POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Mehmet Emre Akbulut** **(10972566)**
Yavuz Samet Topcuoglu **(10967930)**

Academic Year: 2023-2024

Contents

1 | Introduction

Upon the delivery of the project, choosing a dataset in the field of football was a no-brainer for us, considering our enthusiasm for the game of football. The main aspect to take into consideration during the selection of the dataset was that it should be suitable for football analytics and that we should be able to find answers to the questions in our minds.

Graph databases excel in managing and interpreting complex relationships and interconnected data. Unlike traditional relational databases, they allow for more flexible and efficient handling of data that is naturally interconnected. That is exactly why we chose to use a graph database because of the nature of our dataset, which includes a multitude of interrelated elements and relationships.

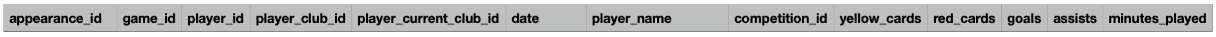
Neo4j, a leading graph database technology, was selected for its robust performance, ease of query language, strong documentation, and existence of materials provided by the lecturers. Its ability to represent complex networks of data as graphs rather than tables makes it an ideal choice for our needs. Neo4j offers powerful querying capabilities through its Cypher language, making it possible to intuitively extract complex relationships and patterns from our dataset.

In conclusion, the choice of a graph database, and Neo4j in particular, aligns perfectly with our dataset's features and our project goals. The flexibility, efficiency, and intuitive nature of Neo4j enable us to delve deeper into the dataset, uncovering valuable insights that would be difficult to extract using other database technologies.

2 | Dataset

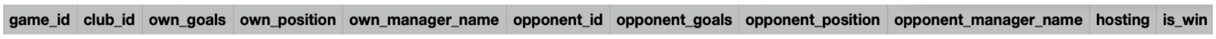
You can find the dataset used in the project from this link.

The dataset has 9 CSV files, each file can be demonstrated with the following tables:
Note: The following figures contain only the column names of the respective csv files.



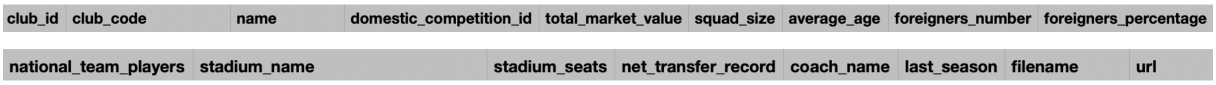
appearance_id	game_id	player_id	player_club_id	player_current_club_id	date	player_name	competition_id	yellow_cards	red_cards	goals	assists	minutes_played

Figure 2.1: Columns of appearances.csv



game_id	club_id	own_goals	own_position	own_manager_name	opponent_id	opponent_goals	opponent_position	opponent_manager_name	hosting	is_win

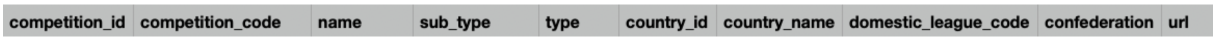
Figure 2.2: Columns of club_games.csv



club_id	club_code	name	domestic_competition_id	total_market_value	squad_size	average_age	foreigners_number	foreigners_percentage

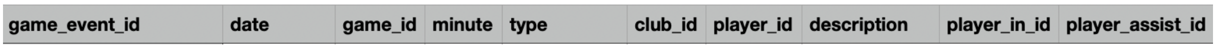
national_team_players	stadium_name	stadium_seats	net_transfer_record	coach_name	last_season	filename	url

Figure 2.3: Columns of clubs.csv



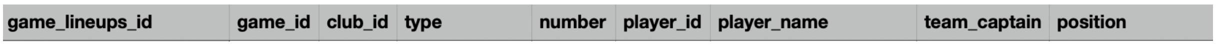
competition_id	competition_code	name	sub_type	type	country_id	country_name	domestic_league_code	confederation	url

Figure 2.4: Columns of competitions.csv



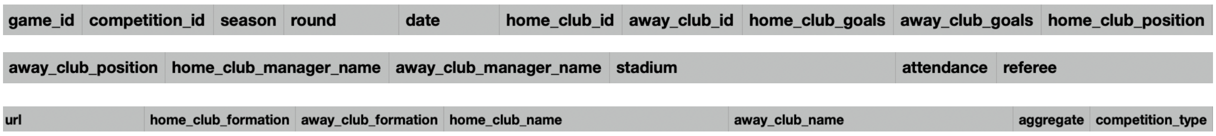
game_event_id	date	game_id	minute	type	club_id	player_id	description	player_in_id	player_assist_id

Figure 2.5: Columns of game_events.csv



game_lineups_id	game_id	club_id	type	number	player_id	player_name	team_captain	position

Figure 2.6: Columns of game_lineups.csv

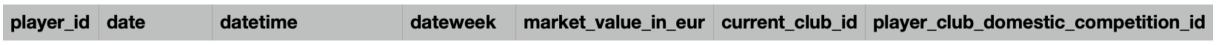


game_id	competition_id	season	round	date	home_club_id	away_club_id	home_club_goals	away_club_goals	home_club_position

away_club_position	home_club_manager_name	away_club_manager_name	stadium	attendance	referee

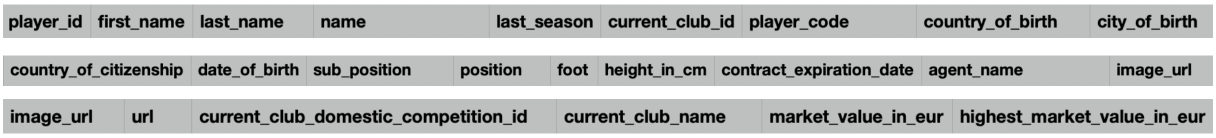
url	home_club_formation	away_club_formation	home_club_name	away_club_name	aggregate	competition_type

Figure 2.7: Columns of games.csv



player_id	date	datetime	dateweek	market_value_in_eur	current_club_id	player_club_domestic_competition_id

Figure 2.8: Columns of player_valuations.csv



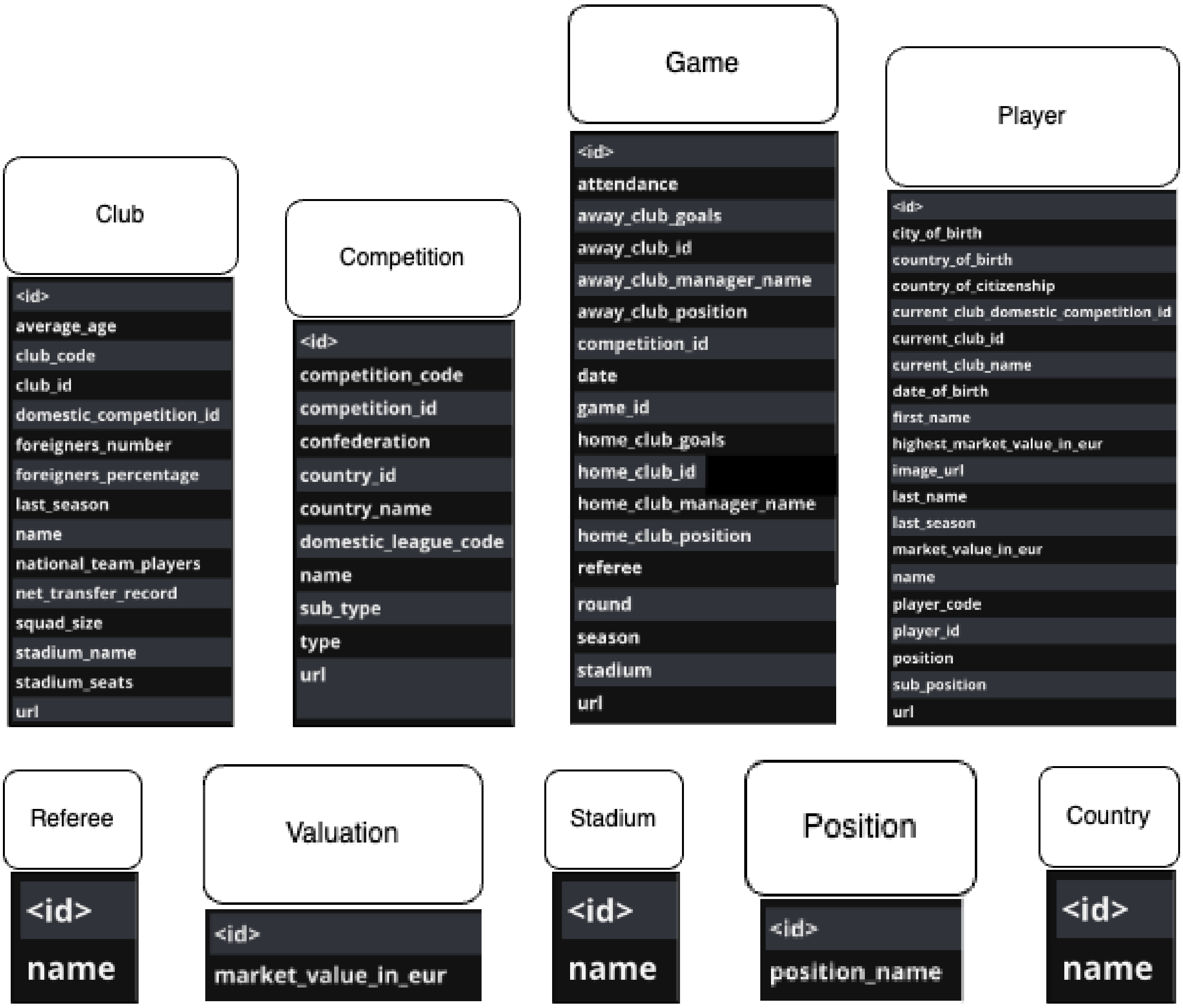
player_id	first_name	last_name	name	last_season	current_club_id	player_code	country_of_birth	city_of_birth

country_of_citizenship	date_of_birth	sub_position	position	foot	height_in_cm	contract_expiration_date	agent_name	image_url

image_url	url	current_club_domestic_competition_id	current_club_name	market_value_in_eur	highest_market_value_in_eur

Figure 2.9: Columns of players.csv

Using these tables, we decided to create the following entities with respective attributes using Python scripts:



Club
<id>
average_age
club_code
club_id
domestic_competition_id
foreigners_number
foreigners_percentage
last_season
name
national_team_players
net_transfer_record
squad_size
stadium_name
stadium_seats
url

Competition
<id>
competition_code
competition_id
confederation
country_id
country_name
domestic_league_code
name
sub_type
type
url

Game
<id>
attendance
away_club_goals
away_club_id
away_club_manager_name
away_club_position
competition_id
date
game_id
home_club_goals
home_club_id
home_club_manager_name
home_club_position
referee
round
season
stadium
url

Player
<id>
city_of_birth
country_of_birth
country_of_citizenship
current_club_domestic_competition_id
current_club_id
current_club_name
date_of_birth
first_name
highest_market_value_in_eur
image_url
last_name
last_season
market_value_in_eur
name
player_code
player_id
position
sub_position
url

Referee
<id>
name

Valuation
<id>
market_value_in_eur

Stadium
<id>
name

Position
<id>
position_name

Country
<id>
name

Figure 2.10: Entities and their attributes.

Please note that all entities and relationships have <*id*> attribute. Since it is common to all different type of entities and relationships, it will not be included in the tables below. <*id*> has type String.

Club

Attribute	**Type**	**Description**
average_age	Float	The average age of the club's players.
club_code	String	The code representing the club.
club_id	Integer	Unique identifier for the club.
domestic_competition_id	Integer	Identifier for the club's domestic competition.
foreigners_number	Integer	Number of foreign players in the club.
foreigners_percentage	Float	Percentage of foreign players in the club.
last_season	Integer	The last season the club was active.
name	String	Full name of the club.
national_team_players	Integer	Number of players in the club who are also national team players.
net_transfer_record	Integer	Net amount spent on transfers (in EUR).
squad_size	Integer	Total number of players in the club.
stadium_name	String	Name of the club's stadium.
stadium_seats	Integer	Number of seats in the club's stadium.
url	String	URL to the club's official website.

Competition

Attribute	**Type**	**Description**
competition_code	String	A unique code assigned to the competition.
competition_id	Integer	The unique identifier for the competition.
confederation	String	The confederation to which the competition belongs.
country_id	Integer	The identifier for the country where the competition is held.
country_name	String	The name of the country where the competition is held.
domestic_league_code	String	A unique code representing the domestic league of the competition.
name	String	The official name of the competition.
sub_type	String	Subcategory or type of the competition if any.
type	String	The type of competition, e.g., league, cup.
url	String	The URL to the competition's official webpage.

Game

Attribute	Type	Description
attendance	Integer	The number of people who attended the game.
away_club_goals	Integer	The number of goals scored by the away club.
away_club_id	Integer	Unique identifier for the away club.
away_club_manager_name	String	The name of the manager of the away club.
away_club_position	String	The league position of the away club at the time of the game.
competition_id	Integer	Identifier for the competition in which the game is played.
date	Date	The date on which the game was played.
game_id	Integer	Unique identifier for the game.
home_club_goals	Integer	The number of goals scored by the home club.
home_club_id	Integer	Unique identifier for the home club.
home_club_manager_name	String	The name of the manager of the home club.
home_club_position	String	The league position of the home club at the time of the game.
referee	String	The name of the referee for the game.
round	String	The round of the competition in which the game took place.
season	String	The season during which the game was played.
stadium	String	The stadium where the game was played.
url	String	The URL to the webpage with details about the game.

Position

Attribute	Type	Description
position_name	String	The name of the position.

Country

Attribute	Type	Description
name	String	The name of the country.

Player

Attribute	Type	Description
city_of_birth	String	The city where the player was born.
country_of_birth	String	The country where the player was born.
country_of_citizenship	String	The country of the player's citizenship.
current_club_domestic_competition_id	Integer	The ID of the current club's domestic competition.
current_club_id	Integer	The ID of the player's current club.
current_club_name	String	The name of the player's current club.
date_of_birth	Date	The player's date of birth.
first_name	String	The player's first name.
highest_market_value_in_eur	Float	The highest market value in EUR for the player.
image_url	String	URL of the player's image.
last_name	String	The player's last name.
last_season	Integer	The last season the player was active.
market_value_in_eur	Float	The current market value in EUR for the player.
name	String	Full name of the player.
player_code	String	A unique code representing the player.
player_id	Integer	Unique identifier for the player.
position	String	The playing position of the player.
sub_position	String	A more specific playing position of the player.
url	String	URL to the player's profile.

Referee

Attribute	Type	Description
name	String	Full name of the referee.

Valuation

Attribute	Type	Description
market_value_in_eur	Integer	The market value in euros.

Stadium

Attribute	Type	Description
name	String	The name of the stadium.

As well as entities, the following relationships are created using Python scripts:

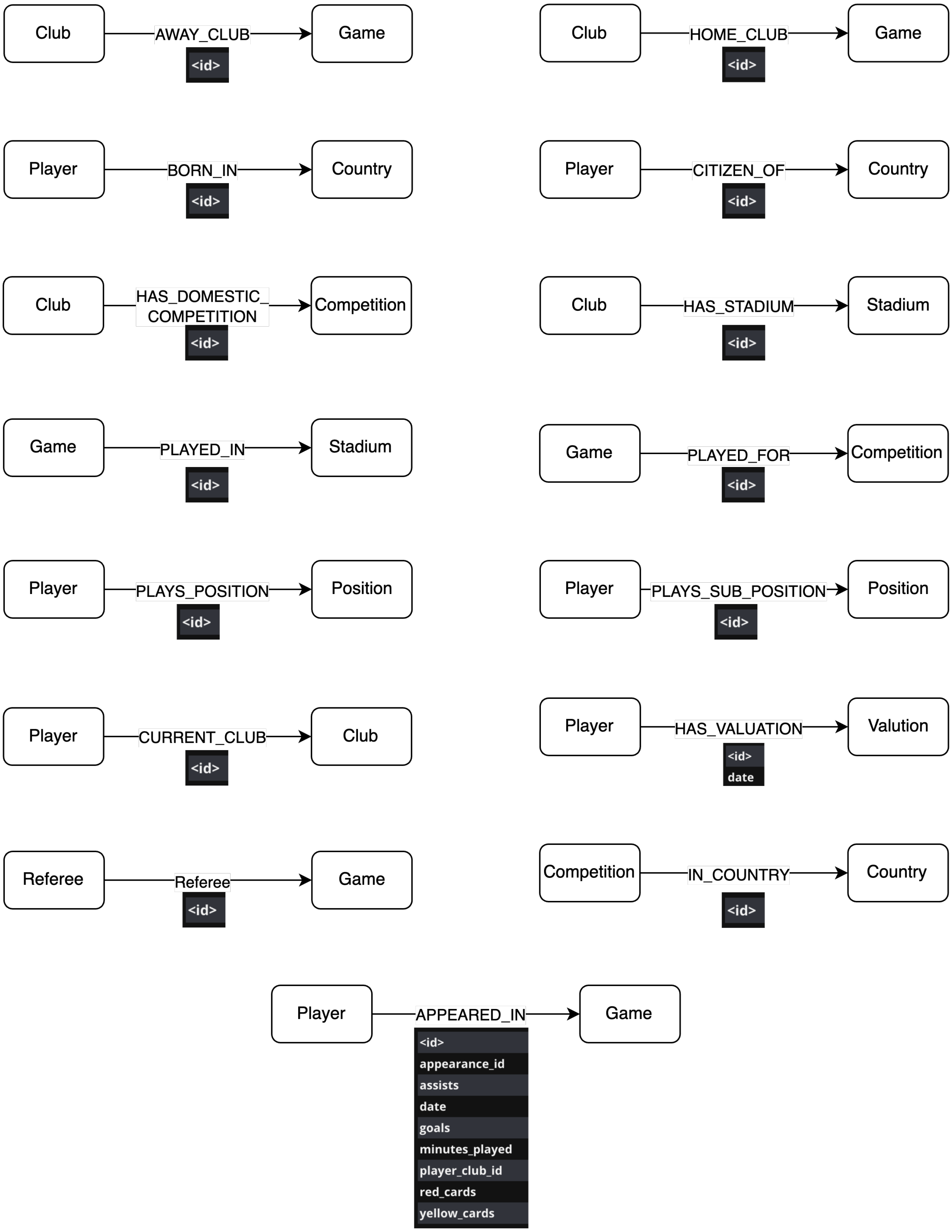


Figure 2.11: Relationships and their attributes.

Except for the APPEARED_IN relationship, all other relationships have only $<id>$ of type String, so their attribute will not be depicted with a table. Below, you can find the most important relationship APPEARED_IN depicted with a table.

APPEARED_IN (Relationship between Player and Game)

Attribute	**Type**	**Description**
appearance_id	Integer	Unique identifier for the appearance entry.
assists	Integer	Number of assists made by the player during the game.
date	Date	The date on which the appearance occurred.
goals	Integer	Number of goals scored by the player during the game.
minutes_played	Integer	Total minutes played by the player in the game.
player_club_id	Integer	Unique identifier of the club the player appeared for.
red_cards	Integer	Number of red cards received by the player during the game.
yellow_cards	Integer	Number of yellow cards received by the player during the game.

As you can see the abundance of relationships, our dataset is very well-suited to be analysed by a graph network. The entities are interconnected to others with important relationships.

3 | Queries

In this chapter, we will look at the queries we have written in detail, one by one.

3.1. Appearances in a Competition

This query returns the players and their appearances grouped by competition and club. Competition Name is input. If he has played in multiple clubs in the given competition, there will be multiple rows for the player. Returns the Player_ID, Player_Name, Club_Name, Appearances, Goals_Scored, Assists, Yellow_Cards, Red_Cards, Minutes_Played.

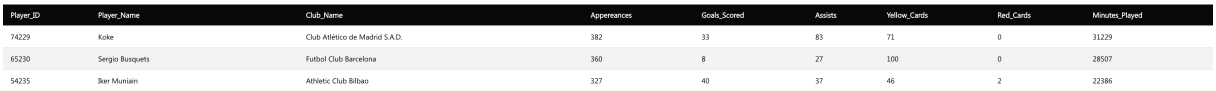
Return format:

```
{
    "Player_ID": 1,
\begin{figure}
        \centering
        \includegraphics[width=0.5\linewidth]{Project Template//Images//query_output/q1.png}
        \caption{Enter Caption}

    \end{figure}
    "Player_Name": "Lionel Messi",
    "Club_Name": "FC Barcelona",
    "Appereances": 10,
    "Goals_Scored": 5,
    "Assists": 3,
    "Yellow_Cards": 2,
    "Red_Cards": 0,
    "Minutes_Played": 900
}
```

Query:

```
MATCH (p:Player)-[r:APPEARED_IN]->(g:Game)-[:HOME_CLUB|AWAY_CLUB]->(c:Club {club_id: r.player_club_id}),
(competition:Competition {name: "laliga"})
WHERE g.competition_id = competition.competition_id and c.name is not null
RETURN p.player_id AS Player_ID, p.name AS Player_Name, c.name AS Club_Name, COUNT(r) AS Appereances, SUM(r.goals) AS Goals_Scored, SUM(r.assists) AS Assists, SUM(r.yellow_cards) AS Yellow_Cards, SUM(r.red_cards) AS Red_Cards, SUM(r.minutes_played) AS Minutes_Played
ORDER BY Appereances DESC
```

Outcome:



Player_ID	Player_Name	Club_Name	Appereances	Goals_Scored	Assists	Yellow_Cards	Red_Cards	Minutes_Played
74229	Koke	Club Atlético de Madrid S.A.D.	382	33	83	71	0	31229
65230	Sergio Busquets	Futbol Club Barcelona	360	8	27	100	0	28507
54235	Iker Muniain	Athletic Club Bilbao	327	40	37	46	2	22386

3.2. Goal and Assist Stats Against a Club

This query returns the players and their goals and assists grouped by clubs. If he has played against multiple clubs, there will be multiple rows for the player. Returns the Player_ID, Player_Name, Own_Club_Name, Against_Club_Name, Goals_Scored, Assists

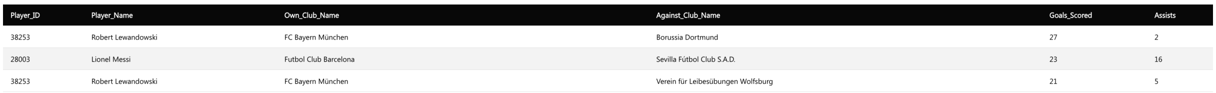
Return format:

```
{
    "Player_ID": 1,
    "Player_Name": "Lionel Messi",
    "Own_Club_Name": "FC Barcelona",
    "Against_Club_Name": "Real Madrid",
    "Goals_Scored": 5,
    "Assists": 3
}
```

Query:

```
MATCH (p:Player)-[r:APPEARED_IN]->(g:Game)-[:HOME_CLUB|AWAY_CLUB]->(c: Club {club_id: r.
    player_club_id}),
(g)-[:HOME_CLUB|AWAY_CLUB]->(against_club: Club)
WHERE against_club.club_id <> c.club_id and c.name is not null and against_club.name is
    not null
WITH p.player_id AS Player_ID, p.name AS Player_Name, c.name AS Own_Club_Name,
    against_club.name AS Against_Club_Name,
SUM(r.goals) AS Goals_Scored, SUM(r.assists) AS Assists
WHERE Goals_Scored > 0
RETURN Player_ID, Player_Name, Own_Club_Name, Against_Club_Name, Goals_Scored, Assists
ORDER BY Goals_Scored DESC, Assists DESC
LIMIT(1000)
```

Outcome:



Player_ID	Player_Name	Own_Club_Name	Against_Club_Name	Goals_Scored	Assists
38253	Robert Lewandowski	FC Bayern München	Borussia Dortmund	27	2
28003	Lionel Messi	Futbol Club Barcelona	Sevilla Fútbol Club S.A.D.	23	16
38253	Robert Lewandowski	FC Bayern München	Verein für Leibesübungen Wolfsburg	21	5

3.3. Two Players Against Each Other

This query returns the players who played against each other the most. Returns the Player_1_ID, Player_1_Name, player_2_id, Player_2_Name, Appereances_Against

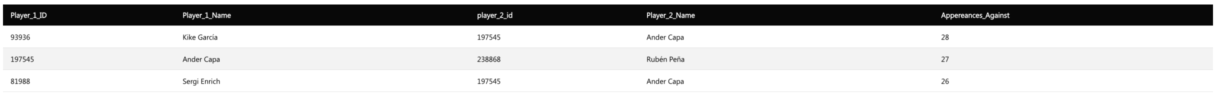
Return format:

```
{
    "Player_1_ID": 1,
    "Player_1_Name": "Lionel Messi",
    "player_2_id": 2,
    "Player_2_Name": "Cristiano Ronaldo",
    "Appereances_Against": 10
}
```

Query:

```
MATCH (p1:Player)-[r1:APPEARED_IN]->(g:Game)<-[r2:APPEARED_IN]-(p2:Player), (competition:
    Competition {name: 'laliga'})
WHERE p1.player_id < p2.player_id and r1.player_club_id <> r2.player_club_id and g.
    competition_id = competition.competition_id
RETURN p1.player_id AS Player_1_ID, p1.name AS Player_1_Name, p2.player_id AS player_2_id
    , p2.name AS Player_2_Name, COUNT(r1) AS Appereances_Against
ORDER BY Appereances_Against DESC
LIMIT(1000)
```

Outcome:



Player_1_ID	Player_1_Name	player_2_id	Player_2_Name	Appereances_Against
93936	Kike García	197545	Ander Capa	28
197545	Ander Capa	238868	Rubén Peña	27
81988	Sergi Enrich	197545	Ander Capa	26

3.4. Average Age in a League

This query returns the leagues and the number of players that have played in the league and the average age of the players in the games played, calculated from the date of the game minus the date of birth of the player. Returns the Competition_ID, Competition_Name, Total_Number_of_Players, Average_Age

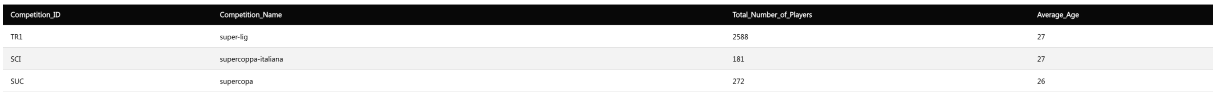
Return format:

```
{
    "Competition_ID": 1,
    "Competition_Name": "La Liga",
    "Total_Number_of_Players": 100,
    "Average_Age": 25
}
```

Query:

```
MATCH (p:Player)-[r:APPEARED_IN]->(g:Game)-[:PLAYED_FOR]->(c:Competition)
WHERE c.name IS NOT NULL
WITH c, p, g, r, duration.between(date(p.date_of_birth), date(g.date)).years AS age
RETURN c.competition_id AS Competition_ID, c.name AS Competition_Name, COUNT(DISTINCT p)
    AS Total_Number_of_Players, toInteger(avg(age)) AS Average_Age
ORDER BY Average_Age DESC
LIMIT(1000)
```

Outcome:



Competition_ID	Competition_Name	Total_Number_of_Players	Average_Age
TR1	super-lig	2588	27
SCI	supercoppa-italiana	181	27
SUC	supercopa	272	26

3.5. Player Wins Against a Club

This query returns the players that got the most wins against a club. The competition name is given by input. Returns the Player_ID, Player_Name, Own_Club_Name, Against_Club_name, Wins, Losses, Draws

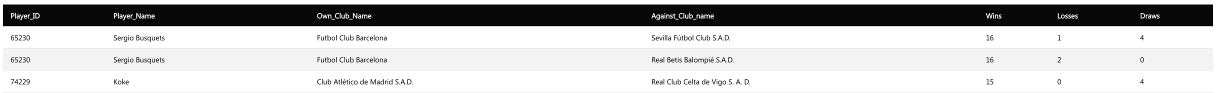
Return format:

```
{
    "Player_ID": 1,
    "Player_Name": "Lionel Messi",
    "Own_Club_Name": "FC Barcelona",
    "Against_Club_name": "Real Madrid",
    "Wins": 10,
    "Losses": 5,
    "Draws": 3
}
```

Query:

```
MATCH (p:Player)-[r:APPEARED_IN]->(g:Game)-[:HOME_CLUB|AWAY_CLUB]->(c:Club {club_id: r.player_club_id}),
(g)-[:HOME_CLUB|AWAY_CLUB]->(against_club: Club),
(competition:Competition {name: 'laliga'})
WHERE g.competition_id = competition.competition_id
RETURN p.player_id AS Player_ID, p.name AS Player_Name, c.name AS Own_Club_Name, against_club.name AS Against_Club_name,
SUM(CASE WHEN g.home_club_goals > g.away_club_goals and c.club_id = g.home_club_id THEN 1
            WHEN g.away_club_goals > g.home_club_goals and c.club_id = g.away_club_id THEN 1
            ELSE 0 END) AS Wins,
SUM(CASE WHEN g.home_club_goals < g.away_club_goals and c.club_id = g.home_club_id THEN 1
            WHEN g.away_club_goals < g.home_club_goals and c.club_id = g.away_club_id THEN 1
            ELSE 0 END) AS Losses,
SUM(CASE WHEN g.home_club_goals = g.away_club_goals THEN 1 ELSE 0 END) AS Draws
ORDER BY Wins DESC, Losses ASC, Draws DESC
LIMIT(1000)
```

Outcome:



Player_ID	Player_Name	Own_Club_Name	Against_Club_name	Wins	Losses	Draws
65230	Sergio Busquets	Futbol Club Barcelona	Sevilla Fútbol Club S.A.D.	16	1	4
65230	Sergio Busquets	Futbol Club Barcelona	Real Betis Balompié S.A.D.	16	2	0
74229	Koke	Club Atlético de Madrid S.A.D.	Real Club Celta de Vigo S. A. D.	15	0	4

3.6. The Fixture with the Most Red Cards

This query returns the pair clubs that played games with the highest red cards. Find red card from the appearances. Returns the Club_1_Name, Club_2_Name, Red_Cards

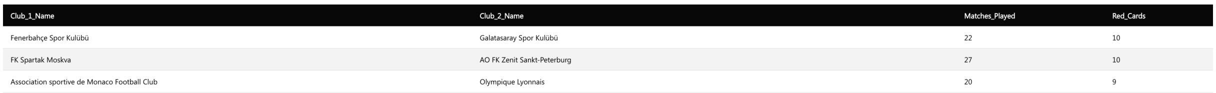
Return format:

```
{
    "Club_1_Name": "FC Barcelona",
    "Club_2_Name": "Real Madrid",
    "Matches_Played": 10,
    "Red_Cards": 10
}
```

Query:

```
MATCH (c1:Club)<-[:HOME_CLUB|AWAY_CLUB]-(g:Game)-[:HOME_CLUB|AWAY_CLUB]->(c2:Club)
WHERE c1.club_id < c2.club_id and c1.name is not null and c2.name is not null
MATCH (p:Player)-[r:APPEARED_IN]->(g)
RETURN c1.name AS Club_1_Name, c2.name AS Club_2_Name, COUNT(DISTINCT g) AS
    Matches_Played, SUM(r.red_cards) AS Red_Cards
ORDER BY Red_Cards DESC, Matches_Played ASC
LIMIT(1000)
```

Outcome:



Club_1_Name	Club_2_Name	Matches_Played	Red_Cards
Fenerbahçe Spor Kulübü	Galatasaray Spor Kulübü	22	10
FK Spartak Moskva	AO FK Zenit Sankt-Peterburg	27	10
Association sportive de Monaco Football Club	Olympique Lyonnais	20	9

3.7. The Games with the Most Expensive Lineups

This query returns the games with most average value of players. Calculates the average value of players in each game. The valuation of players is based on the market value of the player in the game. HAS_VALUATION relationship with the closest and a past date is the real valuation of the player in the game. Returns the Date, Home_Club, Away_Club, Average_Valuation_of_Players.

Return format:

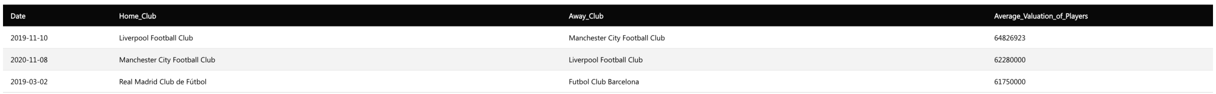
```
 {
    "Date": "FC Barcelona",
    "Home_Club": "Real Madrid",
    "Away_Club": 10,
    "Average_Valuation_of_Players": 10
}
```

Query:

```
MATCH (g:Game)<-[:APPEARED_IN]-(p:Player),
(c_h:Club)<-[:HOME_CLUB]-(g), (c_a:Club)<-[:AWAY_CLUB]-(g)
WHERE c_h.name is not null and c_a.name is not null
WITH g, p, c_h, c_a
CALL {
    WITH g, p
    MATCH (p)-[r:HAS_VALUATION]->(v:Valuation)
    WHERE r.date <= g.date
    RETURN p.player_id AS player_id, g.game_id AS game_id, MAX(r.date) AS
        max_valuation_date
}
WITH g, p, max_valuation_date, c_h, c_a
MATCH (p)-[r:HAS_VALUATION {date: max_valuation_date}]->(v:Valuation)
```

```
RETURN g.date AS Date, c_h.name AS Home_Club, c_a.name AS Away_Club, ROUND(AVG(v.market_value_in_eur)) AS Average_Valuation_of_Players
ORDER BY Average_Valuation_of_Players DESC
LIMIT(1000)
```

Outcome:



Date	Home_Club	Away_Club	Average_Valuation_of_Players
2019-11-10	Liverpool Football Club	Manchester City Football Club	64826923
2020-11-08	Manchester City Football Club	Liverpool Football Club	62280000
2019-03-02	Real Madrid Club de Fútbol	Futbol Club Barcelona	61750000

3.8. The Referee-Club Pairs with Most Bookings

This query returns the referees with their most booked clubs. If the referee has booked multiple clubs, there will be multiple rows for the referee. Returns the Referee_Name, Club_Name, Yellow_Cards, Red_Cards

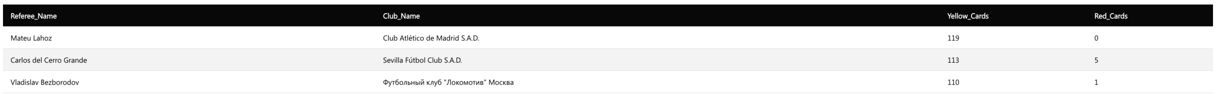
Return format:

```
{
    "Referee_Name": "Mateu Lahoz",
    "Club_Name": "FC Barcelona",
    "Yellow_Cards": 10,
    "Red_Cards": 5
}
```

Query:

```
MATCH (r:Referee)<-[:REFEREE]-(g:Game)<-[ap:APPEARED_IN]-(p:Player), (c:Club)
WHERE ap.player_club_id = c.club_id
WITH r, c, SUM(ap.yellow_cards) AS totalYellow, SUM(ap.red_cards) AS totalRed
order by totalYellow+totalRed desc
WITH r, COLLECT({clubName: c.name, yellowCards: totalYellow, redCards: totalRed})[0] AS mostBookedClub
RETURN r.name AS Referee_Name, mostBookedClub.clubName AS Club_Name, mostBookedClub.yellowCards AS Yellow_Cards, mostBookedClub.redCards AS Red_Cards
order by Yellow_Cards + Red_Cards DESC, Red_Cards DESC, Yellow_Cards DESC
LIMIT(1000)
```

Outcome:



Referee_Name	Club_Name	Yellow_Cards	Red_Cards
Mateu Lahoz	Club Atlético de Madrid S.A.D.	119	0
Carlos del Cerro Grande	Sevilla Fútbol Club S.A.D.	113	5
Vladislav Bezborodov	Футбольный клуб "Локомотив" Москва	110	1

3.9. Goal and Assist Contribution per Minute

This query returns the players with the lowest minutes played for a goal/assist, in a competition, per season of course. Only players with more than 450 minutes played are considered. (roughly 5 matches) Returns the Season, Competition_ID, Player_Name, Minutes_Per_Contibution, Total_Minutes.

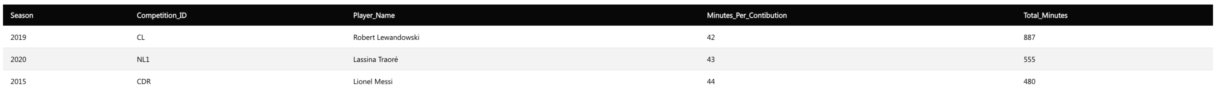
Return format:

```
{
    "Season": "2019",
    "Competition_ID": "CL",
    "Player_Name": "Rpbert Lewandowski",
    "Minutes_Per_Contibution": 42,
    "Total_Minutes": 450
}
```

Query:

```
MATCH (g:Game)<-[ap:APPEARED_IN]-(p:Player), (comp:Competition)
WHERE g.competition_id = comp.competition_id
WITH g.season AS Season, comp.competition_id AS CompetitionID, p, SUM(ap.goals) AS
    TotalGoals, SUM(ap.assists) AS TotalAssists, SUM(ap.minutes_played) AS TotalMinutes
WHERE (TotalGoals + TotalAssists)>0 AND TotalMinutes>450
WITH Season, CompetitionID, p, TotalGoals, TotalAssists, TotalMinutes, (TotalMinutes *
    1.0) / ((TotalGoals + TotalAssists) * 1.0) AS MinutesPerGA
ORDER BY MinutesPerGA ASC
RETURN Season, CompetitionID as Competition_ID, p.name AS Player_Name, ROUND(MinutesPerGA
    ) as Minutes_Per_Contibution,TotalMinutes as Total_Minutes
LIMIT(1000)
```

Outcome:



Season	Competition_ID	Player_Name	Minutes_Per_Contibution	Total_Minutes
2019	CL	Robert Lewandowski	42	887
2020	NL1	Lassina Traoré	43	555
2015	CDR	Lionel Messi	44	480

3.10. Players Carrying the Clubs

This query returns the local heroes. Players that have the highest number of goals / team's goals in competition and season. Returns the Player_Name, Club_Name, Competition_Name, Season, Goals_Scored_By_Player, Total_Club_Goals, Goals_Scored, Ratio

Return format:

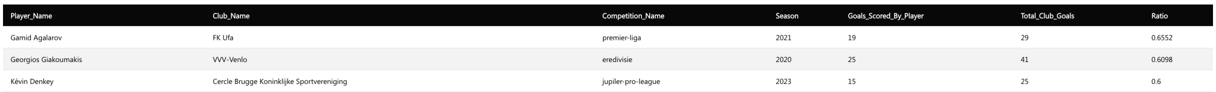
```
{
    "Player_Name": "Lionel Messi",
    "Club_Name": "FC Barcelona",
    "Competition_Name": "La Liga",
    "Season": "2019",
    "Goals_Scored_By_Player": 30,
    "Total_Club_Goals": 40,
    "Ratio": 0.75
}
```

Query:

```
MATCH (comp:Competition),
      (g:Game)-[:PLAYED_FOR]->(comp),
      (c:Club)<-[:HOME_CLUB|AWAY_CLUB]-(g)
where c.name is not null
WITH comp, c, g
```

```
// Calculate total goals for each club in the competition and season
WITH comp, c, g.season as season,
     SUM(CASE
           WHEN g.home_club_id = c.club_id THEN g.home_club_goals
           WHEN g.away_club_id = c.club_id THEN g.away_club_goals
           ELSE 0
         END) AS club_total_goals
// Match players and their goals in the same competition and season
WHERE club_total_goals>20
MATCH (p:Player)-[ap:APPEARED_IN]->(g:Game {season:season})-[:PLAYED_FOR]->(comp),
      (g)-[:HOME_CLUB|AWAY_CLUB]->(c {club_id: ap.player_club_id})
RETURN p.name as Player_Name,
       c.name as Club_Name,
       comp.name as Competition_Name,
       season as Season,
       SUM(ap.goals) AS Goals_Scored_By_Player,
       club_total_goals as Total_Club_Goals,
       ROUND(toFloat(SUM(ap.goals))/club_total_goals, 4) as Ratio
ORDER BY Ratio DESC
LIMIT(1000)
```

Outcome:



Player_Name	Club_Name	Competition_Name	Season	Goals_Scored_By_Player	Total_Club_Goals	Ratio
Gamid Agalarov	FK Ufa	premier-liga	2021	19	29	0.6552
Georgios Giakoumakis	VVV-Venlo	eredivisie	2020	25	41	0.6098
Kévin Denkey	Cercle Brugge Koninklijke Sportvereniging	jupiler-pro-league	2023	15	25	0.6

3.11. Given Country's Top Scorer

This query returns the players with the most goal+assists for a given country and given age range. Returns the Country_Name, Player_Name, Goals_Scored, Assists, Minutes_Played

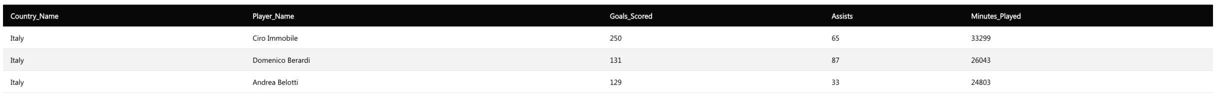
Return format:

```
{
    "Country_Name": "Argentina",
    "Player_Name": "Lionel Messi",
    "Goals_Scored": 30,
    "Assists": 20,
    "Minutes_Played": 4500
}
```

Query:

```
MATCH (p:Player)-[ap:APPEARED_IN]->(g:Game),
      (p)-[:CITIZEN_OF]->(c:Country)
WHERE c.country_name = 'Italy'
WITH p, ap, g, c,
duration.between(date(p.date_of_birth), date(g.date)).years AS age
WHERE age >= 15 AND age <= 40
RETURN c.country_name AS Country_Name, p.name AS Player_Name, SUM(ap.goals) AS
    Goals_Scored, SUM(ap.assists) AS Assists, SUM(ap.minutes_played) AS Minutes_Played
ORDER BY Goals_Scored DESC, Assists DESC, Minutes_Played ASC
LIMIT(1000)
```

Outcome:



Country_Name	Player_Name	Goals_Scored	Assists	Minutes_Played
Italy	Ciro Immobile	250	65	33299
Italy	Domenico Berardi	131	87	26043
Italy	Andrea Belotti	129	33	24803

3.12. Manager Success with respect to Points Gained and Player Valuations

Rate of valuations of players in games to points they get in the game, grouped by each manager. This query calculates the success of the manager. The higher the score, the better the manager. The score is calculated as follows:

$$\text{Score} = \frac{\text{Average Point}}{\log(\text{Average Valuation})} \times 65$$

The reason why we used *log* is because Valuation is so much larger than Point, and we wanted to nerf its contribution to the score calculation. The reason to multiply by 65 is to scale it such that max score will be close to 10.

Group by manager name, sum of points from score, sum of valuation at that time.

Returns the Manager, Average_Point, Matches_Played, Average_Valuation, Score

Return format:

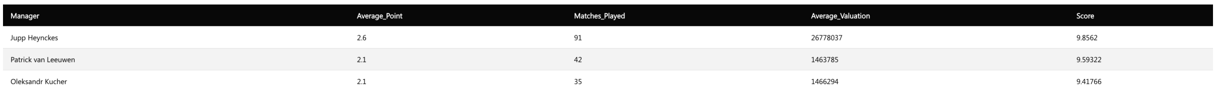
```
{
    "Manager": "Lionel Messi",
    "Average_Point": 1.5,
    "Matches_Played": 100,
    "Average_Valuation": 1000000000,
    "Score": 0.5
}
```

Query:

```
MATCH (g:Game)<-[a:APPEARED_IN]-(p:Player)
MATCH (c_h:Club)<-[:HOME_CLUB]-(g), (c_a:Club)<-[:AWAY_CLUB]-(g)
WHERE c_h.name IS NOT NULL AND c_a.name IS NOT NULL
WITH g, p, c_h, c_a,a

// Find the latest valuation for each player before the game
CALL {
    WITH g, p
    MATCH (p)-[r:HAS_VALUATION]->(v:Valuation)
    WHERE r.date <= g.date
    RETURN p.player_id AS player_id, g.game_id AS game_id, MAX(r.date) AS
        max_valuation_date
}

WITH g, p, max_valuation_date, c_h, c_a, a
```

```
MATCH (p)-[r:HAS_VALUATION {date: max_valuation_date}]->(v:Valuation)

// Aggregate and calculate average valuation for home and away teams
WITH g,
     c_h.name AS home_club,
     c_a.name AS away_club,
     SUM(CASE WHEN a.player_club_id = c_h.club_id THEN v.market_value_in_eur ELSE 0 END) AS home_valuation,
     sum(CASE WHEN a.player_club_id = c_h.club_id THEN 1 ELSE 0 END) as home_player_count,
     SUM(CASE WHEN a.player_club_id = c_a.club_id THEN v.market_value_in_eur ELSE 0 END) AS away_valuation,
     sum(CASE WHEN a.player_club_id = c_a.club_id THEN 1 ELSE 0 END) as away_player_count
WITH g.date AS game_date,
       home_club,
       g.home_club_manager_name as home_manager,
       CASE WHEN g.home_club_goals>g.away_club_goals THEN 3 WHEN g.home_club_goals<g.away_club_goals THEN 0 ELSE 1 END as home_club_point,
       away_club,
       g.away_club_manager_name as away_manager,
       CASE WHEN g.home_club_goals<g.away_club_goals THEN 3 WHEN g.home_club_goals>g.away_club_goals THEN 0 ELSE 1 END as away_club_point,
       home_valuation,
       home_player_count,
       away_valuation,
       away_player_count
WITH home_manager, home_club_point, CASE WHEN home_player_count > 0 THEN toFloat(home_valuation)/home_player_count else NULL END as home_avg_valuation,
away_manager, away_club_point, CASE WHEN away_player_count > 0 THEN
toFloat(away_valuation)/away_player_count else NULL END as away_avg_valuation
WITH [{manager: home_manager, point: home_club_point, avg_valuation: home_avg_valuation},
     {manager: away_manager, point: away_club_point, avg_valuation: away_avg_valuation}] AS managerList
UNWIND managerList AS managerData
WITH managerData.manager AS manager, managerData.point AS point, managerData.avg_valuation as average_valuation
where average_valuation is not null
WITH  manager, avg(point) as avg_point, count(point) as Matches_Played, avg(average_valuation) as avg_valuation
WHERE Matches_Played > 30
return manager as Manager, Round(avg_point,1) as Average_Point, Matches_Played as Matches_Played, ROUND(avg_valuation) as Average_Valuation, ROUND(avg_point/log(avg_valuation)*65, 5) as Score
order by Score desc
limit(1000)
```

Outcome:



Manager	Average_Point	Matches_Played	Average_Valuation	Score
Jupp Heynckes	2.6	91	26778037	9.8562
Patrick van Leeuwen	2.1	42	1463785	9.59322
Oleksandr Kucher	2.1	35	1466294	9.41766

3.13. Average Goal per Game in Each Country

This query returns the countries with most average goal per matches played year by year. Calculated by (Competition)-[:IN_COUNTRY]->(Country) relationship. Input is year. Season attribute of Game node is used for year. Game node has home club goals and away club goals attributes.

Returns the Country_Name, Goals_Scored, Matches_Played, Average_Goals

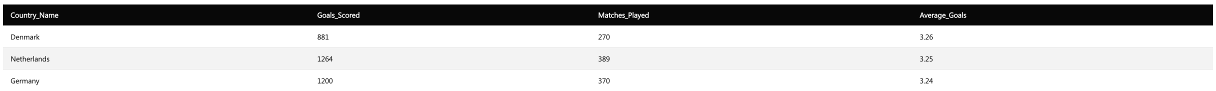
Return format:

```
{
    "Country_Name": "Argentina",
    "Goals_Scored": 10,
    "Matches_Played": 5,
    "Average_Goals": 2
}
```

Query:

```
MATCH (g:Game)-[:PLAYED_FOR]->(c:Competition)-[:IN_COUNTRY]->(co:Country)
WHERE g.season = 2016
WITH co, g
WITH co, count(g) as Matches_Played, sum(g.home_club_goals + g.away_club_goals) as
    Goals_Scored
WHERE Matches_Played > 0
RETURN co.name as Country_Name, Goals_Scored as Goals_Scored, Matches_Played as
    Matches_Played, round(toFloat(Goals_Scored)/Matches_Played, 2) as Average_Goals
ORDER BY Average_Goals DESC, Goals_Scored DESC, Matches_Played ASC
```

Outcome:



Country_Name	Goals_Scored	Matches_Played	Average_Goals
Denmark	881	270	3.26
Netherlands	1264	389	3.25
Germany	1200	370	3.24

3.14. Stadium Aggressiveness with respect to Attendances and Red Cards

This query returns the stadiums according to its heat. Heat is calculated by a Score measure that is:

$$(Average\_Red\_Cards \times 10) + \log_{10}(Average\_Attendance)$$

Retrieved by (p:Player)-[ai:APPEARED_IN]->(g:Game)-[:PLAYED_IN]->(s:Stadium) relationship. ai.red_cards attributes are used for red cards. Input is year. Season attribute of Game node is used for year. Game node has attendance attribute.

Returns the Stadium, Club, Total_Attendance, Matches_Played, Average_Attendance, Red_Cards, Average_Red_Cards, Score

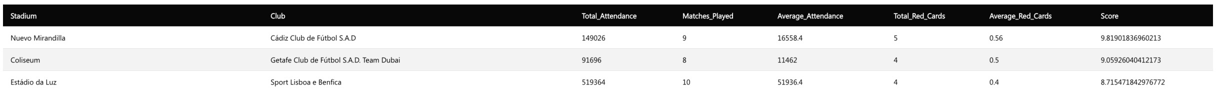
Return format:

```
{
    "Stadium": "Camp Nou",
    "Club": "FC Barcelona",
    "Total_Attendance": 100000,
    "Matches_Played": 10,
    "Average_Attendance": 10000,
    "Red_Cards": 10,
    "Average_Red_Cards": 1,
    "Score": 10
}
```

Query:

```
MATCH (g:Game)-[:PLAYED_IN]->(s:Stadium)<-[:HAS_STADIUM]-(c:Club)
where g.season = 2023
WITH s,c, g
MATCH (p:Player)-[ai:APPEARED_IN]->(g)
WITH s,c, g, sum(ai.red_cards) as Red_Cards
WITH s,c, count(g) as Matches_Played, sum(g.attendance) as total_attendance, sum(
    Red_Cards) as Red_Cards
WHERE Matches_Played > 5
WITH s.name as Stadium, c.name as Club, total_attendance as Total_Attendance,
    Matches_Played as Matches_Played, round(toFloat(total_attendance)/Matches_Played, 1)
    as Average_Attendance, Red_Cards as Red_Cards, round(toFloat(Red_Cards)/
    Matches_Played, 2) as Average_Red_Cards
WHERE Average_Attendance>0
RETURN Stadium, Club, Total_Attendance, Matches_Played, Average_Attendance, Red_Cards,
    Average_Red_Cards,
(Average_Red_Cards*10  + log10(Average_Attendance)) as Score
ORDER BY Score DESC
```

Outcome:



Stadium	Club	Total_Attendance	Matches_Played	Average_Attendance	Total_Red_Cards	Average_Red_Cards	Score
Nuevo Mirandilla	Cádiz Club de Fútbol S.A.D	149026	9	16558.4	5	0.56	9.81901836960213
Coliseum	Getafe Club de Fútbol S.A.D. Team Dubai	91696	8	11462	4	0.5	9.05926040412173
Estádio da Luz	Sport Lisboa e Benfica	519364	10	51936.4	4	0.4	8.715471842976772

3.15. Average Points Gained by Champions in Leagues

This query returns the average point needed to be champion in a competition. Returns the Competition, Average_Championship_Point

Return format:

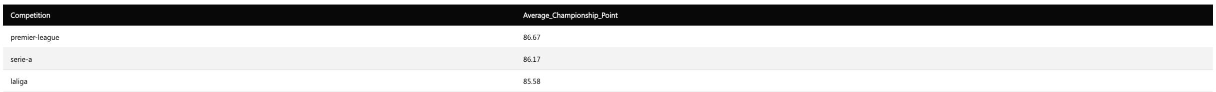
```
{
    "Competition": "La Liga",
    "Average_Championship_Point": 80
}
```

Query:

```
MATCH (comp:Competition)<-[:PLAYED_FOR]-(g:Game),
(c_h:Club)<-[:HOME_CLUB]-(g), (c_a:Club)<-[:AWAY_CLUB]-(g)
```

```
WHERE c_h.name IS NOT NULL AND c_a.name IS NOT NULL and EXISTS(()-[:HAS_DOMESTIC_COMPETITION]->(comp))
WITH g, comp.name as comp,
     c_h.name AS home_club,
     c_a.name AS away_club
WITH g.date AS game_date, g.season as season, comp,
       home_club,
       CASE WHEN g.home_club_goals>g.away_club_goals THEN 3 WHEN g.home_club_goals<g.away_club_goals THEN 0 ELSE 1 END as home_club_point,
       away_club,
       CASE WHEN g.home_club_goals<g.away_club_goals THEN 3 WHEN g.home_club_goals>g.away_club_goals THEN 0 ELSE 1 END as away_club_point
WITH [{name: home_club, point: home_club_point, season: season, comp:comp},
     {name: away_club, point: away_club_point, season: season,comp:comp}] AS clubList
UNWIND clubList as club
with club.season as season, club.comp as competition, club.name as club_name, club.point as point
with season, competition, club_name, sum(point) as total_point
order by total_point desc
with distinct season, competition, collect(club_name) as clubs, collect(total_point) as points
WITH season, competition, clubs[0] as club, points[0] as point
RETURN competition as Competition, round(avg(point),2) as Average_Championship_Point
order by Average_Championship_Point desc
```

Outcome:



Competition	Average_Championship_Point
premier-league	86.67
serie-a	86.17
laliga	85.58

3.16. Average Goal Difference for Each Manager

This query returns the average goal difference for each coach. Returns the Manager, Average_Goal_Difference, Matches_Played

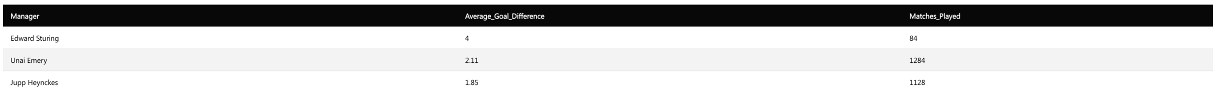
Return format:

```
{
    "Manager": "Lionel Messi",
    "Average_Goal_Difference": 1.5,
    "Matches_Played": 100
}
```

Query:

```
MATCH (g:Game)<-[a:APPEARED_IN]-(p:Player)
MATCH (c_h:Club)<-[:HOME_CLUB]-(g), (c_a:Club)<-[:AWAY_CLUB]-(g)
WHERE c_h.name IS NOT NULL AND c_a.name IS NOT NULL and g.season = 2017
WITH g, p, c_h, c_a,a
// Find the latest valuation for each player before the game
CALL {
    WITH g, p
    MATCH (p)-[r:HAS_VALUATION]->(v:Valuation)
    WHERE r.date <= g.date
    RETURN p.player_id AS player_id, g.game_id AS game_id, MAX(r.date) AS
        max_valuation_date
}

WITH g, p, max_valuation_date, c_h, c_a, a
MATCH (p)-[r:HAS_VALUATION {date: max_valuation_date}]->(v:Valuation)

// Aggregate and calculate average valuation for home and away teams
WITH g,
     c_h.name AS home_club,
     c_a.name AS away_club
WITH g.date AS game_date,
       home_club,
       g.home_club_manager_name as home_manager,
       g.home_club_goals as home_club_goals,
       away_club,
       g.away_club_manager_name as away_manager,
       g.away_club_goals as away_club_goals
WITH [{manager: home_manager, goals_scored: home_club_goals, goals_conceded:
    away_club_goals},
     {manager: away_manager, goals_scored: away_club_goals, goals_conceded:
         home_club_goals}] AS managerList
UNWIND managerList AS managerData
WITH managerData.manager AS manager, managerData.goals_scored AS goals_scored  ,
    managerData.goals_conceded as goals_conceded
WITH goals_scored - goals_conceded as goal_difference, manager
WITH  manager, avg(goal_difference) as average_goal_difference, count(goal_difference) as
     Matches_Played
WHERE Matches_Played > 30
return manager as Manager, round(average_goal_difference, 2) as Average_Goal_Difference,
    Matches_Played as Matches_Played
order by Average_Goal_Difference desc, Matches_Played asc
```

Outcome:



Manager	Average_Goal_Difference	Matches_Played
Edward Sturing	4	84
Unai Emery	2.11	1284
Jupp Heynckes	1.85	1128

3.17. Average Valuation of Players per Country

This query returns the average valuation of players in a given year for each country. Returns the Country_Name, Average_Valuation, Number_of_Players

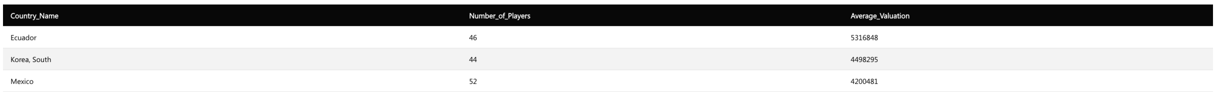
Return format:

```
{
    "Country_Name": "Argentina",
    "Average_Valuation": 1000000,
    "Number_of_Players": 100
}
```

Query:

```
MATCH (p:Player)-[r:HAS_VALUATION]->(v:Valuation)
WITH p, v
WHERE duration.between(date(r.date), date({year:2005, month: 12, day:31})).months < 12
CALL {
    WITH p
    MATCH (p)-[r:HAS_VALUATION]->(v:Valuation)
    RETURN p.player_id AS player_id, MAX(r.date) AS max_valuation_date
}
WITH p, max_valuation_date
MATCH (p)-[r:HAS_VALUATION {date: max_valuation_date}]->(v_n:Valuation)
WITH distinct p, collect(v_n)[0] as v_n
MATCH (p)-[:CITIZEN_OF]->(c:Country)
WITH c, v_n
WITH c.country_name AS country_name, AVG(v_n.market_value_in_eur) AS average_valuation,
    count(v_n) as total_players
WHERE total_players > 5
return country_name as Country_Name, total_players as Number_of_Players, round(
    average_valuation) as Average_Valuation
ORDER BY Average_Valuation  DESC
```

Outcome:



Country_Name	Number_of_Players	Average_Valuation
Ecuador	46	5316848
Korea, South	44	4498295
Mexico	52	4200481

3.18. Goal Ratio of Players: Current Team / Total

This query returns the players with the highest goal ratio for goals scored for the current club and scored in total. Returns the Player_ID, Player_Name, Latest_Club, Goals_For_Latest_Club, Goals_Scored, Goal_Ratio

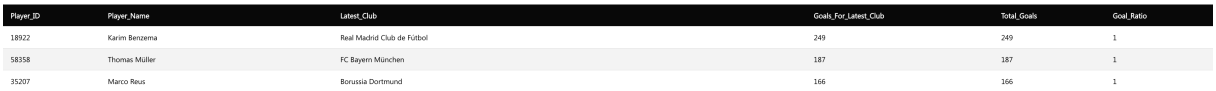
Return format:

```
{
    "Player_ID": 1,
    "Player_Name": "Lionel Messi",
    "Latest_Club": "FC Barcelona",
    "Goals_For_Latest_Club": 190,
    "Goals_Scored": 200,
    "Goal_Ratio": 0.95
}
```

Query:

```
MATCH (p:Player)-[a:APPEARED_IN]->(g:Game)
WITH p, SUM(a.goals) AS Goals_Scored
MATCH (p)-[a_current:APPEARED_IN]->(:Game)
WHERE a_current.player_club_id = p.current_club_id
WITH p, Goals_Scored, SUM(a_current.goals) AS goals_for_current_club
WHERE Goals_Scored > 20 and goals_for_current_club > 0
RETURN p.player_id AS Player_ID,
       p.name AS Player_Name,
       p.current_club_name AS Latest_Club,
        goals_for_current_club as Goals_For_Latest_Club,
        Goals_Scored as Goals_Scored,
        round(goals_for_current_club *1.0 / Goals_Scored, 2) AS Goal_Ratio
ORDER BY Goal_Ratio DESC, Goals_For_Latest_Club DESC
LIMIT(1000)
```

Outcome:

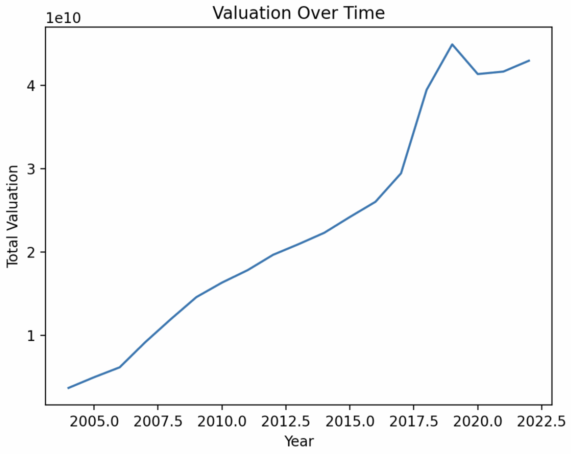


Player_ID	Player_Name	Latest_Club	Goals_For_Latest_Club	Total_Goals	Goal_Ratio
18922	Karim Benzema	Real Madrid Club de Fútbol	249	249	1
58358	Thomas Müller	FC Bayern München	187	187	1
35207	Marco Reus	Borussia Dortmund	166	166	1

3.19. Total Valuation of All Players Changes Over Time

This query calculates the total valuation of all players in the entire dataset in a yearly basis. Returns the year and total valuation in that year.

Using a simple Python script, we plotted a simple graph to see the valuation changes over time.



As you can see from the plot, the valuation of the players across the globe is increasing each and every year except for the COVID-19 era. Even with a simple query and a visualisation, we can see the effects of real world phenomena.

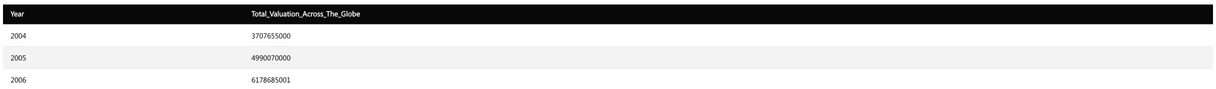
Return format:

```
{
    "Year": 2005,
    "Total_Valuation_Across_The_Globe": 10000000000
}
```

Query:

```
WITH date("2004-01-01") AS startDate, // Start date
     date("2023-12-31") AS endDate // End date
WITH startDate, endDate,
     range(startDate.year, endDate.year) AS years
UNWIND years AS year
WITH date({year: year, month: 1, day: 1}) AS valuationYear

MATCH (p:Player)-[r:HAS_VALUATION]->(v:Valuation)
WITH p, v, valuationYear, r,
     CASE
         WHEN date(r.date).year <= valuationYear.year
         THEN date(r.date).year
         ELSE NULL
     END AS valuationYearMatch
WHERE valuationYearMatch IS NOT NULL
WITH valuationYear, p, MAX(r.date) AS latestValuationDate
MATCH (p)-[r2:HAS_VALUATION {date: latestValuationDate}]->(v2:Valuation)
WITH valuationYear.year AS year, SUM(v2.market_value_in_eur) AS totalValuation
RETURN year as Year, totalValuation as Total_Valuation_Across_The_Globe
ORDER BY Year
```

Outcome:



Year	Total_Valuation_Across_The_Globe
2004	3707655000
2005	4990070000
2006	6178685001

3.20. Games Played per Season for the Given Club

This query calculates the total number of games played per season per club. Returns the season, club name and number of games played.

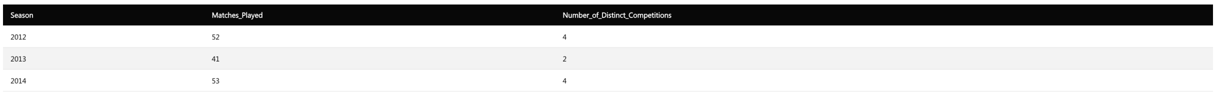
Return format:

```
{
    "Season": 2005,
    "Matches_Played": 50,
    "Number_of_Distinct_Competitions": 3
}
```

Query:

```
MATCH (c:Club {name: 'Liverpool Football Club'})
MATCH (g:Game)
WHERE g.home_club_id = c.club_id OR g.away_club_id = c.club_id
WITH g.season AS season, COUNT(g) AS numberOfGames, COLLECT(DISTINCT g.competition_id) AS
     competitions
RETURN season as Season,
       numberOfGames as Matches_Played,
       SIZE(competitions) AS Number_of_Distinct_Competitions
ORDER BY Season
```

Outcome:



Season	Matches_Played	Number_of_Distinct_Competitions
2012	52	4
2013	41	2
2014	53	4

4 | Extra Work

For better demonstration purposes, we have deployed a webpage that includes the explanations and the outcomes of the queries. You can find the related code from our repository: Football4j. If you are not going to take a look at all 20 query demonstrations, we advise you to look at query 19 with the title *'Prices go up and up! Unless...'*

Besides frontend code, you can find the code that sets up the Neo4j database environment using a pipeline.

Of course, the queries and the supplementary material (figures) used in this documentation can also be found in the repository.

List of Figures