Dataset Time Interval Calculation

- This query calculates the number of days between the oldest and the newest match in the dataset, providing insight into the dataset's time coverage.
- *Query*:

```
SELECT
  DATE_DIFF(MAX (date), MIN (date), day) AS dataset_time_interval
FROM `sql-sandbox-385913.Final_Exercise.match`;
```

Home Goals Analysis per Season and League

- This query generates statistics for home goals scored per season and league, including minimum, average, mid-range, maximum, and sum of goals, useful for understanding scoring trends.
- *Query*:

```
SELECT
  match.season,
  leagues.name AS league_name,
  MIN(match.home_team_goal) AS min_home_goals,
  MAX(match.home_team_goal) AS max_home_goals,
  (MAX(match.home_team_goal) + MIN(match.home_team_goal)) / 2 AS
  mid_range_home_goals,
  ROUND(AVG(match.home_team_goal), 1) AS avg_home_goals,
  SUM(match.home_team_goal) AS total_home_goals
FROM
  `sql-sandbox-385913.Final_Exercise.match` AS match
  LEFT JOIN `sql-sandbox-385913.Final_Exercise.leagues` AS leagues
    ON match.league_id = leagues.id
GROUP BY
  match.season,
  leagues.name
```

```
      ORDER BY total_home_goals DESC;
```

`[If I consider the minimum of goals scored equal to 1 instead of 0, use: MIN(CASE WHEN match.home_team_goal > 0 THEN match.home_team_goal END) AS min_home_goals]`

Unique Seasons and Matches per League Analysis

- This query determines the number of unique seasons and analyzes the number of matches played by each league per season, highlighting any anomalies in match frequencies.
- *Query*:

```
SELECT
  COUNT(DISTINCT season) AS num_seasons
FROM `sql-sandbox-385913.Final_Exercise.match`;
SELECT
  match.season,
  leagues.name AS league,
  COUNT(match.id) AS num_matches
FROM
  `sql-sandbox-385913.Final_Exercise.match` AS match
  JOIN `sql-sandbox-385913.Final_Exercise.leagues` AS leagues
   ON match.league_id = leagues.id
GROUP BY
  match.season,
  leagues.name
ORDER BY
  match.season,
  leagues.name;


*
SELECT
  match.date,
  home_teams.team_long_name AS home_team,
  away_teams.team_long_name AS away_team,
  match.home_team_goal,
  match.away_team_goal
FROM
  `sql-sandbox-385913.Final_Exercise.match` AS match
  JOIN `sql-sandbox-385913.Final_Exercise.leagues` AS leagues
   ON match.league_id = leagues.id
  JOIN `sql-sandbox-385913.Final_Exercise.team` AS home_teams
   ON match.home_team_api_id = home_teams.team_api_id
  JOIN `sql-sandbox-385913.Final_Exercise.team` AS away_teams
   ON match.away_team_api_id = away_teams.team_api_id
WHERE
  leagues.name = 'Belgium Jupiler League' AND match.season = '2013/2014'
ORDER BY
  match.date;
```

Creation and Filtering of the PlayerBMI Table

- This query first creates the PlayerBMI table with the players' weight in kg, height in meters, and BMI, then filters to show only players with an optimal BMI, providing insights into player fitness.

- *Query*:

  Creation of PlayerBMI Table

```
SELECT
  player_name,
  weight / 2.205 AS kg_weight,
  height / 100 AS m_height,
  ROUND((weight / 2.205)/POWER((height / 100), 2), 1) AS BMI
FROM
  `sql-sandbox-385913.Final_Exercise.player`
GROUP BY
  player_name,
  kg_weight,
  m_height,
  BMI;
```

Filtering for Optimal BMI

```
SELECT
  player_name,
  kg_weight,
  m_height,
  BMI
FROM
  `sql-sandbox-385913.Final_Exercise.PlayerBMI`
WHERE BMI between 18.5 AND 24.9
GROUP BY
  player_name,
  kg_weight,
  m_height,
  BMI
ORDER BY
  BMI DESC;
```

Team with Highest Total Number of Goals Analysis

- This query identifies the team with the highest total number of goals (home and away) in the most recent season, useful for recognizing top-performing teams.

- *Query*:

```
SELECT
  player_name,
  kg_weight,
  m_height,
  BMI
FROM
  `sql-sandbox-385913.Final_Exercise.PlayerBMI`
GROUP BY
  player_name,
  kg_weight,
  m_height,
  BMI
HAVING BMI < 18.5 OR BMI > 24.9
ORDER BY
  BMI DESC;
```

8. Which **Team** has scored the highest total number of goals (home + away) during the most recent available season? How many goals has it scored?

```
SELECT
```

```sql
  m.season,
  t.team_long_name AS team,
  SUM(
      CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal ELSE
      away_team_goal END
  ) AS total_goals
FROM
  `sql-sandbox-385913.Final_Exercise.match` AS m
  JOIN `sql-sandbox-385913.Final_Exercise.team` AS t
   ON m.home_team_api_id = t.team_api_id
   OR m.away_team_api_id = t.team_api_id
WHERE
  m.season = (
    SELECT
      MAX(season)
    FROM
      `sql-sandbox-385913.Final_Exercise.match`
  )
GROUP BY
  m.season,
  t.team_long_name
ORDER BY
  total_goals DESC
LIMIT 1;
```

Team Ranking First in Most Seasons Analysis

- This query shows, for each season, the name of the team that ranked first in terms of total goals scored and identifies which team ranked first in the most seasons.
- *Query*:

1st version (with rank)

```sql
SELECT
  season,
  team,
  total_goals
FROM
  (
    SELECT
      m.season,
      t.team_long_name AS team,
      SUM(
          CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal ELSE
          away_team_goal END
      ) AS total_goals,
      RANK() OVER (
        PARTITION BY m.season
        ORDER BY
          SUM(
               CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal
               ELSE away_team_goal END
          ) DESC
      ) AS rank
    FROM
      `sql-sandbox-385913.Final_Exercise.match` AS m
      JOIN `sql-sandbox-385913.Final_Exercise.team` AS t
       ON m.home_team_api_id = t.team_api_id
       OR m.away_team_api_id = t.team_api_id
    GROUP BY
      m.season,
      t.team_long_name
  )
WHERE
  rank = 1
ORDER BY
  season;
```

2nd version

```
SELECT
  max_tot_goals.season,
  tot_goals_per_team.team_long_name AS team,
  max_tot_goals.max_goals AS total_goal
FROM
  (
    SELECT
      m.season,
      t.team_long_name,
      SUM(
              CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal
              ELSE away_team_goal END
      ) AS total_goals
    FROM
      `sql-sandbox-385913.Final_Exercise.match` AS m
      JOIN `sql-sandbox-385913.Final_Exercise.team` AS t
        ON m.home_team_api_id = t.team_api_id
        OR m.away_team_api_id = t.team_api_id
    GROUP BY
      m.season,
      t.team_long_name
  ) AS tot_goals_per_team
  JOIN (
    SELECT
      season,
      MAX(total_goals) AS max_goals
    FROM
      (
        SELECT
          m.season,
          t.team_long_name,
          SUM(
                CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal
                ELSE away_team_goal END
            ) AS total_goals
          FROM
            `sql-sandbox-385913.Final_Exercise.match` AS m
            JOIN `sql-sandbox-385913.Final_Exercise.team` AS t
             ON m.home_team_api_id = t.team_api_id
             OR m.away_team_api_id = t.team_api_id
          GROUP BY
            m.season,
            t.team_long_name
        ) AS t2
      GROUP BY
        season
    ) AS max_tot_goals
      ON tot_goals_per_team.season = max_tot_goals.season
      AND tot_goals_per_team.total_goals = max_tot_goals.max_goals
ORDER BY
  max_tot_goals.season;
```

TopScorer Table Creation and Pair Combinations Analysis

- This query first creates the TopScorer table with the top 10 teams in terms of total goals scored, then shows all possible pair combinations between those teams, useful for match analysis.
- *Query*:

Creation of the TopScorer table

```
SELECT
  m.season,
  t.team_api_id,
  t.team_long_name AS team,
  SUM(
       CASE WHEN m.home_team_api_id = t.team_api_id THEN home_team_goal ELSE
       away_team_goal END
  ) AS total_goals
FROM
```

```
  `sql-sandbox-385913.Final_Exercise.match` AS m
   JOIN `sql-sandbox-385913.Final_Exercise.team` AS t
    ON m.home_team_api_id = t.team_api_id
    OR m.away_team_api_id = t.team_api_id
WHERE
  season = "2015/2016"
GROUP BY
  m.season,
  t.team_long_name,
  t.team_api_id
ORDER BY
  total_goals DESC
LIMIT 10;
```

Query pair combinations

```
SELECT
  t1.team_api_id AS team1_id,
  t1.team AS team1,
  t2.team_api_id AS team2_id,
  t2.team AS team2
FROM
  `sql-sandbox-385913.Final_Exercise.TopScorer` AS t1
  JOIN `sql-sandbox-385913.Final_Exercise.TopScorer` AS t2
   ON t1.team_api_id < t2.team_api_id
   OR t1.team_api_id > t2.team_api_id;
```

Query pair combinations (count)

```
SELECT
  COUNT (*) AS pair_combinations
FROM
  `sql-sandbox-385913.Final_Exercise.TopScorer` AS t1
  JOIN `sql-sandbox-385913.Final_Exercise.TopScorer` as t2
   ON t1.team_api_id < t2.team_api_id
   OR t1.team_api_id > t2.team_api_id;
```