side REAL NOT NULL,
      core_shots INT NOT NULL,
      core_goals INT NOT NULL,
      core_saves INT NOT NULL,
      core_assists INT NOT NULL,
      core_score INT NOT NULL,
```

```
    core_shooting_percentage FLOAT NOT NULL,
    boost_bpm FLOAT NOT NULL,
    boost_bcpm FLOAT NOT NULL,
    boost_avg_amount FLOAT NOT NULL,
    boost_amount_collected INT NOT NULL,
    boost_amount_stolen INT NOT NULL,
    boost_amount_collected_big INT NOT NULL,
    boost_amount_stolen_big INT NOT NULL,
    boost_amount_collected_small INT NOT NULL,
    boost_amount_stolen_small INT NOT NULL,
    boost_count_collected_big INT NOT NULL,
    boost_count_stolen_big INT NOT NULL,
    boost_count_collected_small INT NOT NULL,
    boost_count_stolen_small INT NOT NULL,
    boost_amount_overfill INT NOT NULL,
    boost_amount_overfill_stolen INT NOT NULL,
    boost_amount_used_while_supersonic INT NOT NULL,
    boost_time_zero_boost FLOAT NOT NULL,
    boost_time_full_boost FLOAT NOT NULL,
    boost_time_boost_0_25 FLOAT NOT NULL,
    boost_time_boost_25_50 FLOAT NOT NULL,
    boost_time_boost_50_75 FLOAT NOT NULL,
    boost_time_boost_75_100 FLOAT NOT NULL,
    movement_total_distance FLOAT NOT NULL,
    movement_time_supersonic_speed FLOAT NOT NULL,
    movement_time_boost_speed FLOAT NOT NULL,
    movement_time_slow_speed FLOAT NOT NULL,
    movement_time_ground FLOAT NOT NULL,
    movement_time_low_air FLOAT NOT NULL,
    movement_time_high_air FLOAT NOT NULL,
    movement_time_powerslide FLOAT NOT NULL,
    movement_count_powerslide INT NOT NULL,
    positioning_time_defensive_third FLOAT NOT NULL,
    positioning_time_neutral_third FLOAT NOT NULL,
    positioning_time_offensive_third FLOAT NOT NULL,
    positioning_time_defensive_half FLOAT NOT NULL,
    positioning_time_offensive_half FLOAT NOT NULL,
    positioning_time_behind_ball FLOAT NOT NULL,
    positioning_time_in_front_ball FLOAT NOT NULL,
    demo_inflicted INT NOT NULL,
    demo_taken INT NOT NULL,
    winner VARCHAR(50) NOT NULL,
    PRIMARY KEY (team_id, game_id, player_tag)
```

```
);

CREATE TABLE takes_part_in (
      match_id VARCHAR(255) NOT NULL,
      player_id VARCHAR(255) NOT NULL,
      player_tag VARCHAR(255) NOT NULL,
      core_shots INT NOT NULL,
      core_goals INT NOT NULL,
      core_saves INT NOT NULL,
      core_assists INT NOT NULL,
      core_score INT NOT NULL,
      core_shooting_percentage FLOAT NOT NULL,
      boost_bpm FLOAT NOT NULL,
      boost_bcpm FLOAT NOT NULL,
      boost_avg_amount FLOAT NOT NULL,
      boost_amount_collected INT NOT NULL,
      boost_amount_stolen INT NOT NULL,
      boost_amount_collected_big INT NOT NULL,
      boost_amount_stolen_big INT NOT NULL,
      boost_amount_collected_small INT NOT NULL,
      boost_amount_stolen_small INT NOT NULL,
      boost_count_collected_big INT NOT NULL,
      boost_count_stolen_big INT NOT NULL,
      boost_count_collected_small INT NOT NULL,
      boost_count_stolen_small INT NOT NULL,
      boost_amount_overfill INT NOT NULL,
      boost_amount_overfill_stolen INT NOT NULL,
      boost_amount_used_while_supersonic INT NOT NULL,
      boost_time_zero_boost FLOAT NOT NULL,
      boost_percent_zero_boost FLOAT NOT NULL,
      boost_time_full_boost FLOAT NOT NULL,
      boost_percent_full_boost FLOAT NOT NULL,
      boost_time_boost_0_25 FLOAT NOT NULL,
      boost_time_boost_25_50 FLOAT NOT NULL,
      boost_time_boost_50_75 FLOAT NOT NULL,
      boost_time_boost_75_100 FLOAT NOT NULL,
      boost_percent_boost_0_25 FLOAT NOT NULL,
      boost_percent_boost_25_50 FLOAT NOT NULL,
      boost_percent_boost_50_75 FLOAT NOT NULL,
      boost_percent_boost_75_100 FLOAT NOT NULL,
      movement_avg_speed FLOAT NOT NULL,
      movement_total_distance FLOAT NOT NULL,
      movement_time_supersonic_speed FLOAT NOT NULL,
```

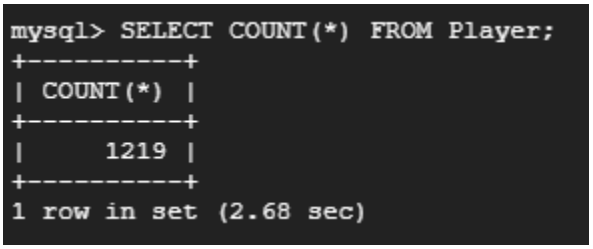
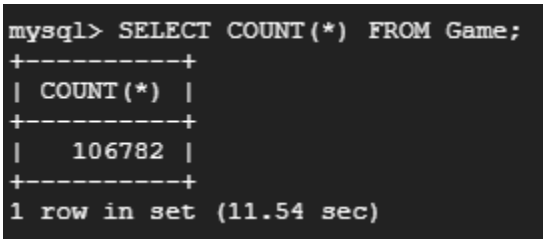
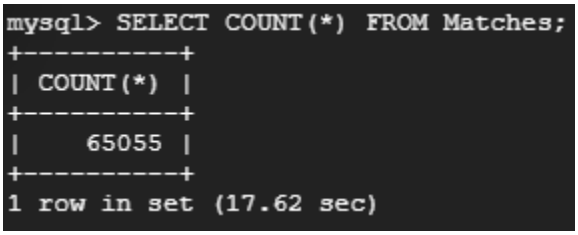
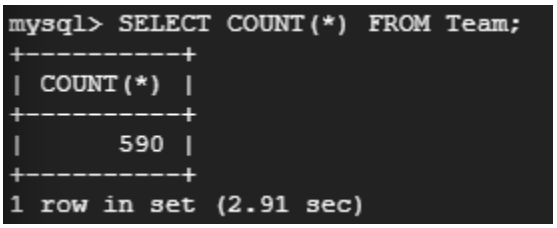
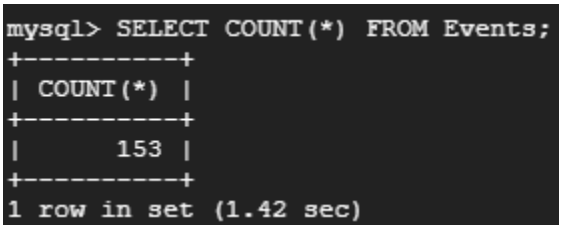
```
    movement_time_boost_speed FLOAT NOT NULL,
    movement_time_slow_speed FLOAT NOT NULL,
    movement_time_ground FLOAT NOT NULL,
    movement_time_low_air FLOAT NOT NULL,
    movement_time_high_air FLOAT NOT NULL,
    movement_time_powerslide FLOAT NOT NULL,
    movement_count_powerslide INT NOT NULL,
    movement_avg_powerslide_duration FLOAT NOT NULL,
    movement_avg_speed_percentage FLOAT NOT NULL,
    movement_percent_slow_speed FLOAT NOT NULL,
    movement_percent_boost_speed FLOAT NOT NULL,
    movement_percent_supersonic_speed FLOAT NOT NULL,
    movement_percent_ground FLOAT NOT NULL,
    movement_percent_low_air FLOAT NOT NULL,
    movement_percent_high_air FLOAT NOT NULL,
    positioning_avg_distance_to_ball FLOAT NOT NULL,
    positioning_avg_distance_to_ball_possession FLOAT NOT NULL,
    positioning_avg_distance_to_ball_no_possession FLOAT NOT NULL,
    positioning_avg_distance_to_mates FLOAT NOT NULL,
    positioning_time_defensive_third FLOAT NOT NULL,
    positioning_time_neutral_third FLOAT NOT NULL,
    positioning_time_offensive_third FLOAT NOT NULL,
    positioning_time_defensive_half FLOAT NOT NULL,
    positioning_time_offensive_half FLOAT NOT NULL,
    positioning_time_behind_ball FLOAT NOT NULL,
    positioning_time_in_front_ball FLOAT NOT NULL,
    positioning_time_most_back FLOAT NOT NULL,
    positioning_time_most_forward FLOAT NOT NULL,
    positioning_goals_against_while_last_defender INT NOT NULL,
    positioning_time_closest_to_ball FLOAT NOT NULL,
    positioning_time_farthest_from_ball FLOAT NOT NULL,
    positioning_percent_defensive_third FLOAT NOT NULL,
    positioning_percent_offensive_third FLOAT NOT NULL,
    positioning_percent_neutral_third FLOAT NOT NULL,
    positioning_percent_defensive_half FLOAT NOT NULL,
    positioning_percent_offensive_half FLOAT NOT NULL,
    positioning_percent_behind_ball FLOAT NOT NULL,
    positioning_percent_in_front_ball FLOAT NOT NULL,
    positioning_percent_most_back FLOAT NOT NULL,
    positioning_percent_most_forward FLOAT NOT NULL,
    positioning_percent_closest_to_ball FLOAT NOT NULL,
    positioning_percent_farthest_from_ball FLOAT NOT NULL,
    demo_inflicted INT NOT NULL,
```

```
      demo_taken INT NOT NULL,
      advanced_goal_participation FLOAT NOT NULL,
      advanced_rating FLOAT NOT NULL,
      score INT NOT NULL,
      player_id VARCHAR(255) NOT NULL,
      player_tag VARCHAR(255) NOT NULL,
      player_country VARCHAR(2) NOT NULL,
      PRIMARY KEY (player_id, team_id)
);

CREATE TABLE competes_during (
      match_id VARCHAR(255) NOT NULL,
      team_id VARCHAR(255) NOT NULL,
      team_name VARCHAR(100) NOT NULL,
      core_shots INT NOT NULL,
      core_goals INT NOT NULL,
      core_saves INT NOT NULL,
      core_assists INT NOT NULL,
      core_score INT NOT NULL,
      core_shooting_percentage FLOAT NOT NULL,
      boost_bpm FLOAT NOT NULL,
      boost_bcpm FLOAT NOT NULL,
      boost_avg_amount FLOAT NOT NULL,
      boost_amount_collected INT NOT NULL,
      boost_amount_stolen INT NOT NULL,
      boost_amount_collected_big INT NOT NULL,
      boost_amount_stolen_big INT NOT NULL,
      boost_amount_collected_small INT NOT NULL,
      boost_amount_stolen_small INT NOT NULL,
      boost_count_collected_big INT NOT NULL,
      boost_count_stolen_big INT NOT NULL,
      boost_count_collected_small INT NOT NULL,
      boost_count_stolen_small INT NOT NULL,
      boost_amount_overfill INT NOT NULL,
      boost_amount_overfill_stolen INT NOT NULL,
      boost_amount_used_while_supersonic INT NOT NULL,
      boost_time_zero_boost FLOAT NOT NULL,
      boost_time_full_boost FLOAT NOT NULL,
      boost_time_boost_0_25 FLOAT NOT NULL,
      boost_time_boost_25_50 FLOAT NOT NULL,
      boost_time_boost_50_75 FLOAT NOT NULL,
      boost_time_boost_75_100 FLOAT NOT NULL,
      movement_total_distance FLOAT NOT NULL,
```

```
      movement_time_supersonic_speed FLOAT NOT NULL,
      movement_time_boost_speed FLOAT NOT NULL,
      movement_time_slow_speed FLOAT NOT NULL,
      movement_time_ground FLOAT NOT NULL,
      movement_time_low_air FLOAT NOT NULL,
      movement_time_high_air FLOAT NOT NULL,
      movement_time_powerslide FLOAT NOT NULL,
      movement_count_powerslide INT NOT NULL,
      positioning_time_defensive_third FLOAT NOT NULL,
      positioning_time_neutral_third FLOAT NOT NULL,
      positioning_time_offensive_third FLOAT NOT NULL,
      positioning_time_defensive_half FLOAT NOT NULL,
      positioning_time_offensive_half FLOAT NOT NULL,
      positioning_time_behind_ball FLOAT NOT NULL,
      positioning_time_in_front_ball FLOAT NOT NULL,
      demo_inflicted INT NOT NULL,
      demo_taken INT NOT NULL,
      score INT NOT NULL,
      winner VARCHAR(50) NOT NULL,
PRIMARY KEY (team_id, match_id)
);
```

Tables

Out of our five entities, we have three that have above 1000 rows.

- Player has 1219 rows
  - 
```
mysql> SELECT COUNT(*) FROM Player;
+----------+
| COUNT(*) |
+----------+
|     1219 |
+----------+
1 row in set (2.68 sec)
```
- Game has 106,782 rows
  - 
```
mysql> SELECT COUNT(*) FROM Game;
+----------+
| COUNT(*) |
+----------+
|   106782 |
+----------+
1 row in set (11.54 sec)
```
- Matches has 65,055 rows
  - 
```
mysql> SELECT COUNT(*) FROM Matches;
+----------+
| COUNT(*) |
+----------+
|    65055 |
+----------+
1 row in set (17.62 sec)
```
- Team has 590 rows (DOES NOT COUNT)
  - 
```
mysql> SELECT COUNT(*) FROM Team;
+----------+
| COUNT(*) |
+----------+
|      590 |
+----------+
1 row in set (2.91 sec)
```
- Events has 153 rows (DOES NOT COUNT)
  - 
```
mysql> SELECT COUNT(*) FROM Events;
+----------+
| COUNT(*) |
+----------+
|      153 |
+----------+
1 row in set (1.42 sec)
```

Out of our four relationships, we have four that have above 1000 rows.

- participates_in has 106,796 rows

```
mysql> SELECT COUNT(*) FROM participates_in;
+----------+
| COUNT(*) |
+----------+
|   106796 |
+----------+
1 row in set (49.23 sec)
```

- competes_in has 35,595 rows

```
mysql> SELECT COUNT(*) FROM competes_in;
+----------+
| COUNT(*) |
+----------+
|    35595 |
+----------+
1 row in set (21.91 sec)
```

- takes_part_in has 26,177 rows

```
mysql> SELECT COUNT(*) FROM takes_part_in;
+----------+
| COUNT(*) |
+----------+
|    26177 |
+----------+
1 row in set (9.51 sec)
```

- competes_during has 10,595 rows

```
mysql> SELECT COUNT(*) FROM competes_during;
+----------+
| COUNT(*) |
+----------+
|    10595 |
+----------+
1 row in set (17.42 sec)
```

This shows that we have 3+ tables that contain at least 1000 rows.

Advanced Queries

Here are the four advanced queries that we have developed:

Query #1:

```
mysql> SELECT
    -> p.player_id,
    -> p.player_name,
    -> SUM(g.core_goals) AS total_score
    -> FROM
    -> players_db p
    -> JOIN
    -> games_by_players g
    -> ON
    -> p.player_id = g.player_id
    -> GROUP BY
    -> p.player_id,
    -> p.player_name
    -> ORDER BY
    -> total_score DESC
    -> LIMIT 15;
+--------------------------+-------------------------+-------------+
| player_id                | player_name             | total_score |
+--------------------------+-------------------------+-------------+
| 5f3d8fdd95f40596eae23ef5 | Shogo Ikeyama           |         354 |
| 5f3d8fdd95f40596eae2414a | Shoki Minamigawa        |         348 |
| 5f3d8fdd95f40596eae23f9e | Finlay Ferguson         |         344 |
| 5f3d8fdd95f40596eae23f88 | Ahmad Abdullah          |         341 |
| 5f3d8fdd95f40596eae24148 | Louis Christian Thamrun |         334 |
| 5f3d8fdd95f40596eae23dba | Reed Wilen              |         315 |
| 5faeab91e9ce4ed313ea9570 | Axel Touret             |         311 |
| 60c46a3a9fc1a47e5f11199a | David Morgenrood        |         306 |
| 60bb4abf6d9c1362119ca7d4 | Gareth Spiers           |         304 |
| 5fc43098133774d1c57b03f0 | Mohammed Alotaibi       |         302 |
| 5f3d8fdd95f40596eae23f8f | Max Ng                  |         297 |
| 5f3d8fdd95f40596eae23eb6 | Jason Corral            |         295 |
| 5fd40fc10e831f1d52bd96a5 | Enzo Grondein           |         294 |
| 5f3d8fdd95f40596eae23f60 | Gabriel Vieira Cardoso  |         284 |
| 5f3d8fdd95f40596eae24337 | Yan Nolasco             |         284 |
+--------------------------+-------------------------+-------------+
15 rows in set (41.17 sec)
```

- This advanced query finds the top scoring players
- Join multiple relations
- Aggregation via GROUP BY

Code:

```
SELECT
      p.player_id,
      p.player_name,
      SUM(g.core_goals) AS total_score
FROM
      Player p
JOIN
      games_by_players g
ON
      p.player_id = g.player_id
GROUP BY
      p.player_id,
      p.player_name
ORDER BY
      total_score DESC
LIMIT 15;
```

Query #2:

```
mysql> SELECT
    -> m.player_tag,
    -> COUNT(g.game_id) AS matches_played
    -> FROM
    -> GameS g
    -> JOIN
    -> MatchS m
    -> ON
    -> g.player_tag = m.player_tag
    -> GROUP BY
    -> m.player_tag
    -> ORDER BY
    -> matches_played DESC
    -> LIMIT 15;
+---------------+----------------+
| player_tag    | matches_played |
+---------------+----------------+
| Scout         |            308 |
| Tobee         |            308 |
| Shmoopernator |            308 |
| cavemanben    |            225 |
| Siki          |            210 |
| Decka         |            210 |
| Smash         |            204 |
| Viic          |            204 |
| RoToR         |            204 |
| Snowy         |            200 |
| Cobbo         |            200 |
| vortexioz     |            200 |
| LionBlaze     |            190 |
| Roll Dizz     |            190 |
| Shock         |            190 |
+---------------+----------------+
15 rows in set (1.67 sec)
```

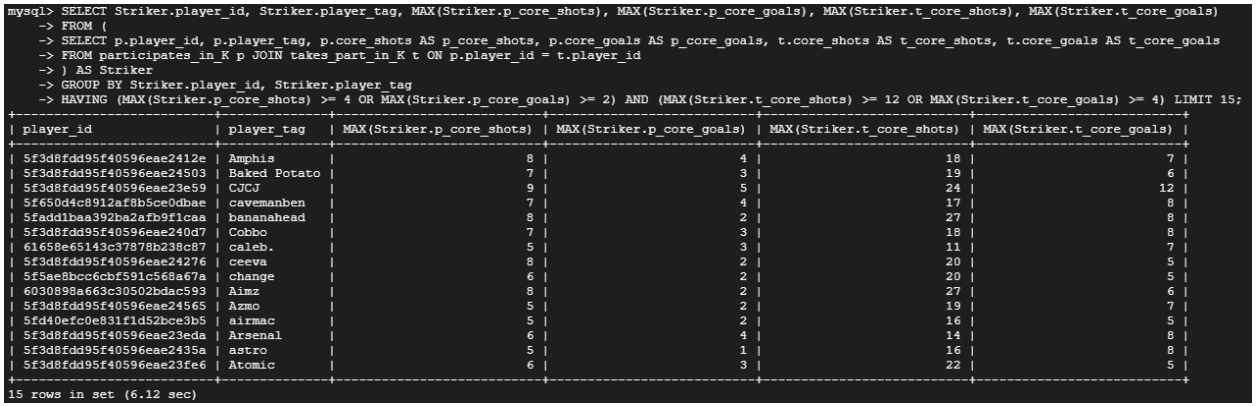
- This advanced query gives the number of matches by player
- Join multiple relations
- Aggregation via GROUP BY

Code:

```
SELECT
```

```
m.player_tag,
COUNT(g.game_id) AS matches_played
FROM
GameK g
JOIN
MatchK m
ON
g.player_tag = m.player_tag
GROUP BY
m.player_tag
ORDER BY
matches_played DESC
LIMIT 15;
```

Query #3:



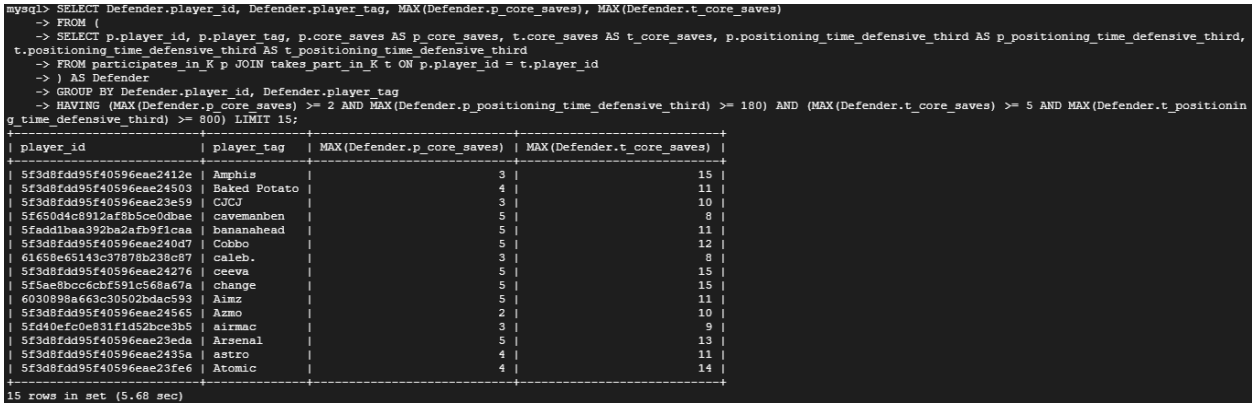
```
mysql> SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots), MAX(Striker.p_core_goals), MAX(Striker.t_core_shots), MAX(Striker.t_core_goals)
    -> FROM (
    -> SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots, p.core_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS t_core_goals
    -> FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id = t.player_id
    -> ) AS Striker
    -> GROUP BY Striker.player_id, Striker.player_tag
    -> HAVING (MAX(Striker.p_core_shots) >= 4 OR MAX(Striker.p_core_goals) >= 2) AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4) LIMIT 15;
+--------------------------+-------------+-------------------------+-------------------------+-------------------------+-------------------------+
| player_id                | player_tag  | MAX(Striker.p_core_shots) | MAX(Striker.p_core_goals) | MAX(Striker.t_core_shots) | MAX(Striker.t_core_goals) |
+--------------------------+-------------+-------------------------+-------------------------+-------------------------+-------------------------+
| 5f3d8fdd95f40596eae2412e | Amphis      |                       8 |                       4 |                      18 |                       7 |
| 5f3d8fdd95f40596eae24503 | Baked Potato |                       7 |                       3 |                      19 |                       6 |
| 5f3d8fdd95f40596eae23e59 | CJCJ        |                       9 |                       5 |                      24 |                      12 |
| 5f650d4c8912af8b5ce0dbae | cavemanben  |                       7 |                       4 |                      17 |                       8 |
| 5fadd1baa392ba2afb9f1caa | bananahead  |                       8 |                       2 |                      27 |                       8 |
| 5f3d8fdd95f40596eae240d7 | Cobbo       |                       7 |                       3 |                      18 |                       8 |
| 61658e65143c37878b238c87 | caleb.      |                       5 |                       3 |                      11 |                       7 |
| 5f3d8fdd95f40596eae24276 | ceeva       |                       8 |                       2 |                      20 |                       5 |
| 5f5ae8bcc6cbf591c568a67a | change      |                       6 |                       2 |                      20 |                       5 |
| 6030898a663c30502bdac593 | Aimz        |                       8 |                       2 |                      27 |                       6 |
| 5f3d8fdd95f40596eae24565 | Azmo        |                       5 |                       2 |                      19 |                       7 |
| 5fd40efc0e831f1d52bce3b5 | airmac      |                       5 |                       2 |                      16 |                       5 |
| 5f3d8fdd95f40596eae23eda | Arsenal     |                       6 |                       4 |                      14 |                       8 |
| 5f3d8fdd95f40596eae2435a | astro       |                       5 |                       1 |                      16 |                       8 |
| 5f3d8fdd95f40596eae23fe6 | Atomic      |                       6 |                       3 |                      22 |                       5 |
+--------------------------+-------------+-------------------------+-------------------------+-------------------------+-------------------------+
15 rows in set (6.12 sec)
```

- This advanced query shows the top strikers in the tournament
- Aggregation via GROUP BY
- Subqueries that cannot be easily replaced by a join.

Code:

```
CREATE TABLE StrikersByPlayer
AS
SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots),
MAX(Striker.p_core_goals), MAX(Striker.t_core_shots),
MAX(Striker.t_core_goals)
FROM (
SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots,
p.core_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS
t_core_goals
FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id =
t.player_id
) AS Striker
GROUP BY Striker.player_id, Striker.player_tag
HAVING (MAX(Striker.p_core_shots) >= 4 OR MAX(Striker.p_core_goals) >= 2)
AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4)
LIMIT 15;
```

Query #4:



```
mysql> SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves), MAX(Defender.t_core_saves)
    -> FROM (
    -> SELECT p.player_id, p.player_tag, p.core_saves AS p_core_saves, t.core_saves AS t_core_saves, p.positioning_time_defensive_third AS p_positioning_time_defensive_third,
 t.positioning_time_defensive_third AS t_positioning_time_defensive_third
    -> FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id = t.player_id
    -> ) AS Defender
    -> GROUP BY Defender.player_id, Defender.player_tag
    -> HAVING (MAX(Defender.p_core_saves) >= 2 AND MAX(Defender.p_positioning_time_defensive_third) >= 180) AND (MAX(Defender.t_core_saves) >= 5 AND MAX(Defender.t_positionin
g_time_defensive_third) >= 800) LIMIT 15;
+--------------------------+-------------+---------------------------+---------------------------+
| player_id                | player_tag  | MAX(Defender.p_core_saves) | MAX(Defender.t_core_saves) |
+--------------------------+-------------+---------------------------+---------------------------+
| 5f3d8fdd95f40596eae2412e | Amphis      |                         3 |                        15 |
| 5f3d8fdd95f40596eae24503 | Baked Potato |                         4 |                        11 |
| 5f3d8fdd95f40596eae23e59 | CJCJ        |                         3 |                        10 |
| 5f650d4c8912af8b5ce0dbae | cavemanben  |                         5 |                         8 |
| 5fadd1baa392ba2afb9f1caa | bananahead  |                         5 |                        11 |
| 5f3d8fdd95f40596eae240d7 | Cobbo       |                         5 |                        12 |
| 61658e65143c37878b238c87 | caleb.      |                         3 |                         8 |
| 5f3d8fdd95f40596eae24276 | ceeva       |                         5 |                        15 |
| 5f5ae8bcc6cbf591c568a67a | change      |                         5 |                        15 |
| 6030898a663c30502bdac593 | Aimz        |                         5 |                        11 |
| 5f3d8fdd95f40596eae24565 | Azmo        |                         2 |                        10 |
| 5fd40efc0e831f1d52bce3b5 | airmac      |                         3 |                         9 |
| 5f3d8fdd95f40596eae23eda | Arsenal     |                         5 |                        13 |
| 5f3d8fdd95f40596eae2435a | astro       |                         4 |                        11 |
| 5f3d8fdd95f40596eae23fe6 | Atomic      |                         4 |                        14 |
+--------------------------+-------------+---------------------------+---------------------------+
15 rows in set (5.68 sec)
```

- This advanced query shows the top defenders in the tournament
- Aggregation via GROUP BY
- Subqueries that cannot be easily replaced by a join.

Code:

```
CREATE TABLE DefendersByPlayer
AS
SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves),
MAX(Defender.t_core_saves)
FROM (
SELECT p.player_id, p.player_tag, p.core_saves AS p_core_saves,
t.core_saves AS t_core_saves, p.positioning_time_defensive_third AS
p_positioning_time_defensive_third, t.positioning_time_defensive_third AS
t_positioning_time_defensive_third
FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id =
t.player_id
) AS Defender
GROUP BY Defender.player_id, Defender.player_tag
HAVING (MAX(Defender.p_core_saves) >= 2 AND
MAX(Defender.p_positioning_time_defensive_third) >= 180) AND
(MAX(Defender.t_core_saves) >= 5 AND
MAX(Defender.t_positioning_time_defensive_third) >= 800) LIMIT 15;
```

Indexing Analysis

Query #1:

Default:

- Total Cost: 127596.60

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    -> p.player_id,
    -> p.player_name,
    -> SUM(g.core_goals) AS total_score
    -> FROM
    -> Player p
    -> JOIN
    -> games_by_players g
    -> ON
    -> p.player_id = g.player_id
    -> GROUP BY
    -> p.player_id,
    -> p.player_name
    -> ORDER BY
    -> total_score DESC;
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
| -> Sort: total_score DESC  (actual time=1430.854..1430.947 rows=1219 loops=1)
    -> Table scan on <temporary>  (actual time=1429.977..1430.339 rows=1219 loops=1)
        -> Aggregate using temporary table  (actual time=1429.975..1429.975 rows=1219 loops=1)
            -> Nested loop inner join  (cost=127596.60 rows=103043) (actual time=105.521..1223.429 rows=106796 loops=1)
                -> Filter: (g.player_id is not null)  (cost=14249.30 rows=103043) (actual time=72.152..855.840 rows=106796 loops=1)
                    -> Table scan on g  (cost=14249.30 rows=103043) (actual time=72.148..843.230 rows=106796 loops=1)
                -> Single-row index lookup on p using PRIMARY (player_id=g.player_id)  (cost=1.00 rows=1) (actual time=0.003..0.003 rows=1 loops=106796)
 |
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
1 row in set (1.43 sec)
```

Index on player_name:

- Total Cost: 127596.60 (Same)

```
mysql> CREATE INDEX idx_player_player_name ON Player(player_name);
Query OK, 0 rows affected (0.41 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT p.player_id, p.player_name, SUM(g.core_goals) AS total_score FROM Player p JOIN games_by_players g ON p.player_id = g.playe
r_id GROUP BY p.player_id, p.player_name ORDER BY total_score DESC;
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
-----------------+
| EXPLAIN




                 |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
-----------------+
| -> Sort: total_score DESC  (actual time=1192.774..1192.858 rows=1219 loops=1)
    -> Table scan on <temporary>  (actual time=1191.833..1192.205 rows=1219 loops=1)
        -> Aggregate using temporary table  (actual time=1191.830..1191.830 rows=1219 loops=1)
            -> Nested loop inner join  (cost=127596.60 rows=103043) (actual time=53.180..993.158 rows=106796 loops=1)
                -> Filter: (g.player_id is not null)  (cost=14249.30 rows=103043) (actual time=53.129..744.965 rows=106796 loops=1)
                    -> Table scan on g  (cost=14249.30 rows=103043) (actual time=53.124..734.875 rows=106796 loops=1)
                -> Single-row index lookup on p using PRIMARY (player_id=g.player_id)  (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1 loops=106796)
 |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
-----------------+
1 row in set (1.19 sec)
```

Index on core_goals:

- Total Cost: 127596.60 (Same)

```
mysql> CREATE INDEX idx_player_core_goals ON games_by_players(core_goals);
Query OK, 0 rows affected (3.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT p.player_id, p.player_name, SUM(g.core_goals) AS total_score FROM Player p JOIN games_by_players g ON p.player_id = g.playe
r_id GROUP BY p.player_id, p.player_name ORDER BY total_score DESC;
+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--------+
| EXPLAIN
                                                                                                                                                          |
+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--------+
| -> Sort: total_score DESC  (actual time=730.269..730.355 rows=1219 loops=1)
    -> Table scan on <temporary>  (actual time=729.344..729.681 rows=1219 loops=1)
        -> Aggregate using temporary table  (actual time=729.342..729.342 rows=1219 loops=1)
            -> Nested loop inner join  (cost=127596.60 rows=103043) (actual time=4.691..557.812 rows=106796 loops=1)
                -> Filter: (g.player_id is not null)  (cost=14249.30 rows=103043) (actual time=2.630..301.727 rows=106796 loops=1)
                    -> Table scan on g  (cost=14249.30 rows=103043) (actual time=2.627..290.799 rows=106796 loops=1)
                -> Single-row index lookup on p using PRIMARY (player_id=g.player_id)  (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1 loops=106796)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--------+
1 row in set (0.74 sec)
```

Analysis:

Adding indexes on ‘player_name’ in ‘Player’ and ‘core_goals’ in ‘games_by_players’ did not change the total cost of the advanced query, which remained at 127596.60 for every index. This tells us that the query performance was not improved by using these indexes. We assume the reason is that the query’s performance is mainly driven by our JOIN operation and the GROUP BY operation, neither of which benefited majorly from the attributes we indexed.

Due to the nature of this advanced query and the hints on not indexing primary keys, our indexing options were limited here. The only attributes we have in our query are ‘player_id’, ‘player_name’, and ‘core_goals’. Our ‘player_id’ is a primary key which means we cannot use it. As a result, we were not able to find a third indexing strategy that follows the guidelines and improves performance.

Despite our attempts, neither index ‘player_name’ or ‘core_goals’ resulted in improvements to our performance. This leads us to believe that for this specific advanced query, these indexing strategies do not have any benefits. The optimizer consistently used the primary key index on ‘player_id’ because it is the most efficient for the JOIN and GROUP BY operations, ignoring the other indexes entirely.

Query #2:

Default:

- Total Cost: 4765.19

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    -> m.player_tag,
    -> COUNT(g.game_id) AS matches_played
    -> FROM
    -> GameK g
    -> JOIN
    -> MatchK m
    -> ON
    -> g.player_tag = m.player_tag
    -> GROUP BY
    -> m.player_tag
    -> ORDER BY
    -> matches_played DESC;
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
| EXPLAIN

+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
| -> Sort: matches_played DESC  (actual time=15.433..15.438 rows=96 loops=1)
    -> Table scan on <temporary>  (actual time=15.361..15.384 rows=96 loops=1)
        -> Aggregate using temporary table  (actual time=15.359..15.359 rows=96 loops=1)
            -> Nested loop inner join  (cost=4765.19 rows=10417) (actual time=0.142..7.917 rows=9488 loops=1)
                -> Filter: (m.player_tag is not null)  (cost=102.00 rows=1000) (actual time=0.100..0.806 rows=1000 loops=1)
                    -> Table scan on m  (cost=102.00 rows=1000) (actual time=0.097..0.701 rows=1000 loops=1)
                -> Covering index lookup on g using PRIMARY (player_tag=m.player_tag)  (cost=3.62 rows=10) (actual time=0.004..0.006 rows=9 loops=1000)
 |
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
1 row in set (0.04 sec)
```

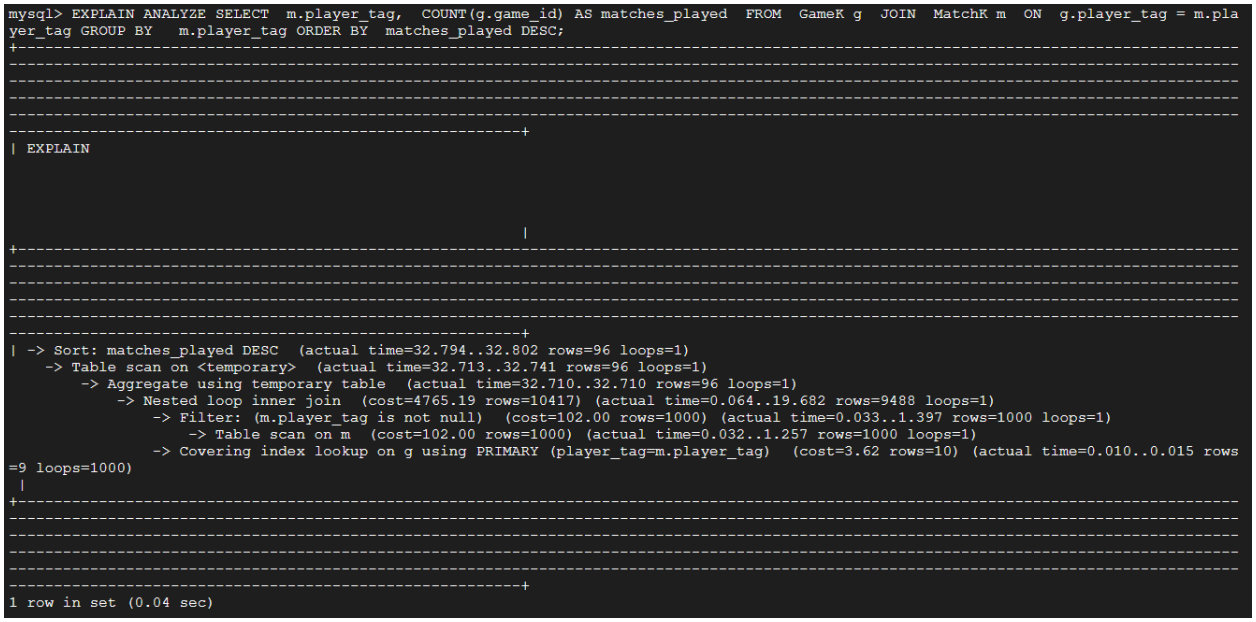
Index on Primary Key player_tag:

- Total Cost: 3659.29

```
mysql> CREATE INDEX idx_game_player_tag ON GameK(player_tag);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT  m.player_tag,  COUNT(g.game_id) AS matches_played  FROM  GameK g  JOIN  MatchK m  ON  g.player_tag = m.pla
yer_tag GROUP BY   m.player_tag ORDER BY  matches_played DESC;
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
| EXPLAIN




                                                                 |
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
| -> Sort: matches_played DESC  (actual time=12.249..12.255 rows=96 loops=1)
    -> Table scan on <temporary>  (actual time=12.182..12.204 rows=96 loops=1)
        -> Aggregate using temporary table  (actual time=12.179..12.179 rows=96 loops=1)
            -> Nested loop inner join  (cost=3659.29 rows=10417) (actual time=0.080..7.307 rows=9488 loops=1)
                -> Filter: (m.player_tag is not null)  (cost=102.00 rows=1000) (actual time=0.038..0.812 rows=1000 loops=1)
                    -> Table scan on m  (cost=102.00 rows=1000) (actual time=0.036..0.707 rows=1000 loops=1)
                -> Covering index lookup on g using idx_game_player_tag (player_tag=m.player_tag)  (cost=2.52 rows=10) (actual time=0.004
..0.006 rows=9 loops=1000)
 |
+------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
1 row in set (0.01 sec)
```

Index on Primary Key game_id:

- Total Cost: 4765.19 (Same)



Analysis:

Indexing on the primary key ‘player_tag’, which is also in the GROUP BY clause, drastically reduces the cost on this table join. The GROUP BY clause is a heavy contributor to the cost since there is a large number of players to go through, so the index ‘player_tag’ improving the performance of that section of the query hugely improved the cost performance. The slight extra costs from the filter, table_scan, and index lookup are vastly outweighed by the cost cut from the ‘player_tag” index on the table join. The index ‘game_id’ did not provide any improvements to cost.

We used the primary keys because we simply did not have any other attributes to index with that weren’t primary keys. However the cost improvement from 4765.19 to 3659.29 on the index ‘player_tag’ speaks for itself, so it is the final index implementation for this query. The index ‘game_id’ did not improve performance, most likely due to it not being in any further filtering or grouping besides in the initial SELECT, so we do not deem it necessary to be in the final implementation.

Query #3:

Default:

- Total Cost: 89781.21

```
mysql> EXPLAIN ANALYZE
    -> SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots), MAX(Striker.p_core_goals), MAX(Striker.t_core_shots), MAX(Striker.t_core_goals)
    -> FROM (
    -> SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots, p.core_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS t_core_goals
    -> FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id = t.player_id
    -> ) AS Striker
    -> GROUP BY Striker.player_id, Striker.player_tag
    -> HAVING (MAX(Striker.p_core_shots) >= 4 OR MAX(Striker.p_core_goals) >= 2) AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4);
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN


+--------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Filter: (((max(p.core_shots) >= 4) or (max(p.core_goals) >= 2)) and ((max(t.core_shots) >= 12) or (max(t.core_goals) >= 4)))  (actual time=295.832..295.864 rows=46 loops=1)
    -> Table scan on <temporary>  (actual time=295.052..295.079 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=295.050..295.050 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=201.136..266.035 rows=13567 loops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=3.670..66.518 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=94.131..196.285 rows=1000 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.31 sec)
```

Index on player_id:

- Total Cost: 6132.08

```
mysql> CREATE INDEX idx_par_player_id ON participates_in_K(player_id);
Query OK, 0 rows affected (0.57 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots), MAX(Striker.p_core_goals), MAX
(Striker.t_core_shots), MAX(Striker.t_core_goals) FROM ( SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots, p.cor
e_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS t_core_goals FROM participates_in_K p JOIN takes_part_i
n_K t ON p.player_id = t.player_id ) AS Striker GROUP BY Striker.player_id, Striker.player_tag HAVING (MAX(Striker.p_core_shot
s) >= 4 OR MAX(Striker.p_core_goals) >= 2) AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4);
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| EXPLAIN





                                                                                   |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
| -> Filter: (((max(p.core_shots) >= 4) or (max(p.core_goals) >= 2)) and ((max(t.core_shots) >= 12) or (max(t.core_goals) >= 4
)))  (actual time=60.153..60.186 rows=46 loops=1)
    -> Table scan on <temporary>  (actual time=58.955..58.984 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=58.951..58.951 rows=52 loops=1)
            -> Nested loop inner join  (cost=6132.08 rows=17228) (actual time=0.723..34.768 rows=13567 loops=1)
                -> Filter: (t.player_id is not null)  (cost=102.25 rows=945) (actual time=0.065..1.445 rows=1000 loops=1)
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.062..1.338 rows=1000 loops=1)
                -> Index lookup on p using idx_par_player_id (player_id=t.player_id)  (cost=4.56 rows=18) (actual time=0.004..
0.031 rows=14 loops=1000)
 |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+
1 row in set (0.09 sec)
```

Index on player_tag:

- Total Cost: 89781.21 (Same)

```
mysql> CREATE INDEX idx_par_player_tag ON participates_in_K(player_tag);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots), MAX(Striker.p_core_goals), MAX
(Striker.t_core_shots), MAX(Striker.t_core_goals) FROM ( SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots, p.cor
e_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS t_core_goals FROM participates_in_K p JOIN takes_part_i
n_K t ON p.player_id = t.player_id ) AS Striker GROUP BY Striker.player_id, Striker.player_tag HAVING (MAX(Striker.p_core_shot
s) >= 4 OR MAX(Striker.p_core_goals) >= 2) AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4);
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| EXPLAIN                                                                          |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| -> Filter: (((max(p.core_shots) >= 4) or (max(p.core_goals) >= 2)) and ((max(t.core_shots) >= 12) or (max(t.core_goals) >= 4
)))  (actual time=23.911..23.938 rows=46 loops=1)
    -> Table scan on <temporary>  (actual time=23.899..23.919 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=23.896..23.896 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=1.907..4.916 rows=13567 lo
ops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=0.032..1.444 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.056..1.270 rows=1000 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
1 row in set (0.03 sec)
```

Index on core_shots:

- Total Cost: 89781.21 (Same)

```
mysql> CREATE INDEX idx_par_core_shots ON participates_in_K(core_shots)
    -> ;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Striker.player_id, Striker.player_tag, MAX(Striker.p_core_shots), MAX(Striker.p_core_goals), MAX
(Striker.t_core_shots), MAX(Striker.t_core_goals) FROM ( SELECT p.player_id, p.player_tag, p.core_shots AS p_core_shots, p.cor
e_goals AS p_core_goals, t.core_shots AS t_core_shots, t.core_goals AS t_core_goals FROM participates_in_K p JOIN takes_part_i
n_K t ON p.player_id = t.player_id ) AS Striker GROUP BY Striker.player_id, Striker.player_tag HAVING (MAX(Striker.p_core_shot
s) >= 4 OR MAX(Striker.p_core_goals) >= 2) AND (MAX(Striker.t_core_shots) >= 12 OR MAX(Striker.t_core_goals) >= 4);
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| EXPLAIN                                                                          |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
| -> Filter: (((max(p.core_shots) >= 4) or (max(p.core_goals) >= 2)) and ((max(t.core_shots) >= 12) or (max(t.core_goals) >= 4
)))  (actual time=53.790..53.832 rows=46 loops=1)
    -> Table scan on <temporary>  (actual time=53.781..53.810 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=53.777..53.777 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=3.490..9.423 rows=13567 lo
ops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=0.033..3.118 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.145..2.550 rows=1000 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
1 row in set (0.05 sec)
```

Analysis:

We tried indexing on attributes ‘player_tag’, ‘player_id’, and ‘core_shots’. While the indices on ‘player_tag’ and ‘core_shots’ did not result in any improvements to cost performance, the ‘player_id’ index resulted in a noticeable improvement in query performance. The attribute ‘player_id’ is unique to each player and is used in a GROUP BY clause. Therefore, using indexing on ‘player_id’ allowed an optimization during the GROUP BY that reduced cost dramatically. Despite also being in the GROUP BY, ‘player_tag’ is not able to provide improvements with indexing because of the existence of duplicate player tags, which need the player_id in addition to be effectively filtered and grouped. Indexing by ‘core_shots’ also did not improve performance as the main grouping is tied to the player, so there isn’t any improvement to be had there. The final index used is on ‘player_id’.

Query #4:

Default:

- Total Cost: 89781.21

```
mysql> EXPLAIN ANALYZE
    -> SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves), MAX(Defender.t_core_saves)
    -> FROM (
    -> SELECT p.player_id, p.player_tag, p.core_saves AS p_core_saves, t.core_saves AS t_core_saves, p.positioning_time_defensive_third A
S p_positioning_time_defensive_third, t.positioning_time_defensive_third AS t_positioning_time_defensive_third
    -> FROM participates_in_K p JOIN takes_part_in_K t ON p.player_id = t.player_id
    -> ) AS Defender
    -> GROUP BY Defender.player_id, Defender.player_tag
    -> HAVING (MAX(Defender.p_core_saves) >= 2 AND MAX(Defender.p_positioning_time_defensive_third) >= 180) AND (MAX(Defender.t_core_save
s) >= 5 AND MAX(Defender.t_positioning_time_defensive_third) >= 800);
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| EXPLAIN


                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| -> Filter: ((max(p.core_saves) >= 2) and (max(p.positioning_time_defensive_third) >= 180) and (max(t.core_saves) >= 5) and (max(t.posit
ioning_time_defensive_third) >= 800))  (actual time=22.801..22.839 rows=41 loops=1)
    -> Table scan on <temporary>  (actual time=21.860..21.888 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=21.858..21.858 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=1.652..4.584 rows=13567 loops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=0.026..1.395 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.043..1.095 rows=1000 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.02 sec)
```

Index on player_id:

- Total Cost: 6132.08

```
mysql> CREATE INDEX idx_par_player_id ON participates_in_K(player_id);
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves), MAX(Defender.t_core_saves) FROM ( SELE
CT p.player_id, p.player_tag, p.core_saves AS p_core_saves, t.core_saves AS t_core_saves, p.positioning_time_defensive_third AS p_positio
ning_time_defensive_third, t.positioning_time_defensive_third AS t_positioning_time_defensive_third FROM participates_in_K p JOIN takes_p
art_in_K t ON p.player_id = t.player_id ) AS Defender GROUP BY Defender.player_id, Defender.player_tag HAVING (MAX(Defender.p_core_saves)
 >= 2 AND MAX(Defender.p_positioning_time_defensive_third) >= 180) AND (MAX(Defender.t_core_saves) >= 5 AND MAX(Defender.t_positioning_ti
me_defensive_third) >= 800);
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| EXPLAIN


                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| -> Filter: ((max(p.core_saves) >= 2) and (max(p.positioning_time_defensive_third) >= 180) and (max(t.core_saves) >= 5) and (max(t.posit
ioning_time_defensive_third) >= 800))  (actual time=60.165..60.205 rows=41 loops=1)
    -> Table scan on <temporary>  (actual time=60.150..60.177 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=60.145..60.145 rows=52 loops=1)
            -> Nested loop inner join  (cost=6132.08 rows=17228) (actual time=1.340..37.875 rows=13567 loops=1)
                -> Filter: (t.player_id is not null)  (cost=102.25 rows=945) (actual time=0.048..1.490 rows=1000 loops=1)
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.045..1.378 rows=1000 loops=1)
                -> Index lookup on p using idx_par_player_id (player_id=t.player_id)  (cost=4.56 rows=18) (actual time=0.004..0.034 rows=
14 loops=1000)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.07 sec)
```

Index on player_tag:

- Total Cost: 89781.21 (Same)

```
mysql> CREATE INDEX idx_par_player_tag ON participates_in_K(player_tag);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves), MAX(Defender.t_core_saves)
FROM ( SELECT p.player_id, p.player_tag, p.core_saves AS p_core_saves, t.core_saves AS t_core_saves, p.positioning_time_defensive_third A
S p_positioning_time_defensive_third, t.positioning_time_defensive_third AS t_positioning_time_defensive_third FROM participates_in_K p J
OIN takes_part_in_K t ON p.player_id = t.player_id ) AS Defender GROUP BY Defender.player_id, Defender.player_tag HAVING (MAX(Defender.p_
core_saves) >= 2 AND MAX(Defender.p_positioning_time_defensive_third) >= 180) AND (MAX(Defender.t_core_saves) >= 5 AND MAX(Defender.t_pos
itioning_time_defensive_third) >= 800);
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                        |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------| -> Filter: ((max(p.c
ore_saves) >= 2) and (max(p.positioning_time_defensive_third) >= 180) and (max(t.core_saves) >= 5) and (max(t.positioning_time_defensive_
third) >= 800))  (actual time=23.360..23.386 rows=41 loops=1)
    -> Table scan on <temporary>  (actual time=23.350..23.368 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=23.347..23.347 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=1.618..4.854 rows=13567 loops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=0.019..1.539 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.043..1.112 rows=1000 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.02 sec)
```

Index on core_shots:

- Total Cost: 89781.21 (Same)

```
mysql> CREATE INDEX idx_par_core_shots ON participates_in_K(core_shots);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT Defender.player_id, Defender.player_tag, MAX(Defender.p_core_saves), MAX(Defender.t_core_saves) FROM ( SELE
CT p.player_id, p.player_tag, p.core_saves AS p_core_saves, t.core_saves AS t_core_saves, p.positioning_time_defensive_third AS p_positio
ning_time_defensive_third, t.positioning_time_defensive_third AS t_positioning_time_defensive_third FROM participates_in_K p JOIN takes_p
art_in_K t ON p.player_id = t.player_id ) AS Defender GROUP BY Defender.player_id, Defender.player_tag HAVING (MAX(Defender.p_core_saves)
 >= 2 AND MAX(Defender.p_positioning_time_defensive_third) >= 180) AND (MAX(Defender.t_core_saves) >= 5 AND MAX(Defender.t_positioning_ti
me_defensive_third) >= 800);
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                        |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| -> Filter: ((max(p.core_saves) >= 2) and (max(p.positioning_time_defensive_third) >= 180) and (max(t.core_saves) >= 5) and (max(t.posit
ioning_time_defensive_third) >= 800))  (actual time=22.560..22.583 rows=41 loops=1)
    -> Table scan on <temporary>  (actual time=22.550..22.565 rows=52 loops=1)
        -> Aggregate using temporary table  (actual time=22.547..22.547 rows=52 loops=1)
            -> Inner hash join (p.player_id = t.player_id)  (cost=89781.21 rows=89586) (actual time=1.706..4.741 rows=13567 loops=1)
                -> Table scan on p  (cost=0.11 rows=948) (actual time=0.023..1.375 rows=1000 loops=1)
                -> Hash
                    -> Table scan on t  (cost=102.25 rows=945) (actual time=0.090..1.205 rows=1000 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.03 sec)
```

Analyze:

The query for Defenders follows the same line of reasoning as the Strikers query above.

We tried indexing on attributes ‘player_tag’, ‘player_id’, and ‘core_shots’. While the indices on ‘player_tag’ and ‘core_shots’ did not result in any improvements to cost performance, the ‘player_id’ index resulted in a noticeable improvement in query performance. The attribute ‘player_id’ is unique to each player and is used in a GROUP BY clause. Therefore, using indexing on ‘player_id’ allowed an optimization during the GROUP BY that reduced cost dramatically. Despite also being in the GROUP BY, ‘player_tag’ is not able to provide improvements with indexing because of the existence of duplicate player tags, which need the player_id in addition to be effectively filtered and grouped. Indexing by ‘core_shots’ also did not improve performance as the main grouping is tied to the player, so there isn’t any improvement to be had there. The final index used is on ‘player_id’.