POLITECNICO

MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Cristiano Battistini 244466**

Carlo Cardignan 250325

Group Number: **GroupNumber**

Academic Year: 2023-2024

Contents

1 | Introduction

The advent of data-driven strategies in football has revolutionized the sport, providing insights that drive decisions, from player selection to game tactics. There are indeed many examples of clubs that have scouted top talent by leveraging data analysis. What's more, the millions of fans of the sport love to follow the statistics, to see the analytics of their idols in smartphone apps.

Our project builds on this data-centric approach, leveraging a detailed data set that encapsulates the multifaceted nature of football, including player statistics, club data, and competition history. We chose MongoDB because its non-relational structure is particularly well suited to handle the diversity and volume of data we are dealing with. Unlike traditional relational databases, MongoDB's flexible data model allows us to store and process different types of data without the need for a predefined schema.

MongoDB, being a document-oriented database, allows each player, club or competition to be represented as a document with a rich and dynamic set of attributes. In addition, MongoDB's horizontal scalability through automatic sharding is critical for handling large volumes of data, such as those generated in modern football. The performance of this technology is in fact optimal even as the data size increases.

All of this allows us to dynamically adapt to the evolving nature of the dataset, reflecting the real-world fluidity of football team compositions and league structures, keeping in mind that nowadays there is always a competition game to watch.

Our goal is to extract meaningful patterns and insights that can influence various factors, such as talent scouting but also match analysis. The dataset includes detailed details on players, with their appearances and ratings, club situations, and the competitive landscape of the major leagues.

2 | Data Wrangling/Data Generation

2.1. Chapter Introduction

A cleanup and standardization of football-related data was necessary. This process included replacing long, detailed descriptive strings with one- or two-character abbreviations to make the data more manageable and optimize the size of the database. The operation was done manually using the "find and replace" function of the Visual Studio Code software. Specifically, the "description" attribute within the "game_events" dataset was simplified by removing unnecessary details such as the total number of season goals or the specification of tournament goals, keeping only the essentials such as the type of shot that led to the goal. This process made the data more streamlined and focused on relevant aspects for later analysis. The following tables describe what was changed in the initial datasets.

2.1.1. Players

Original Dataset	Attribute	Old Value	New Value
players	sub_position	Attacking Midfield	AM
players	sub_position	Defensive Midfield	DM
players	sub_position	Goalkeeper	G
players	sub_position	Centre-Forward	CF
players	sub_position	Centre-Back	CB
players	sub_position	Central Midfield	CM
players	sub_position	Left Winger	LW
players	sub_position	Right Winger	RW
players	sub_position	Right-Back	RB
players	sub_position	Left-Back	LB
players	sub_position	Left Midfield	LM
players	sub_position	Right Midfield	RM
players	sub_position	Second Striker	ST
players	position	Defender	D
players	position	Midfield	C
players	position	Attack	A
players	position	Missing	M
players	position	GoalKeeper	G
players	foot	right	R
players	foot	left	L

Table 2.1: Players Data Wrangling

2.1.2. Game events

Original Dataset	Attribute	Old Value	New Value
game_events	description	Right-footed shot	R
game_events	description	Penalty	P
game_events	description	Direct free kick	F
game_events	description	Left-footed shot	L
game_events	description	Header	H
game_events	description	Tap-in	T
game_events	description	Deflected shot on goal	D
game_events	description	Long distance kick	K
game_events	description	Own goal	O
game_events	description	Solo run	S
game_events	description	Counter attack goal	C
game_events	description	Chest	Q
game_events	description	Penalty rebound	B
game_events	description	Direct corner	N

Table 2.2: Game Event Data Wrangling

2.1.3. club games

Original Dataset	Attribute	Old Value	New Value
club_games	hosting	Home	H
club_games	hosting	Away	A

Table 2.3: Clubs Data Wrangling

2.2. Original Data

Initially, the data were distributed in several datasets: 'clubs', 'club_games', 'competitions', 'games', 'game_events', 'player', 'player_valuations' and 'appearances'. To optimize organization and accessibility, a restructuring into three main collections in the 'football' database was adopted. These are the initial datasets and their attributes:

Dataset name	Attributes
players	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, market_value_in_eur, highest_market_value_in_eur, contract_expiration_date, agent_name, image_url, url, current_club_domestic_competition_id, current_club_name
player_valuations	player_id, last_season, datetime, date, dateweek, market_value_in_eur, n, current_club_id, player_club_domestic_competition_id

Dataset name	Attributes
appearances	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
competitions	competition_id, competition_code, name, sub_type, type, country_id, country_name, domestic_league_code, confederation, url, club_games, game_id, club_id, own_goals, own_position, own_manager_name, opponent_id, opponent_goals, opponent_position, opponent_manager_name, hosting, is_win

Dataset name	Attributes
games	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_name,
	away_club_name,
	aggregate,
	competition_type,
	clubs,
	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,
	url

Dataset name	Attributes
game_events	game_id, minute, type, club_id, player_id, description, player_in_id

The 'player' collection now integrates player information with related 'valuations' and 'appearances', thanks to the Python script 'players_complete_info.py'. The 'clubs' collection encapsulates within it the 'club_games', which in turn contain details about 'game_events', aggregated via the 'club_and_games.py' script. Finally, the 'competitions' collection was enriched with the associated 'games' and their respective 'game_events' through the use of the 'competitions_and_games.py' script. In merging multiple datasets, attributes that were repeated were removed both to save space and because they were unnecessary to the context outlined. In the following repository
(at this URL `https://github.com/cristianobattistini/smbud`) it is possible to observe the python code for the updates to the original datasets.
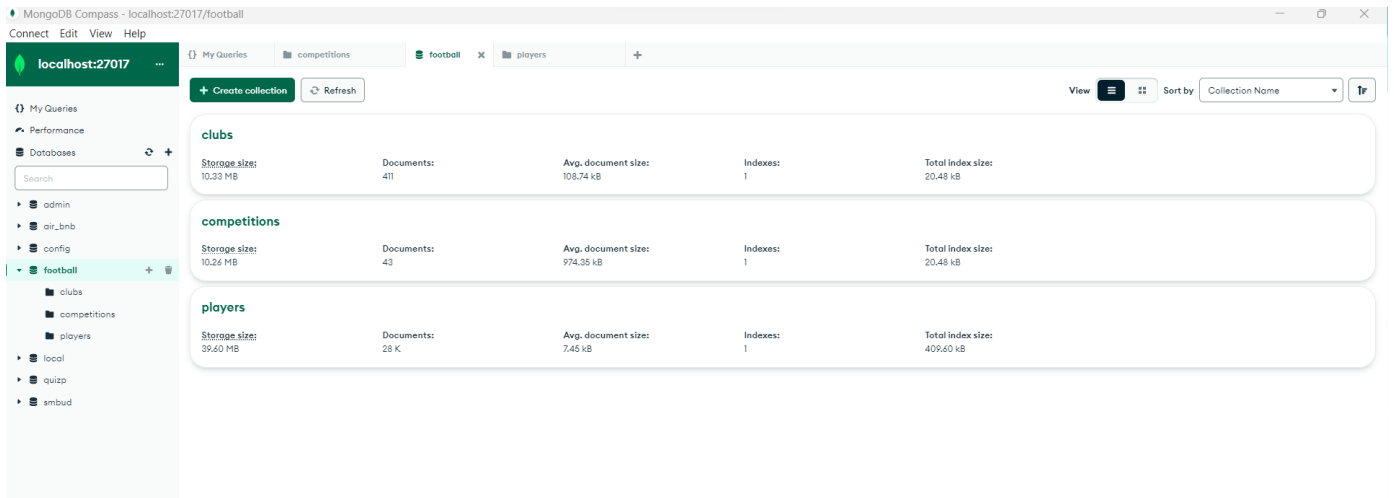
3 | Dataset

3.1. Dataset

The selected dataset is a large collection of football data, derived primarily from Transfermarkt. Updated regularly, it offers accurate data on more than 60,000 global competition matches, details on 400 clubs, and statistics on more than 30,000 players, including current and historical market values, physical characteristics, team membership, and individual performances. More than 1.2 million records detail competitive performances, such as appearances and cards. It is possible to observe the original one, saved in `www.kaggle.com` , at this link: `https://www.kaggle.com/datasets/thedevastator/football-data-competitions-clubs-players-statist` After the Data Wrangling/Data Generation changes, the MongoDB database, called football, has the following collections and statistics:

- 'Clubs' collection: 411 documents, with an average size of 108.74 kB per document. Total size of indexes: 20.48 kB.

- 'Competitions' collection: 43 documents, with an average size of 974.35 kB per document. Total size of indexes: 20.48 kB.

- 'Players' collection: 28,459 documents, with an average size of 7.45 kB per document. Total size of indexes: 409.60 kB.

All the collections contain one or more arrays of sub-dcouments. Some sub-documents contain also other sub-documents.
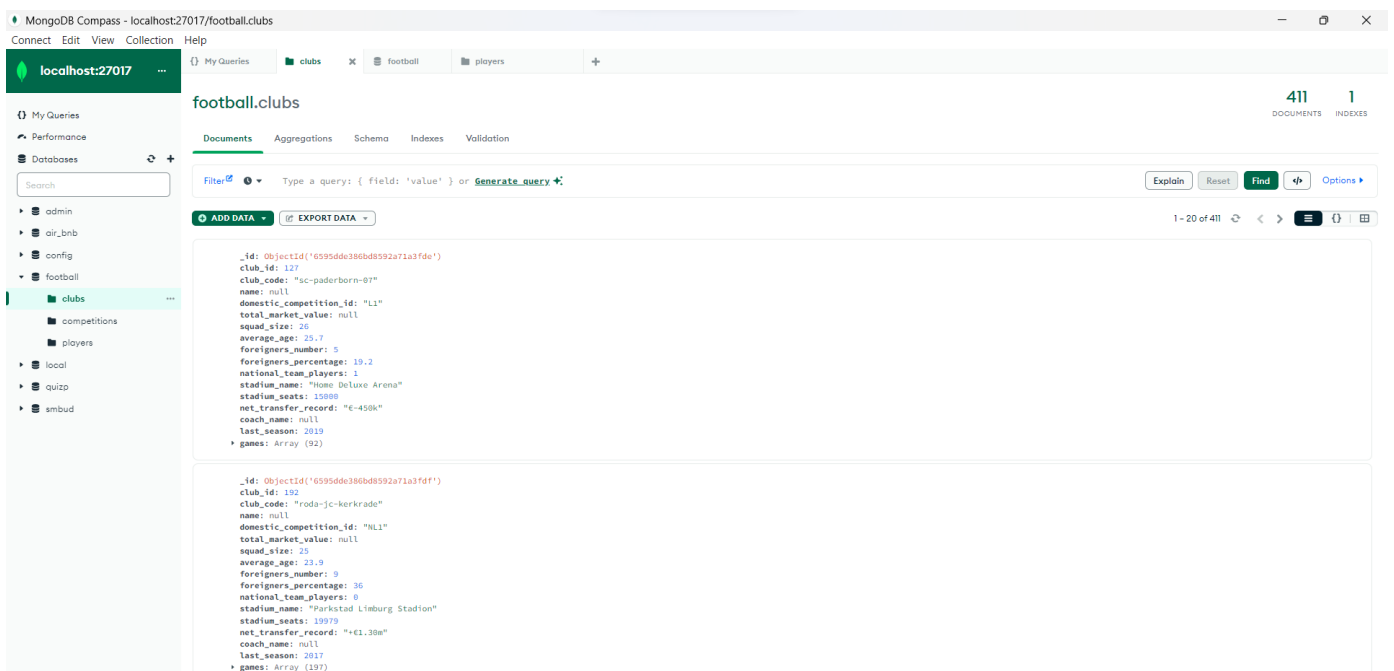
Figure 3.1: Football Dataset

3.2.   Collections

3.2.1.   Clubs

Figure 3.2: Clubs Collection

Attribute	Type	Description
_id	ObjectId	A unique identifier for the document.
club_id	Int32	An integer representing the club's unique ID.
club_code	String	A textual code that uniquely identifies the club.
name	String	The official name of the club.
domestic_competition_id	String	The ID of the domestic league in which the club competes.
total_market_value	Null	The total market value of the club, currently not available.
squad_size	Int32	The number of players in the club's squad.
average_age	Double	The average age of the players in the squad.
foreigners_number	Int32	The count of foreign players in the squad.
foreigners_percentage	Double	The percentage of foreign players relative to the total squad size.
national_team_players	Int32	The number of players who are also national team members.
stadium_name	String	The name of the club's home stadium.
stadium_seats	Int32	The seating capacity of the club's stadium.
net_transfer_record	Null	The net financial record of player transfers, currently not available.
coach_name	String	The name of the club's coach.
last_season	Int32	The most recent season the club competed in.
games	Array (of objects)	A textual code that uniquely identifies the club.

Game Object inside Club

Attribute	Type	Description
game_id	Int32	The unique identifier for the game.
own_goals	Int32	The number of goals scored by the club.
own_position	Int32	The league position of the club at the time of the game.
own_manager_name	String	The name of the club's manager.
opponent_id	Int32	The unique identifier for the opponent club.
opponent_goals	Int32	The number of goals scored by the opponent.
opponent_position	Int32	The league position of the opponent at the time of the game.
opponent_manager_name	String	The name of the opponent's manager.
hosting	String	A character indicating whether the club was hosting the game ('H' for home, 'A' for away).
is_win	Int32	Indicates the outcome of the game (e.g., 0 for loss or draw, 1 for win).
events	Array (of objects)	A list of significant events during the game, with each event as an object containing its own set of attributes.

Events inside Games inside Clubs

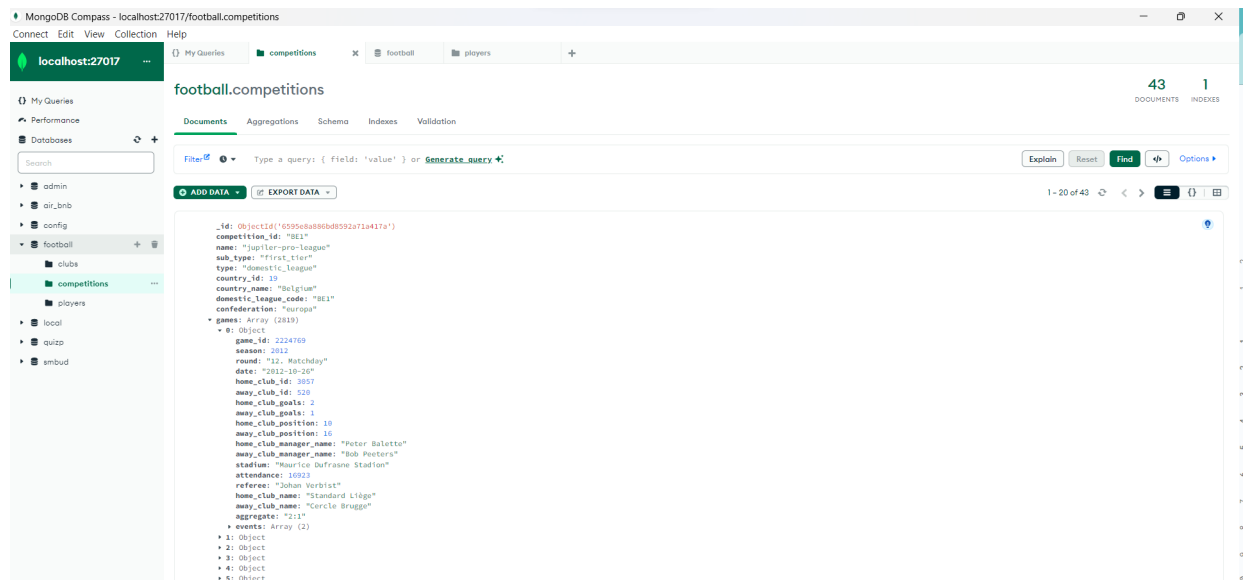| Attribute	Type	Description
minute	Int32	The match time in minutes when the event occurred.
type	String	The category of the event, e.g., "Substitutions" or "Goals"
player_id	Int32	The unique identifier of the player involved in the event.
description	Null/String	A detailed description of the event, if available.
player_in_id	Int32	The unique identifier of the player substituted into the game, relevant for substitution events.

3.2.2.   Competition

Figure 3.3: Competitions Collection

Attribute	Type	Description
_id	ObjectId	A unique identifier for the document.
competition_id	String	An identifier for the competition.
name	String	The official name of the competition.
sub_type	String	A category within the competition, such as "first_tier".
type	String	The nature of the competition, for example, "domestic_league".
country_id	Int32	A numeric identifier for the country associated with the competition.
country_name	String	The name of the country.
domestic_league_cod	String	A unique code representing the domestic league.
confederation	String	The football confederation to which the competition belongs.
games	Array	A collection of game records associated with the competition.

Game Object inside Competition

Attribute	Type	Description
game_id	Int32	The unique identifier for the game.
season	Int32	The year of the football season.
round	String	Distinct round: ['1. Matchday' '3. Matchday' '4. Matchday' '11. Matchday' ...
date	String	When the game was played.
home_club_id	Int32	Identifier for the home club.
away_club_id	Int32	Identifier for the away club.
home_club_goals	Int32	Goals scored by the home club.
away_club_goals	Int32	Goals scored by the away club.
home_club_position	Int32	League position of the home club at game time.
away_club_position	Int32	League position of the away club at game time.
home_club_manager_name	String	Name of the home club's manager.
away_club_manager_name	String	Name of the away club's manager.
stadium	String	Name of the stadium where the game was played.
attendance	Int32	Number of people who attended the game.
referee	String	Name of the referee of the game.
home_club_name	String	Name of the home club.
away_club_name	String	Name of the away club.
aggregate	String	Overall score
events	Array (of objects)	An array detailing significant events during the game.

Events inside Games inside Competions

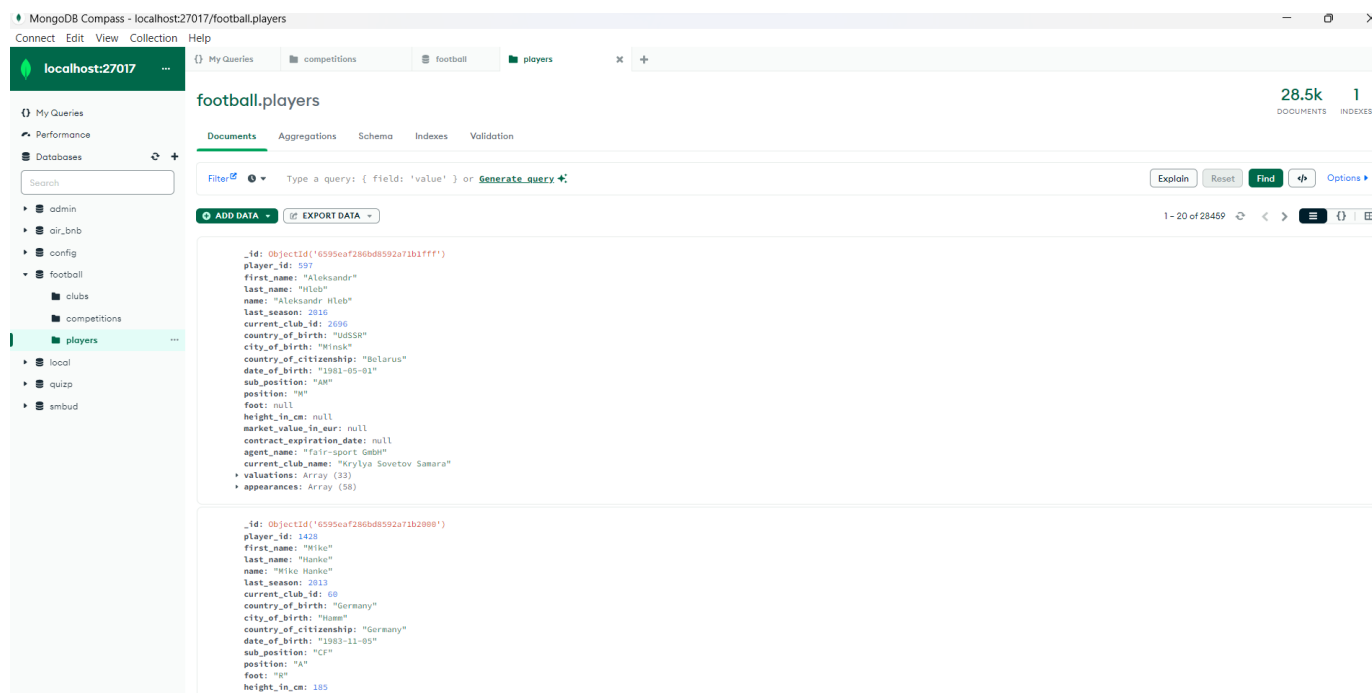| Attribute	Type	Description
minute	Int32	The match time in minutes when the event occurred.
type	String	The category of the event, e.g., "Substitutions" or "Goals"
player_id	Int32	The unique identifier of the player involved in the event.
description	Null/String	A detailed description of the event, if available.
player_in_id	Int32	The unique identifier of the player substituted into the game, relevant for substitution events.

3.2.3. Players

Figure 3.4: Players Collection

Attribute	Type	Description
_id	ObjectId	A unique identifier for the document.
player_id	Int32	A numeric identifier for the player.
first_name	String	The player's first name.
last_name	String	The player's surname.
name	String	The full name of the player.
last_season	Int32	The last active season of the player.
current_club_id	Int32	Identifier for the player's current club.
country_of_birth	String	The country where the player was born.
city_of_birth	String	The city where the player was born.
country_of_citizenship	String	The country of the player's citizenship.
date_of_birth	String	The player's birthdate.
sub_position	String	The player's specific position on the field.
position	String	The general position category the player occupies.
foot	Null/String	Preferred foot of the player.
height_in_cm	Null/Int32	The player's height in centimeters.
market_value_in_eur	Null/Int32	The player's market value in euros.
contract_expiration_date	String	When the player's contract is set to expire.
agent_name	String	The name of the player's agent.
current_club_name	String	The name of the player's current club.
valuations	Array (of objects)	A history of the player's market value evaluations.
appearances	Array (of objects)	A record of the player's appearances in games.

Player Valuations

Attribute	Type	Description
last_season	Int32	The season year of the valuation.
market_value_in_eur	Int32	The player's market value in euros at that time.
current_club_id	Int32	The identifier for the club the player was with during that season.

Player Appearances

Attribute	Type	Description
game_id	Int32	The identifier for the game.
date	String	The date when the game was played.
competition_id	String	The identifier for the competition in which the game took place.
yellow_cards	Int32	The number of yellow cards received by the player in the game.
red_cards	Int32	The number of red cards received by the player in the game.
goals	Int32	The number of goals scored by the player in the game.
assists	Int32	The number of assists made by the player in the game.
minutes_played	Int32	The number of minutes the player played in the game.

4 | Queries

4.1. Leagues analysed

The competitions analyzed will include the Champions League and the major national leagues of Europe: Ligue 1 (France), LaLiga (Spain), Premier League (England), Serie A (Italy), and Bundesliga (Germany). The Champions League is Europe's most prestigious international competition, featuring Europe's best teams. The other national leagues are among the most important in Europe, characterized by top players, significant economic power of the clubs, and a high degree of competitiveness and visibility at the international level.

LEAGUE TYPE	LEAGUE COUNTRY	LEAGUE CODE	LEAGUE NAME	
International (European) League		CL	Champions League	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)
Domestic League	FRANCE	FR1	Ligue1	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)
Domestic League	SPAIN	ES1	LaLiga	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)

LEAGUE TYPE	LEAGUE COUNTRY	LEAGUE CODE	LEAGUE NAME	
Domestic League	ENGLAND	GB1	Premier League	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)
Domestic League	GERMANY	L1	Bundesliga	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)
Domestic League	ITALY	IT1	Serie A	_id: ObjectId('6593e60f0f39fcbfdbd5cbc1') competition_id: "CL" name: "uefa-champions-league" sub_type: "uefa_champions_league" type: "international_cup" country_id: -1 country_name: null domestic_league_code: null confederation: "europa" ▸ games: Array (1360)

4.2. Players Collection

4.2.1. Top Football Agents

This query lists agents according to the number of players they represent in the database, ordered from highest to lowest, clearly showing the most influential agents, or companies, in the world of football. In recent years, the figure of the agent or company, which looks after the interests of players, especially in terms of contracts, has had an enormous increase in power.

The agents or the most important companies manipulate the market trying to profit for the player but also for them: every transfer or contract in fact provides compensation for the agents.

In recent years, many agents have played the big game, cornering many clubs and earning huge amounts of money.

Simply, the query's behavior is to group by the attribute agent_name and then to compute the total players for each agent_name.

```
db.players.aggregate([
    { $match: { "agent_name": { $ne: null } } },
    { $group: { _id: "$agent_name", numberOfPlayers: { $sum: 1 } } },
    { $sort: { numberOfPlayers: -1 } }
  ])
```


Figure 4.1: Top Football Agents Result

4.2.2.   Competitions statistics

Knowing the statistics of players in competitions is certainly important to identify those talents to be acquired during the football market phase. There are many characteristics that can be evaluated for a player: for example, his goals, assists, cards taken or minutes played in a specific competition. There are many competitions around the world, but only the most important ones will be listed: the five top European leagues (England, Spain, France, Italy, Germany) and the Champions League.

Top GoalScorers

The most coveted record or award is surely to become the top scorer in a competition. This query shows the best champions, usually strikers, to have scored the most goals. The query starts with $unwind to break down the appearances array of each player document, then filters the appearances for a given competition_id. Next, $group aggregates the data by player, adding up the goals scored and capturing the player's name. Finally, $sort and $limit sort the players from highest to lowest number of goals and limit the output to the top 10.

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $match: { "appearances.competition_id": "<competition_id>" } },
  {
    $group: {
      _id: "$_id",
      totalGoals: { $sum: "$appearances.goals" },
      playerName: { $first: "$name" }
    }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
])
```

- CL

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "CL" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb0986bd8592a71b5032'),
    totalGoals: 74,
    playerName: 'Robert Lewandowski'
  }
  {
    _id: ObjectId('6595eb2686bd8592a71b85de'),
    totalGoals: 73,
    playerName: 'Cristiano Ronaldo'
  }
  {
    _id: ObjectId('6595eb2386bd8592a71b7f41'),
    totalGoals: 62,
    playerName: 'Lionel Messi'
  }
```

- IT1

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "IT1" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb2386bd8592a71b8077'),
    totalGoals: 165,
    playerName: 'Ciro Immobile'
  }
  {
    _id: ObjectId('6595eb2786bd8592a71b865a'),
    totalGoals: 108,
    playerName: 'Gonzalo Higuaín'
  }
  {
    _id: ObjectId('6595eb1286bd8592a71b5fcc'),
    totalGoals: 106,
    playerName: 'Paulo Dybala'
  }
```

- ES1

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "ES1" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb2386bd8592a71b7f41'),
    totalGoals: 231,
    playerName: 'Lionel Messi'
  }
  {
    _id: ObjectId('6595eafb86bd8592a71b3456'),
    totalGoals: 176,
    playerName: 'Luis Suárez'
  }
  {
    _id: ObjectId('6595eafd86bd8592a71b3ad0'),
    totalGoals: 161,
    playerName: 'Karim Benzema'
  }
```

- FR1

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "FR1" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb1286bd8592a71b60f5'),
    totalGoals: 155,
    playerName: 'Kylian Mbappé'
  }
  {
    _id: ObjectId('6595eb1086bd8592a71b5cf0'),
    totalGoals: 122,
    playerName: 'Edinson Cavani'
  }
  {
    _id: ObjectId('6595eb1486bd8592a71b6505'),
    totalGoals: 111,
    playerName: 'Wissam Ben Yedder'
  }
```

- **GB1**

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "GB1" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eafb86bd8592a71b3748'),
    totalGoals: 203,
    playerName: 'Harry Kane'
  }
  {
    _id: ObjectId('6595eb0986bd8592a71b5200'),
    totalGoals: 134,
    playerName: 'Jamie Vardy'
  }
  {
    _id: ObjectId('6595eb0b86bd8592a71b5629'),
    totalGoals: 133,
    playerName: 'Mohamed Salah'
  }
```

- **L1**

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "L1" } },
    {
      $group: {
        _id: "$_id",
        totalGoals: { $sum: "$appearances.goals" },
        playerName: { $first: "$name" }
      }
    },
    { $sort: { totalGoals: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb0986bd8592a71b5032'),
    totalGoals: 238,
    playerName: 'Robert Lewandowski'
  }
  {
    _id: ObjectId('6595eafb86bd8592a71b3464'),
    totalGoals: 97,
    playerName: 'Andrej Kramaric'
  }
  {
    _id: ObjectId('6595eb2786bd8592a71b87e6'),
    totalGoals: 96,
    playerName: 'Timo Werner'
  }
```

4.2.3. Best Assist-men

Making an assist means putting a teammate in a position to put the ball in the net. This is also a very important statistic, often peculiar to side or full-back defenders, midfielders or wingers. The query starts with $unwind to break down the appearances array of each player document, then filters the appearances for a given competition_id. Next, $group aggregates the data by player, adding up the assists and capturing the player's name. Finally, there is a descending sort and a limit for the best 10 assist-men.

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $match: { "appearances.competition_id": "<competition_id>" } },
  {
    $group: {
      _id: "$_id",
      totalAssists: { $sum: "$appearances.assists" },
      playerName: { $first: "$name" }
    }
  },
  { $sort: { totalAssists: -1 } },
  { $limit: 10 }
])
```


Figure 4.2: Best Assist-men Executed with CL competition

4.2.4.   Players with multiple yellow cards

There are also many players who make impetuousness and confrontation their strong point. The statistics on yellow cards say a lot about those players who exploit their physical strength, but because of this attitude are prone to committing fouls, punishable by yellow cards. The query starts with *unwindtobreakdowntheappearancesarrayofeachplayerdocument, thenfi* aggregates the data by player, adding up the yellow cards taken and showing the player's name.

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $match: { "appearances.competition_id": "<competition_id>" } },
  { $group: { _id: "$_id", totalYellowCards:
+{ $sum: "$appearances.yellow_cards" },  playerName: { $first: "$name" }  } },
  { $sort: { totalYellowCards: -1 } },
  { $limit: 10 }
])
```

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "IT1" } },
    { $group: { _id: "$_id", totalYellowCards: { $sum: "$appearances.yellow_cards" },  playerName: { $first: "$name" }  } },
    { $sort: { totalYellowCards: -1 } },
    { $limit: 10 }
])
< {
    _id: ObjectId('6595eb0986bd8592a71b50ce'),
    totalYellowCards: 79,
    playerName: 'Tomás Rincón'
  }
  {
    _id: ObjectId('6595eb2186bd8592a71b7a2c'),
    totalYellowCards: 74,
    playerName: 'Marcelo Brozovic'
  }
  {
    _id: ObjectId('6595eb1d86bd8592a71b742f'),
    totalYellowCards: 69,
    playerName: 'Nicolò Barella'
  }
```

Figure 4.3: Players with multiple yellow cards Executed with IT1 competition

4.2.5.   Players with multiple red cards

In contrast to yellow cards, which can happen in the course of a match, getting a red card means, most of the time, having committed something serious, such as a bad foul, violent conduct or repeated protests to the referee. This statistic shows those players who struggle most to maintain control on the pitch and whose attitudes risk leaving the team one down. The query starts with *unwindtobreakdowntheappearancesarrayofeachplayerdocument, thenfilters:* aggregates the data by player, adding up the red cards for each player.

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $match: { "appearances.competition_id": "<competition_id>" } },
  { $group: { _id: "$_id", totalRedCards:
  { $sum: "$appearances.red_cards" }, playerName: { $first: "$name" } } },
  { $sort: { totalRedCards: -1 } },
  { $limit: 10 }
])
```

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "GB1" } },
    { $group: { _id: "$_id", totalRedCards: { $sum: "$appearances.red_cards" }, playerName: { $first: "$name" } } },
    { $sort: { totalRedCards: -1 } },
    { $limit: 10 }
  ])
< {
    _id: ObjectId('6595eb0786bd8592a71b4e22'),
    totalRedCards: 4,
    playerName: 'David Luiz'
  }
  {
    _id: ObjectId('6595eb1d86bd8592a71b757e'),
    totalRedCards: 4,
    playerName: 'Granit Xhaka'
  }
  {
    _id: ObjectId('6595eb0e86bd8592a71b56cc'),
    totalRedCards: 3,
    playerName: 'Fernandinho'
  }
```

Figure 4.4: Players with multiple red cards Executed with IT1 competition

4.2.6.   Players with the most minutes played

This analysis shows those players who are certainties for their clubs: the so-called immovable starters. These players, workaholics par excellence, are usually the players who are almost always at the top, avoiding injuries. The query breaks down appearances, then it filters the appearances in a certain league, and then sums up the minutes played by each player, showing also the name of him.

```
db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "<competition_id>" } },
    { $group: { _id: "$_id", totalMinutesPlayed:
    { $sum: "$appearances.minutes_played" }, playerName: { $first: "$name" } } },
    { $sort: { totalMinutesPlayed: -1 } },
    { $limit: 10 }
])
```

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $match: { "appearances.competition_id": "ES1" } },
    { $group: { _id: "$_id", totalMinutesPlayed: { $sum: "$appearances.minutes_played" }, playerName: { $first: "$name" } } },
    { $sort: { totalMinutesPlayed: -1 } },
    { $limit: 10 }
])
< {
    _id: ObjectId('6595eb2086bd8592a71b79e1'),
    totalMinutesPlayed: 26361,
    playerName: 'Jan Oblak'
}
{
    _id: ObjectId('6595eb0986bd8592a71b5098'),
    totalMinutesPlayed: 26359,
    playerName: 'Dani Parejo'
}
{
    _id: ObjectId('6595eb1486bd8592a71b6455'),
    totalMinutesPlayed: 24491,
    playerName: 'Koke'
}
```

Figure 4.5: Players with the most minutes played Executed with IT1 competition

4.2.7.   The players with the highest market value

This query is designed to identify the top football players in the database, highlighting those with the highest market value. It provides an overview of the most valuable talents

in the world of football, revealing the names of players who have not only demonstrated excellent performances on the field, but are also considered valuable investments in the football landscape. Players can sometimes be overestimated or underestimated. This data is important because it takes into account not only performance but also a player's age. Younger players with important statistics will command high market prices.

The first proposed query shows the current situation of market values for the players in the dataset ordered by their value in a descendent way.

```
db.players.find({},{ name: 1, market_value_in_eur: 1 }).sort(
    { market_value_in_eur: -1 }).limit(10)
```


Figure 4.6: The players with the highest market value First Query Execution

The second query shows the best valuation for each player during its career and then orders these values in a descendent way.

```
db.players.aggregate([
  { $unwind: "$valuations" },
  { $group: { _id: "$_id", name: { $first: "$name" }, maxMarketValue:
    { $max: "$valuations.market_value_in_eur" }, season:
```

```
          { $first: "$valuations.last_season" } } },
      { $sort: { maxMarketValue: -1 } }
    ])
```


Figure 4.7: The players with the highest market value Second Query Execution

4.2.8.   The cities where the most players were born

This query lists the cities that produced the most players. It uses the $group operation to group players by their city of birth, counts the number of players for each city with $sum, and then sorts the results descendingly to show which cities generated the most football talents.

```
db.players.aggregate([
  { $group: { _id: "$city_of_birth", numberOfPlayers: { $sum: 1 } } },
  { $sort: { numberOfPlayers: -1 } }
])
```

Not all the players in the database have a city of birth, so there are a count also of the players with this attribute null.

```
> db.players.aggregate([
    { $group: { _id: "$city_of_birth", numberOfPlayers: { $sum: 1 } } },
    { $sort: { numberOfPlayers: -1 } }
  ])
< {
    _id: null,
    numberOfPlayers: 1789
  }
  {
    _id: 'London',
    numberOfPlayers: 394
  }
  {
    _id: 'Istanbul',
    numberOfPlayers: 269
  }
  {
    _id: 'Amsterdam',
    numberOfPlayers: 186
  }
```

Figure 4.8: The cities where the most players were born Query Execution

4.2.9. The most prolific and expensive strikers

This query aims to identify the center forwards (position "A") with the highest market value and the most goals scored. We start by filtering the players by their position, then use the $unwind operator to decompose the array of appearances. Next, you aggregate the data to calculate the total number of goals and sort the results by market value and goals in a descending fashion. The first returned players are likely to be stars of the football.

```
db.players.aggregate([
    { $match: { position: "A" } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals:
        { $sum: "$appearances.goals" }, marketValue:
            { $first: "$market_value_in_eur" } } },
    { $sort: { marketValue: -1, totalGoals: -1 } }
  ])
```

```
> db.players.aggregate([
    { $match: { position: "A" } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals: { $sum: "$appearances.goals" }, marketValue: { $first: "$market_value_in_eur" } } },
    { $sort: { marketValue: -1, totalGoals: -1 } }
  ])
< {
    _id: ObjectId('6595eb1286bd8592a71b60f5'),
    name: 'Kylian Mbappé',
    totalGoals: 196,
    marketValue: 180000000
  }
  {
    _id: ObjectId('6595eb2186bd8592a71b7c66'),
    name: 'Erling Haaland',
    totalGoals: 141,
    marketValue: 170000000
  }
  {
    _id: ObjectId('6595eb2886bd8592a71b89a8'),
    name: 'Vinicius Junior',
    totalGoals: 57,
    marketValue: 120000000
  }
  {
    _id: ObjectId('6595eb1f86bd8592a71b7662'),
    name: 'Phil Foden',
    totalGoals: 59,
    marketValue: 110000000
  }
```

Figure 4.9: The most prolific and expensive strikers Query Execution

4.2.10. Midfielders with the most goals and assists and a bounded valuation

The query focuses on midfielders (position "M") with a market value lower and upper bound, but with a high number of goals and assists. After filtering by position, it uses $unwind to decompose the array of appearances. It then aggregates the data to calculate the total goals and assists for each player, and finally sorts the results by goals and assists in descending order, taking into account only those players whose market value falls within a predefined average range. As can be understood, if a club has a certain budget available and wants to look for the best ones to fill a role, this query will be very useful.

```
db.players.aggregate([
    { $match: { position: "M", market_value_in_eur:
     { $gte: 10000000, $lte: 20000000 } } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals:
        { $sum: "$appearances.goals" }, totalAssists:
            { $sum: "$appearances.assists" } } },
    { $sort: { totalGoals: -1, totalAssists: -1 } }
   ])
```

```
> db.players.aggregate([
    { $match: { position: "M", market_value_in_eur: { $gte: 10000000, $lte: 20000000 } } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals: { $sum: "$appearances.goals" }, totalAssists: { $sum: "$appearances.assists" } } },
    { $sort: { totalGoals: -1, totalAssists: -1 } }
])
< {
    _id: ObjectId('6595eb1286bd8592a71b5f09'),
    name: 'Steven Berghuis',
    totalGoals: 121,
    totalAssists: 95
}
{
    _id: ObjectId('6595eb0e86bd8592a71b58ad'),
    name: 'Davy Klaassen',
    totalGoals: 96,
    totalAssists: 59
}
{
    _id: ObjectId('6595eb0386bd8592a71b438a'),
    name: 'Hans Vanaken',
    totalGoals: 91,
    totalAssists: 63
}
{
    _id: ObjectId('6595eb0586bd8592a71b4a82'),
    name: 'Anastasios Bakasetas',
    totalGoals: 75,
    totalAssists: 50
}
```

Figure 4.10: Midfielders with the most goals and assists and a bounded valuation Query Execution.(min=10000000,max=20000000)

4.2.11. The cheapest but most prolific full-back defenders

This query aims to identify right-backs (RB) and left-backs (LB) with the lowest market value but high assists and goals. It filters players by RB and LB sub-positions, calculates the total assists and goals, and sorts first by ascending market value and then by descending assists and goals. The query highlights those full-backs who, while not having a high market value, have a significant impact in terms of their contribution to the game. Despite being defenders, offensive-minded teams need these players to increase their number of goals.

```
db.players.aggregate([
    { $match: {
        sub_position: { $in: ["RB", "LB"] },
        market_value_in_eur: { $ne: null }    }},
    { $unwind: "$appearances" },
    { $group: {
        _id: "$_id",
        name: { $first: "$name" },
        marketValue: { $first: "$market_value_in_eur" },
        totalAssists: { $sum: "$appearances.assists" },
        totalGoals: { $sum: "$appearances.goals" }
```

```
    }},
    { $sort: { marketValue: 1, totalAssists: -1, totalGoals: -1 } }
  ])
```

```
db.players.aggregate([
  { $match: {
      sub_position: { $in: ["RB", "LB"] },
      market_value_in_eur: { $ne: null }    }},
  { $unwind: "$appearances" },
  { $group: {
      _id: "$_id",
      name: { $first: "$name" },
      marketValue: { $first: "$market_value_in_eur" },
      totalAssists: { $sum: "$appearances.assists" },
      totalGoals: { $sum: "$appearances.goals" }
  }},
  { $sort: { totalAssists: -1, totalGoals: -1, marketValue: 1  } }
])


{
  _id: ObjectId('6595eb1086bd8592a71b5d55'),
  name: 'Jordi Alba',
  marketValue: 5000000,
  totalAssists: 87,
  totalGoals: 21
}
{
  _id: ObjectId('6595eb0586bd8592a71b4977'),
  name: 'James Tavernier',
  marketValue: 8000000,
  totalAssists: 82,
  totalGoals: 74
}
```

Figure 4.11: The cheapest but most prolific full-back defenders Query Execution

4.2.12. Players in a Specific Match

This aggregation query lists all players who appeared in a specific match, identified by the game_ID. It begins by deconstructing the appearances array with $unwind. Then, it filters the documents with $match to only those where the game_id within appearances matches the given ID. Finally, $project is used to exclude the valuations field from the output, returning all other player details. This query is helpful for analyzing participation in particular games. In the case of this query the game:id chosen was 2469936 (of course it could be changed).

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $match: { "appearances.game_id": 2469936 } },
  { $project:  { valuations: 0 } }
])
```


Figure 4.12: Players in a Specific Match Query Execution

4.2.13. Defender with most cards (Yellow and Red cards summed up)

This query identifies the defenders who received the highest total number of cards, summing yellow and red cards. After filtering players by defensive position, the query breaks down the array of appearances, aggregates the total number of red and yellow cards per player, and finally sorts the results in descending order to display the most "undisciplined" defenders.

```
db.players.aggregate([
  { $match: { position: "D" } },
  { $unwind: "$appearances" },
  { $group: { _id: "$_id", name: { $first: "$name" }, totalCards:
  { $sum: { $add:
    ["$appearances.yellow_cards", "$appearances.red_cards"] } } } },
  { $sort: { totalCards: -1 } }
])
```

```
> db.players.aggregate([
    { $match: { position: "D" } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalCards: { $sum: { $add: ["$appearances.yellow_cards", "$appearances.red_cards"] } } } },
    { $sort: { totalCards: -1 } }
  ])
< {
    _id: ObjectId('6595eb1f86bd8592a71b7878'),
    name: 'Sergio Ramos',
    totalCards: 105
  }
  {
    _id: ObjectId('6595eb0586bd8592a71b49a8'),
    name: 'Damián Suárez',
    totalCards: 104
  }
  {
    _id: ObjectId('6595eb2386bd8592a71b8057'),
    name: 'Kalidou Koulibaly',
    totalCards: 100
  }
  {
    _id: ObjectId('6595eb1286bd8592a71b5f26'),
    name: 'Daniel Carvajal',
    totalCards: 100
  }
```

Figure 4.13: Defender with most cards Query Execution

4.2.14. The nations with the best talents in terms of market value during the years

This query identifies the nations that produce the players with the highest market values by analyzing the maximum market value achieved by each player. After decomposing the valuations array, it aggregates the maximum market value for each player and then averages these maximum values by nation, sorting the nations by this average.

```
db.players.aggregate([
  { $unwind: "$valuations" },
  { $group: { _id: "$_id", country:
    { $first: "$country_of_citizenship" }, maxMarketValue:
    { $max: "$valuations.market_value_in_eur" } } },
  { $group: { _id: "$country", averageMaxMarketValue:
    { $avg: "$maxMarketValue" } } },
  { $sort: { averageMaxMarketValue: -1 } }
])
```

```
> db.players.aggregate([
    { $unwind: "$valuations" },
    { $group: { _id: "$_id", country: { $first: "$country_of_citizenship" }, maxMarketValue: { $max: "$valuations.market_value_in_eur" } } },
    { $group: { _id: "$country", averageMaxMarketValue: { $avg: "$maxMarketValue" } } },
    { $sort: { averageMaxMarketValue: -1 } }
])
< {
    _id: 'Mexico',
    averageMaxMarketValue: 10108510.638297873
  }
  {
    _id: 'Uruguay',
    averageMaxMarketValue: 8180847.953216374
  }
  {
    _id: 'Argentina',
    averageMaxMarketValue: 7845841.209829868
  }
  {
    _id: 'Colombia',
    averageMaxMarketValue: 7009615.384615385
  }
  {
    _id: 'Korea, South',
    averageMaxMarketValue: 6959146.341463415
  }
```

Figure 4.14: Nations with the best talents in terms of market value Query Execution

4.2.15.   The nations with most goals scored by their players

This query identifies the nation that produces the players with the most goals. It first breaks down the array of each player's appearances, then groups the players by nation and adds up the goals scored. Finally, it sorts the nations by total number of goals to find out which nation produced the best scorers.

```
db.players.aggregate([
  { $unwind: "$appearances" },
  { $group: { _id: "$country_of_citizenship", totalGoals:
    { $sum: "$appearances.goals" } } },
  { $sort: { totalGoals: -1 } }
])
```

```
> db.players.aggregate([
    { $unwind: "$appearances" },
    { $group: { _id: "$country_of_citizenship", totalGoals: { $sum: "$appearances.goals" } } },
    { $sort: { totalGoals: -1 } }
  ])
< {
    _id: 'Spain',
    totalGoals: 7852
  }
  {
    _id: 'France',
    totalGoals: 7084
  }
  {
    _id: 'Brazil',
    totalGoals: 6551
  }
  {
    _id: 'Netherlands',
    totalGoals: 6391
  }
  {
    _id: 'England',
    totalGoals: 5111
  }
  {
    _id: 'Portugal',
    totalGoals: 4918
  }
```

Figure 4.15: Nations with most goals scored by their players Query Execution

4.2.16.   Average Market Value for each club

This query calculates the average market value of the players for each club. It uses the $group operation to group players by their current club, then calculates the average market value (found in the valuations array) for each club. Finally, sort the results to show the clubs by the average market value of their players.

```
db.players.aggregate([
  { $unwind: "$valuations" },
  { $group: { _id: "$current_club_name", averageMarketValue:
    { $avg: "$valuations.market_value_in_eur" } } },
  { $sort: { averageMarketValue: -1 } }
])
```

```
> db.players.aggregate([
    { $unwind: "$valuations" },
    { $group: { _id: "$current_club_name", averageMarketValue: { $avg: "$valuations.market_value_in_eur" } } },
    { $sort: { averageMarketValue: -1 } }
  ])
< {
    _id: 'Paris Saint-Germain',
    averageMarketValue: 24325240.9626506
  }
  {
    _id: 'Real Madrid',
    averageMarketValue: 23218034.55723542
  }
  {
    _id: 'FC Barcelona',
    averageMarketValue: 19935009.57854406
  }
  {
    _id: 'Manchester City',
    averageMarketValue: 19796327.751196172
  }
  {
    _id: 'Manchester United',
    averageMarketValue: 19772046.924757283
  }
  {
    _id: 'Liverpool FC',
    averageMarketValue: 18877150.837988827
  }
  {
    _id: 'Chelsea FC',
    averageMarketValue: 16200668.89632107
  }
```

Figure 4.16: Average Market Value for each club Query Execution

4.2.17.   Most Prolific free agents

This query identifies free agent players with the best goal scoring statistics in the last season played. It starts by filtering players without a current club (free agent), then examines their performance in the last season, summing the total number of goals scored. Finally, it sorts the results to show the best scorers among free agents. Many teams resort to signing players without contracts to save on transfer costs, paying only contract fees and some fees to agents or at signing.

```
db.players.aggregate([
  { $match: { current_club_name: null } },
  { $unwind: "$appearances" },
  { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals:
    { $sum: "$appearances.goals" }, lastSeason:
        { $max: "$appearances.last_season" } } },
  { $sort: { totalGoals: -1 } }
])
```
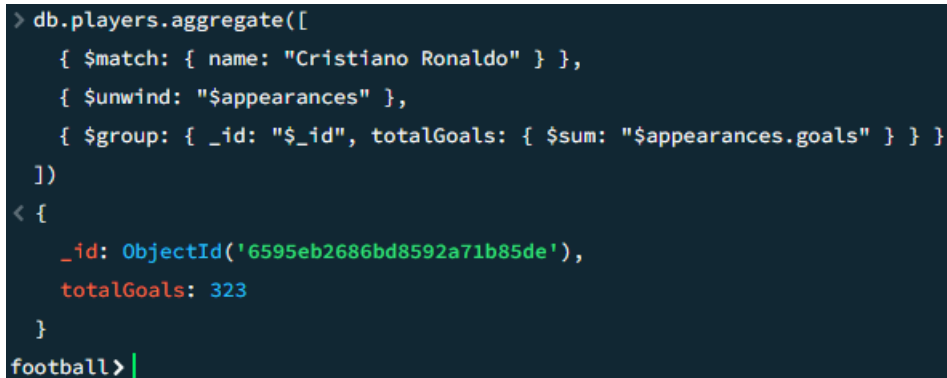
```
> db.players.aggregate([
    { $match: { current_club_name: null } },
    { $unwind: "$appearances" },
    { $group: { _id: "$_id", name: { $first: "$name" }, totalGoals: { $sum: "$appearances.goals" }, lastSeason: { $max: "$appearances.last_season" } } },
    { $sort: { totalGoals: -1 } }
])
< {
    _id: ObjectId('6595eb0186bd8592a71b4271'),
    name: 'Leigh Griffiths',
    totalGoals: 108,
    lastSeason: null
  }
  {
    _id: ObjectId('6595eaff86bd8592a71b3c5e'),
    name: 'Carlos Bacca',
    totalGoals: 106,
    lastSeason: null
  }
  {
    _id: ObjectId('6595eaf486bd8592a71b27a9'),
    name: 'Teemu Pukki',
    totalGoals: 86,
    lastSeason: null
  }
  {
    _id: ObjectId('6595eb0586bd8592a71b4994'),
    name: 'Lucas Pérez',
    totalGoals: 69,
    lastSeason: null
  }
```

Figure 4.17: Most Prolific free agents Query Execution

4.2.18. Cristiano Ronaldo (specific player) goals

To calculate the total number of goals scored by Cristiano Ronaldo, the query should first locate the specific player in the database (e.g., by name or unique ID). Next, it uses $unwind to decompose the array of appearances and $group to sum the goals scored. Finally, it presents the total goals. Obviously, by changing names and searching for another player, one can see how many goals the requested player has scored.

```
db.players.aggregate([
  { $match: { name: "Cristiano Ronaldo" } },
  { $unwind: "$appearances" },
  { $group: { _id: "$_id", totalGoals: { $sum: "$appearances.goals" } } }
])
```


Figure 4.18: Cristiano Ronaldo Query Execution

4.2.19.   The youngest player

To find the youngest player in the database, the query sorts all players by their date of
birth, from newest to oldest. In this way, the first player in the resulting list will be the
youngest. Of course, to find the oldest you need to change "-1" with "1".

```
db.players.find().sort({ date_of_birth: -1 }).limit(1)
```


Figure 4.19: The youngest player Query Execution
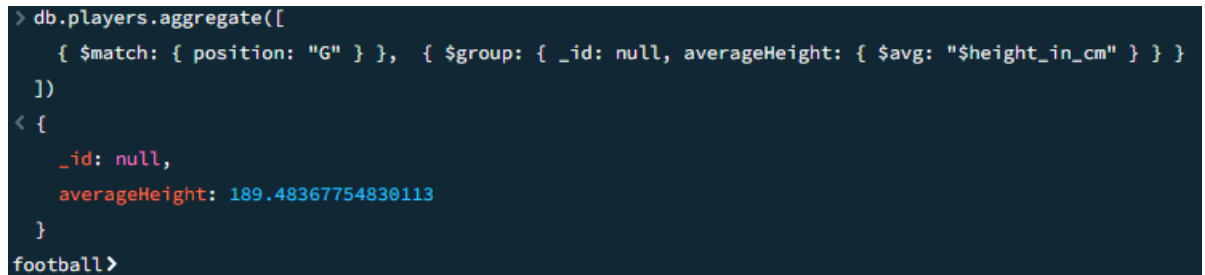
4.2.20.   Average height of goalkeepers

Within the 'players' collection the query first filters by position, selecting only players clas-
sified as goalkeepers (e.g., position "G" for goalkeeper), then groups the data to calculate
the average height.

```
db.players.aggregate([
  { $match: { position: "G" } },
  { $group: { _id: null, averageHeight: { $avg: "$height_in_cm" } } }
])
```


Figure 4.20: Average height of goalkeepers Query Execution
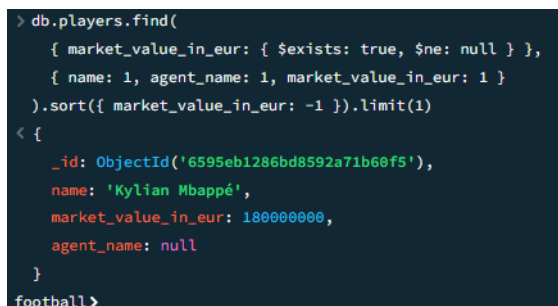
4.2.21. Agent with the most valuable player

This query identifies the agent of the player with the highest market value recorded directly in the player's document. Each player is assumed to have a market value defined in the market_value_in_eur field. The query sorts all player documents by this value in a descending manner and selects the first document, which represents the player with the highest value, and then projects the name of the agent.

```
db.players.find(
  { market_value_in_eur: { $exists: true, $ne: null } },
  { name: 1, agent_name: 1, market_value_in_eur: 1 }
).sort({ market_value_in_eur: -1 }).limit(1)
```


Figure 4.21: Agent with the most valuable player Query Execution

However, in this specific case, the result will show "null" for the agent_name field indicating that there is no registered agent, which may correspond to situations where a family member, such as the mother in Mbappé's case, assists the player in contractual matters.

4.2.22.   The italian players with the highest goal ratio

This query shows the best italian players with the highest ratio between goal and appearances. To get the Italian players with the highest goal-to-presence ratio, the query would first calculate the total goals and total appearances for each Italian player, then divide the total goals by the total number of appearances to get the ratio. Here is how it would be structured:

1. Filter players by country_of_citizenship set to "Italy."

2. Expand the appearances array.

3. Group the results by player, adding up the goals scored and counting the appearances.

4. Calculate the goals/appearances ratio for each player.

5. Sort the results by goal/attendance ratio in a descending manner.

The statistics are important only for players with more than 50 appearances.

```
db.players.aggregate([
  { $match: { country_of_citizenship: "Italy" } },
  { $unwind: "$appearances" },
  { $group: {
      _id: "$_id",
      name: { $first: "$name" },
      totalGoals: { $sum: "$appearances.goals" },
      totalAppearances: { $sum: 1 }
    }
  },
  { $match: { totalAppearances: { $gte: 50 } } },
  { $project: {
      name: 1,
      goalRatio: { $divide: [ "$totalGoals", "$totalAppearances" ] }
    }
  },
  { $sort: { goalRatio: -1 } }
])
```

```
> db.players.aggregate([
    { $match: { country_of_citizenship: "Italy" } },
    { $unwind: "$appearances" },
    { $group: {
        _id: "$_id",
        name: { $first: "$name" },
        totalGoals: { $sum: "$appearances.goals" },
        totalAppearances: { $sum: 1 }
      }
    },
    { $match: { totalAppearances: { $gte: 50 } } },
    { $project: {
        name: 1,
        goalRatio: { $divide: [ "$totalGoals", "$totalAppearances" ] }
      }
    },
    { $sort: { goalRatio: -1 } }
])
< {
    _id: ObjectId('6595eb2386bd8592a71b8077'),
    name: 'Ciro Immobile',
    goalRatio: 0.6085714285714285
  }
  {
    _id: ObjectId('6595eb0586bd8592a71b486b'),
    name: 'Luca Toni',
    goalRatio: 0.46875
  }
  {
    _id: ObjectId('6595eb0186bd8592a71b4269'),
    name: 'Mario Balotelli',
    goalRatio: 0.41578947368421054
  }
```

Figure 4.22: The italian players with the highest goal ratio Query Execution

4.2.23.   The French players with the highest minutes ratio

This query shows the best italian players with the highest ratio between minutes and appearances. To get the French players with the highest minutes-to-apperance ratio, the query would first calculate the total minutes and total appearances for each player, then divide the total minutes by the total number of appearances to get the ratio. Here is how it would be structured:

1. Filter players by country_of_citizenship set to "France."

2. Expand the appearances array.

3. Group the results by player, adding up the minutes played and counting the appearances.

4. Calculate the minutes/appearances ratio for each player.

5. Sort the results by minutes/attendance ratio in a descending manner.

6. The statistics are important only for players with more than 50 appearances.

```
db.players.aggregate([
  { $match: { country_of_citizenship: "France" } },
  { $unwind: "$appearances" },
  { $group: {
      _id: "$_id",
      name: { $first: "$name" },
      totaMinutes: { $sum: "$appearances.minutes_played" },
      totalAppearances: { $sum: 1 }
    }
  },
  { $match: { totalAppearances: { $gte: 50 } } },
  { $project: {
      name: 1,
      minutesRatio: { $divide: [ "$totaMinutes", "$totalAppearances" ] }
    }
  },
  { $sort: { minutesRatio: -1 } }
])
```
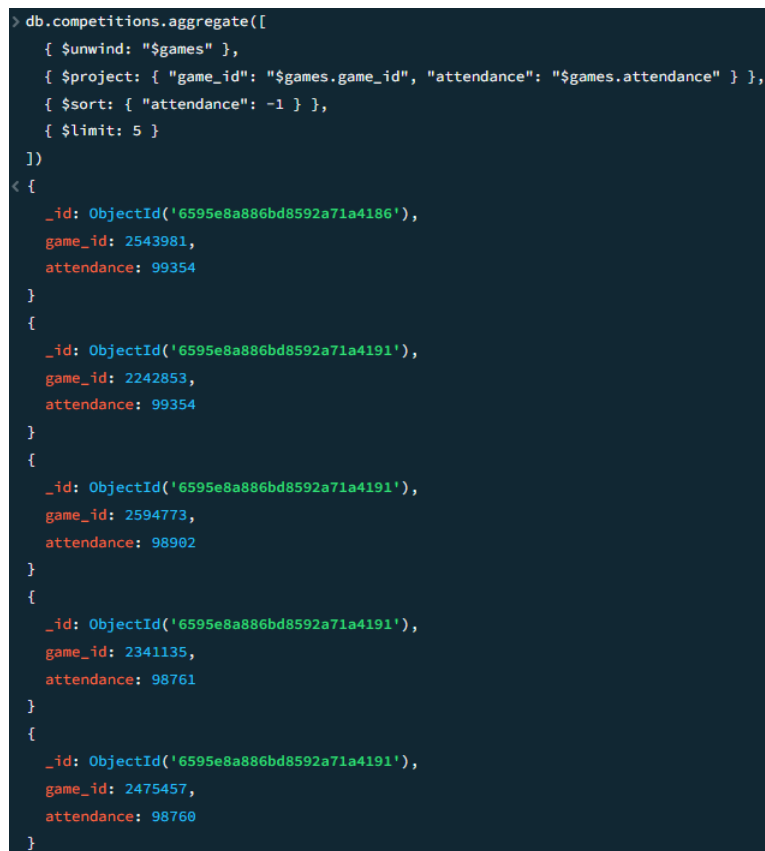

Figure 4.23: The French players with the highest minutes ratio Query Execution

4.3.   Competitions Collection

4.3.1.   Most Followed Matches

The query selects the games from the 'competitions' collection that had the most spectators. It uses the 'attendance' field within the array of 'games' for each competition. The goal is to identify those games that attracted the most significant audience attention, an indicator of the match's popularity or importance. The first query returns the most followed matches among all the competitions.

```
db.competitions.aggregate([
  { $unwind: "$games" },
  { $project:
    { "game_id": "$games.game_id", "attendance": "$games.attendance" } },
  { $sort: { "attendance": -1 } },
  { $limit: 5 }
])
```


Figure 4.24: Most Followed Matches First Query Execution

The second query returns the most followed matches filtering by competition. It is inserted a placeholder "<competition_id>" to search the preferred league. In the picture competition_id is equal to CL (Champions League)

```
db.competitions.aggregate([
    { $match: { "competition_id": "<competition_id>" } },
    { $unwind: "$games" },
    { $project: { "game_id":
        "$games.game_id", "attendance": "$games.attendance", "competition_id": 1 } },
    { $sort: { "attendance": -1 } },
    { $limit: 5 }
])
```


Figure 4.25: Most Followed Matches Second Query Execution

4.3.2.   Most goals scored by a single team

These queries are designed to retrieve the games with the highest number of goals scored by a single team, with the second one specifically filtering for games within the "Champions League" competition. They unwind the games array, calculate the maximum goals scored in a game, sort the results by this maximum in descending order, and limit the output to the top 5 records.

```
db.competitions.aggregate([
  { $unwind: "$games" },
  { $project: { game_id: "$games.game_id", maxGoals:
  { $max: ["$games.home_club_goals", "$games.away_club_goals"] } } },
  { $sort: { maxGoals: -1 } },
  { $limit: 5 }
])
```


Figure 4.26: Most Goals by single team First Query Execution

```
db.competitions.aggregate([
  { $match: { "competition_id": "CL" } },
  { $unwind: "$games" },
  { $project: { game_id: "$games.game_id", maxGoals:
  { $max: ["$games.home_club_goals", "$games.away_club_goals"] } } },
  { $sort: { maxGoals: -1 } },
  { $limit: 5 }
])
```

```
> db.competitions.aggregate([
    { $match: { "competition_id": "CL" } },
    { $unwind: "$games" },
    { $project: { game_id: "$games.game_id", maxGoals: { $max: ["$games.home_club_goals", "$games.away_club_goals"] } } },
    { $sort: { maxGoals: -1 } },
    { $limit: 5 }
  ])
< {
    _id: ObjectId('6595e8a886bd8592a71a4188'),
    game_id: 2755715,
    maxGoals: 8
  }
  {
    _id: ObjectId('6595e8a886bd8592a71a4188'),
    game_id: 2618627,
    maxGoals: 8
  }
  {
    _id: ObjectId('6595e8a886bd8592a71a4188'),
    game_id: 2645819,
    maxGoals: 8
  }
  {
    _id: ObjectId('6595e8a886bd8592a71a4188'),
    game_id: 3414688,
    maxGoals: 8
  }
  {
    _id: ObjectId('6595e8a886bd8592a71a4188'),
    game_id: 3956684,
    maxGoals: 7
  }
```

Figure 4.27: Most Goals by single team Second Query Execution

4.4. Clubs Collection

4.4.1. Clubs with more late goals

The query runs through the 'clubs' collection, expands the 'games' array and its 'events' for each club, then filtering out events that are goals scored beyond the 80th minute. It then groups the results by 'club_code' and counts the late goals for each club, sorting the clubs by the number of these goals.

```
db.clubs.aggregate([
  { $unwind: "$games" },
  { $unwind: "$games.events" },
  { $match: { "games.events.type": "Goals", "games.events.minute":{ $gt: 80 } } },
  { $group: { _id: "$club_code", lateGoals: { $sum: 1 } } },
  { $sort: { lateGoals: -1 } }
])
```

```
> db.clubs.aggregate([
    { $unwind: "$games" },
    { $unwind: "$games.events" },
    { $match: { "games.events.type": "Goals", "games.events.minute": { $gt: 80 } } },
    { $group: { _id: "$club_code", lateGoals: { $sum: 1 } } },
    { $sort: { lateGoals: -1 } }
  ])
< {
    _id: 'real-madrid',
    lateGoals: 109
  }
  {
    _id: 'fc-barcelona',
    lateGoals: 89
  }
  {
    _id: 'borussia-dortmund',
    lateGoals: 84
  }
  {
    _id: 'fc-valencia',
    lateGoals: 83
  }
  {
    _id: 'ajax-amsterdam',
    lateGoals: 80
  }
```

Figure 4.28: Late goals Query Execution

4.4.2.   Clubs with most foreign players

The query selects all clubs and projects only the 'club_code' and the 'foreigners_percentage,'
which is the percentage of foreign players in the team. It then sorts the clubs from highest
to lowest percentage, to show which clubs have the largest proportion of foreign players
in their roster.

```
db.clubs.find({},
{"club_code": 1, "foreigners_percentage":1}).sort(
    { foreigners_percentage: -1 }).limit(5)
```

```
> db.clubs.find({}, {"club_code": 1, "foreigners_percentage":1}).sort({ foreigners_percentage: -1 }).limit(5)

< {
    _id: ObjectId('6595dde386bd8592a71a409e'),
    club_code: 'as-monaco',
    foreigners_percentage: 100
  }
  {
    _id: ObjectId('6595dde386bd8592a71a4095'),
    club_code: 'fc-fulham',
    foreigners_percentage: 91.7
  }
  {
    _id: ObjectId('6595dde386bd8592a71a415d'),
    club_code: 'fc-toulouse',
    foreigners_percentage: 88.5
  }
  {
    _id: ObjectId('6595dde386bd8592a71a3ff1'),
    club_code: 'cardiff-city',
    foreigners_percentage: 85.7
  }
  {
    _id: ObjectId('6595dde386bd8592a71a4073'),
    club_code: 'wolverhampton-wanderers',
    foreigners_percentage: 85.7
  }
```

Figure 4.29: Clubs with most foreign players Query Execution

4.4.3. Clubs with more wins at home

This query identifies clubs that have scored the most goals in home games. After expanding the array games, it filters for games played at home and sums the goals scored in these games. Finally, it sorts the clubs by the total number of goals scored at home.

```
db.clubs.aggregate([
  { $unwind: "$games" },
  { $match: { "games.hosting": "H" } },
  { $group: { _id: "$club_code", homeWins:
    { $sum: { $cond: [{ $eq: ["$games.is_win", 1] }, 1, 0] } } } },
  { $sort: { homeWins: -1 } }
])
```

```
> db.clubs.aggregate([
    { $unwind: "$games" },
    { $match: { "games.hosting": "H" } },
    { $group: { _id: "$club_code", homeWins: { $sum: { $cond: [{ $eq: ["$games.is_win", 1] }, 1, 0] } } } },
    { $sort: { homeWins: -1 } }
  ])
< {
    _id: 'fc-barcelona',
    homeWins: 230
  }
  {
    _id: 'juventus-turin',
    homeWins: 228
  }
  {
    _id: 'real-madrid',
    homeWins: 222
  }
  {
    _id: 'manchester-city',
    homeWins: 216
  }
  {
    _id: 'fc-bayern-munchen',
    homeWins: 215
  }
  {
    _id: 'celtic-glasgow',
    homeWins: 214
  }
  {
    _id: 'atletico-madrid',
    homeWins: 198
```

Figure 4.30: Clubs with more wins at home Query Execution

5 | Extra

5.1. Top Football Agents Analysis

In this section, we will analyze the documents obtained from the query described in (4.2.1). We will divide them into clusters to examine the distribution of the number of players in relation to those owned by the group, by category, and we will also analyze the impact that the 10 agents with the most players will have on the total number of players.

5.1.1. Result of Analysis

Percentage Distribution of Players by Agent Cluster

Summary of Agent and Player Distribution:
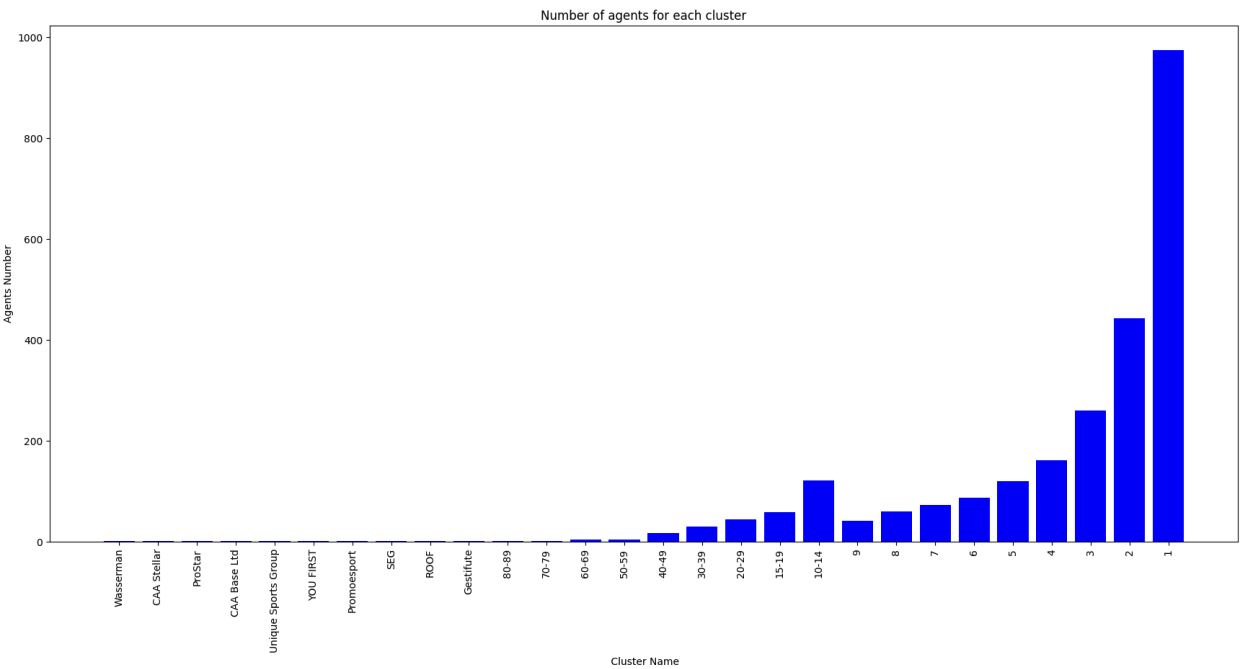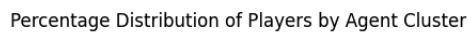
Percentage Distribution of Aget by group:
- Top 10 Category   10   Agents
- 80-89 Category: 2   Agents
- 70-79 Category: 2   Agents
- 60-69 Category: 5   Agents
- 50-59 Category: 5   Agents
- 40-49 Category: 17   Agents
- 30-39 Category: 30   Agents
- 20-29 Category: 45   Agents
- 15-19 Category: 59   Agents
- 10-14 Category: 122   Agents
- 9   Category: 41   Agents
- 8   Category: 60   Agents
- 7   Category: 73   Agents
- 6   Category: 87   Agents
- 5   Category: 120   Agents
- 4   Category: 162   Agents

```
- 3       Category: 260   Agents
- 2       Category: 443   Agents
- 1       Category: 974   Agents


Percentage Distribution of player by Agent group:
- Top 10 Category: 2098  Players 15.00 % of Total Player:
- 80-89  Category: 163   Players 1.17  % of Total Player:
- 70-79  Category: 154   Players 1.10  % of Total Player:
- 60-69  Category: 314   Players 2.24  % of Total Player:
- 50-59  Category: 269   Players 1.92  % of Total Player:
- 40-49  Category: 747   Players 5.34  % of Total Player:
- 30-39  Category: 1006  Players 7.19  % of Total Player:
- 20-29  Category: 1065  Players 7.61  % of Total Player:
- 15-19  Category: 995   Players 7.11  % of Total Player:
- 10-14  Category: 1406  Players 10.05 % of Total Player:
- 9       Category: 369   Players 2.64  % of Total Player:
- 8       Category: 480   Players 3.43  % of Total Player:
- 7       Category: 511   Players 3.65  % of Total Player:
- 6       Category: 522   Players 3.73  % of Total Player:
- 5       Category: 600   Players 4.29  % of Total Player:
- 4       Category: 648   Players 4.63  % of Total Player:
- 3       Category: 780   Players 5.58  % of Total Player:
- 2       Category: 886   Players 6.33  % of Total Player:
- 1       Category: 974   Players 6.96  % of Total Player:


 Top 10 Agents (Total: 15.00%):
- Wasserman          : 407   Players 2.91  % of Total Player:
- CAA Stellar        : 328   Players 2.35  % of Total Player:
- ProStar            : 315   Players 2.25  % of Total Player:
- CAA Base Ltd       : 222   Players 1.59  % of Total Player:
- Unique Sports Group : 198   Players 1.42  % of Total Player:
- YOU FIRST          : 158   Players 1.13  % of Total Player:
- Promoesport        : 132   Players 0.94  % of Total Player:
- SEG                : 129   Players 0.92  % of Total Player:
- ROOF               : 111   Players 0.79  % of Total Player:
- Gestifute          : 98    Players 0.70  % of Total Player:
```

5.1.2.　Python Code

```python
import matplotlib.pyplot as plt
import pymongo
import numpy as np
from collections import defaultdict

client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client.football
collection = db.players

# Query Definition
query = [
    { "$match": { "agent_name": { "$ne": None } } },
    { "$group": { "_id": "$agent_name", "numberOfPlayers": { "$sum": 1 } } },
    { "$sort": { "numberOfPlayers": -1 } }
]
# Query Execution
result = collection.aggregate(query)

# Range of visualization
ranges = [(10, 14), (15, 19), (20, 29), (30, 39), (40, 49), (50, 59), (60, 69), (70, 79), (80, 89), (90, 99)]

grouped_agents = defaultdict(int)
grouped_players = defaultdict(int)
sumOfTopTen = 0 #Sum of players of top ten agents

#Agent clustering
for i,record in enumerate(result):
    players = record['numberOfPlayers']

    if i < 10: #Top Ten
        grouped_agents[record['_id']] += 1
        grouped_players[record['_id']] += players
        sumOfTopTen += players

    elif players < 10: #Few players clustering
        grouped_agents[str(players)] += 1
        grouped_players[str(players)] += players

    elif players >= 100:#Many players clustering
        grouped_agents["100+"] += 1
        grouped_players["100+"] += players

    else: #Range clustering
        for r in ranges:
            if r[0] <= players <= r[1]:
                key = f'{r[0]}-{r[1]}'
                grouped_agents[key] += 1
                grouped_players[key] += players
                break

# Histogram of number of agent per cluster
............
categories_bar = list(grouped_agents.keys())
```

```python
agents_count = list(grouped_agents.values())

plt.figure(figsize=(12, 6))
plt.bar(categories_bar, agents_count, color='blue')
plt.xlabel(' Cluster Name')
plt.ylabel('Agents Number')
plt.title('Number of agents for each cluster')
plt.xticks(rotation=90)
plt.show()

#Pie chart of the percentage of players on the total for each cluster

categories_pie = list(grouped_players.keys())
total_players_pie = list(grouped_players.values())

plt.figure(figsize=(10, 10))
plt.pie(total_players_pie, labels=categories_pie, autopct='%1.1f%%', startangle=140)
plt.title('Percentage Distribution of Players by Agent Cluster')
plt.show()


# Text Output and Top Ten Analysis

summary_text = "Summary of Agent and Player Distribution:\n\n"
summary_text += "Percentage Distribution of Aget by group:\n"

summary_text += "- Top 10 : 10 Agents\n"
i=0
for range, count in grouped_agents.items():
    if(i>9):
        summary_text += f"- {range:<6} Category: {count:<5} Agents\n"
    i+=1

summary_text += "\nPercentage Distribution of player by Agent group:\n"
total_players_count = sum(grouped_players.values())
sumOfTopTenPerc= (sumOfTopTen / total_players_count) * 100
summary_text += f"- Top 10 Agents  : {sumOfTopTen:<5} Players {sumOfTopTenPerc:<6.2f}% of Total Player: \n"
i=0
for range, player_count in grouped_players.items():
    if(i>9):
        percentage = (player_count / total_players_count) * 100
        summary_text += f"- {range:<6} Category: {player_count:<5} Players {percentage:<6.2f}% of Total Player: \n"
    i+=1

summary_text += f"\n Top 10 Agents (Total: {sumOfTopTenPerc:.2f}%):\n"
i=0
for range, player_count in grouped_players.items():
    if(i<10):
        percentage = (player_count / total_players_count) * 100
        summary_text += f"- {range:<20} : {player_count:<5} Players {percentage:<6.2f}% of Total Player: \n"
    else: break
    i+=1
print(summary_text)
```

5.2. Players with the most appearances in the most-watched matches

In this section, we will analyze the documents obtained from the query in 4.3.1, to which we applied the query from 4.2.12. This is to find all the players who participated in the top 100 most-followed matches. Following this, we will count the appearances of these players in these matches and extract the 15 players with the most appearances.

5.2.1. Result of Analysis

The best 15 players by number of games played in the top 100 most-followed matches.

```
Sergio Busquets           with 52  games (Club ID: 131 )
Marc-André ter Stegen     with 50  games (Club ID: 131 )
Gerard Piqué              with 48  games (Club ID: 131 )
Lionel Messi              with 46  games (Club ID: 583 )
Sergi Roberto             with 46  games (Club ID: 131 )
Ivan Rakitic              with 45  games (Club ID: 368 )
Luis Suárez               with 43  games (Club ID: 13  )
Jordi Alba                with 42  games (Club ID: 131 )
Andrés Iniesta            with 30  games (Club ID: 131 )
Javier Mascherano         with 26  games (Club ID: 131 )
Neymar                    with 25  games (Club ID: 583 )
Ousmane Dembélé           with 19  games (Club ID: 131 )
Philippe Coutinho         with 16  games (Club ID: 405 )
Dani Alves                with 15  games (Club ID: 131 )
Nélson Semedo             with 15  games (Club ID: 543 )
```

From this analysis, it is evident that most of the players participating in these matches belong to the club with the ID 131.

Which is the club whose stadium has the most seating capacity in the dataset.

It can be concluded that although players have an impact on increasing the viewership of a match, the size of the stadium remains a significant factor in boosting the follower count of the matches.

5.2.2.   Python Code

```
1   import pymongo
2
3   client = pymongo.MongoClient("mongodb://localhost:27017/")
4   db = client.football
5
6   # Query Definition
7   query_games = [
8       { "$unwind": "$games" },
9       { "$project": { "game_id": "$games.game_id", "attendance": "$games.attendance" } },
10      { "$sort": { "attendance": -1 } },
11      { "$limit": 100 }
12  ]
13
14  #First Query Execution
15  resultGames = db.competitions.aggregate(query_games)
16
17  topNgames= []
18
19  for record in resultGames:
20      topNgames.append(record)
21
22  playerInTopGames = []
23
24  for record in topNgames:
25      query_player = [
26      { "$unwind": "$appearances" },
27      { "$match": { "appearances.game_id": record['game_id'] } },
28      { "$project": { "valuations": 0 } }
29  ]
30  #Second Query Execution
31      resultPlayer = db.players.aggregate(query_player)
32
33      for recordPlayer in resultPlayer:
34          playerInTopGames.append(recordPlayer)
35
36
37  player_presence_count = {}
38  for player in playerInTopGames:
39      key = (player['name'], player['current_club_id'])
40      player_presence_count[key] = player_presence_count.get(key, 0) + 1
41
42  # Sorting Players by Presence
43  sorted_player_by_presence = sorted(player_presence_count.items(), key=lambda x: x[1], reverse=True)
44
45  # Creating Summary Text
46  summary_text = "The best 15 players by number of games played in the top 100 most-followed matches.\n\n"
47  i = 0
48  for (player, club_id), games in sorted_player_by_presence:
49      summary_text += f"{player:<25} with {str(games):<3} games (Club ID: {club_id:<4})\n"
50      i += 1
51      if i > 14:
52          break
53  print(summary_text)
```

List of Figures