Project 2 - Graph Database Design and Cypher Query

Josh Mance (2340013)

Nguyen Le Cam Anh (23043141)

May 26, 2024

1 Introduction

This project involves designing a property graph, implementing it in Neo4j, and executing various queries using Cypher on a dataset of FIFA 2014 World Cup players. There are several key processes used: designing a property graph to visually represent nodes and relationships; performing ETL operations using CSV files and Python to convert and prepare the dataset for import into Neo4j; importing the transformed data into Neo4j and verifying data integrity; and developing and executing Cypher queries to extract insights. This report aims to cover the design and implementation of the property graph, the ETL process, the Neo4j implementation, query execution, and a discussion on the capabilities of graph databases compared to relational databases, concluding with reflections on the project's outcomes.

2 Database design

This is the sample database design for the project:

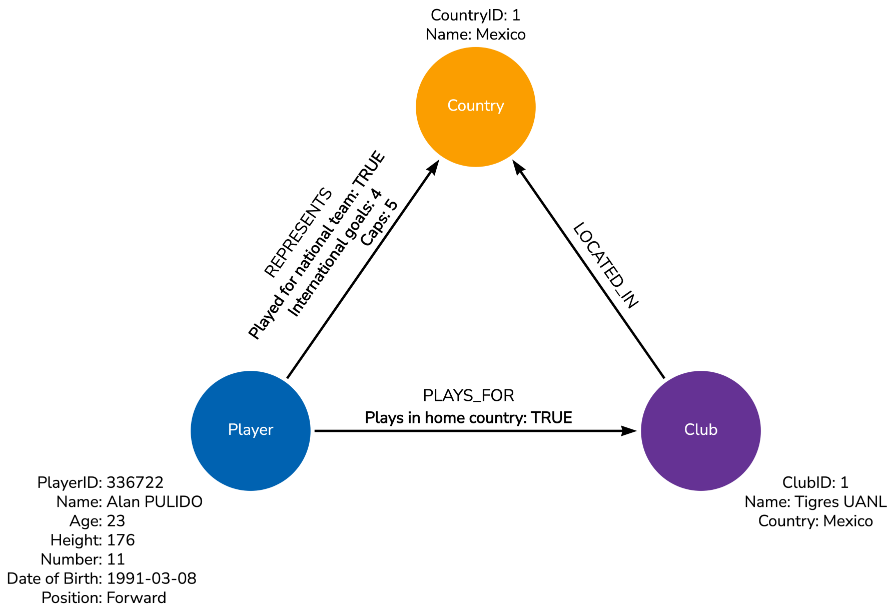


Figure 1: Property graph

Nodes are the fundamental entities in our graph. They hold a number of attributes (key value pairs) called property. Our design has three types of node/ labels which are: Player, Country and Club.

- **Player Node**: This node represents individual players and includes the following properties: Player ID, Name, Age, Height, Number, and Position.
- **Club Node**: This node represents football clubs and includes the properties: Club Name and Country.
- **Country Node**: This node represents countries and includes the property: Country Name.

Relationships provide relevant connections between different nodes. In our design:

- **REPRESENTS relationship**: This connects the Player node to the Country node, indicating the player represents that country. This relationship holds quantitative properties such as international goals and caps.
- **PLAYS_FOR relationship**: This connects the Player node to the Club node, indicating the player plays for that club.
- **LOCATED_IN relationship**: This connects the Club node to the Country node, representing the locating country of the club.

Relationships can also store properties, similar to nodes. For example, the quantitative properties like international goals and caps are stored in the REPRESENTS relationship. We have two categorical variables in our model:

- **Played for National Team:** This boolean variable indicates whether the player has played for the national team. It is treated as a property on the REPRESENTS relationship.
- **Plays in Home Country:** This boolean variable indicates whether the player plays for a club in their home country. It is treated as a property on the PLAYS_FOR relationship.

Both of these categorical variables are simple and take only two values: True or False. Thus, they do not require the complexity of separate nodes or labels, making the model more efficient.

A new property named "Played for national team" was added to indicate whether a player has represented their national team in international matches. This property is a boolean, meaning it can only take two values: True or False.

- **True:** The player has played for the national team at least once. This is determined by checking if the player has any caps (appearances in international matches).
- **False:** The player has not played for the national team. This is indicated by having zero caps.

3 ETL process

3.1 Data cleaning

We first construct the full file path to the CSV file. We use “getcwd()” to get the current working directory (the directory from which the script is being run) and appends the relative path data_raw/along with the file name. And then using pandas “pd.read_csv” to read the CSV file into a Dataframe.

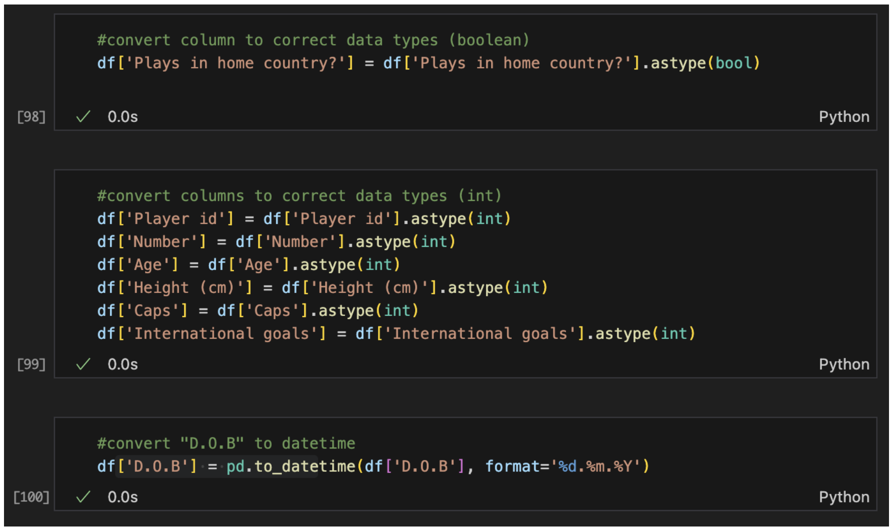
Once the data is extracted, we check for any missing values, duplicates and trailing white spaces. This function reads in the dataframe and the columns and counts the number of duplicate rows based on the columns. It then retrieves the duplicate rows if they exist and then prints it out.

```
def check_duplicates(df, subset=None):
```

This function check if any element in the column has trailing white space

```
def check_whitespace(column):
```

Next, we need to validate the data to ensure it conforms to the expected format and quality. The data is transformed by converting relevant columns to appropriate data types, such as integers for quantitative data and booleans for categorical data like ”Plays in home country?” and datetime for “D.O.B”.



```
#convert column to correct data types (boolean)
df['Plays in home country?'] = df['Plays in home country?'].astype(bool)
```

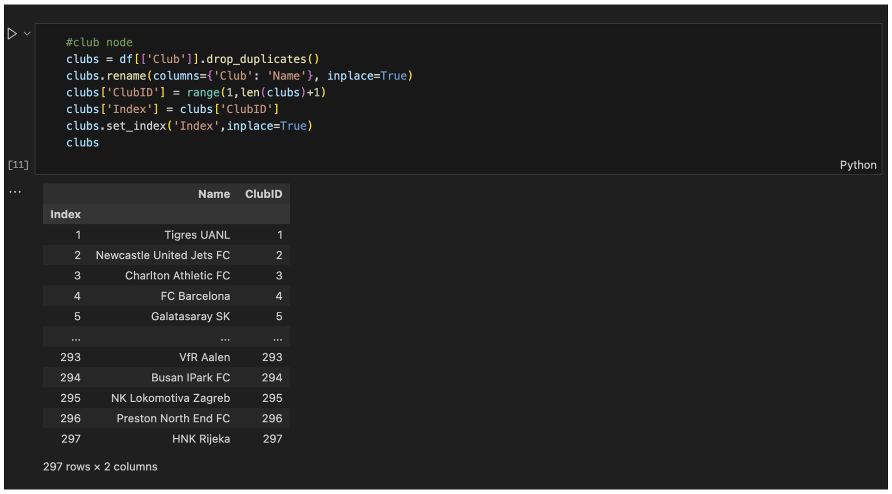
```
#convert columns to correct data types (int)
df['Player id'] = df['Player id'].astype(int)
df['Number'] = df['Number'].astype(int)
df['Age'] = df['Age'].astype(int)
df['Height (cm)'] = df['Height (cm)'].astype(int)
df['Caps'] = df['Caps'].astype(int)
df['International goals'] = df['International goals'].astype(int)
```

```
#convert "D.O.B" to datetime
df['D.O.B'] = pd.to_datetime(df['D.O.B'], format='%d.%m.%Y')
```

Figure 2: Enter Caption

New property ”Played for national team” is added to the REPRESENTS relationship, which is derived from the ”Caps” column.

To generate the data files for the nodes, we first combined relevant columns from the DataFrame into a single series. Next, we assigned a unique identifier to each entry, starting from 1 up to the total number of unique entries. This unique identifier was then set as the index of the DataFrame for easy reference. These DataFrames are then extracted as csv files



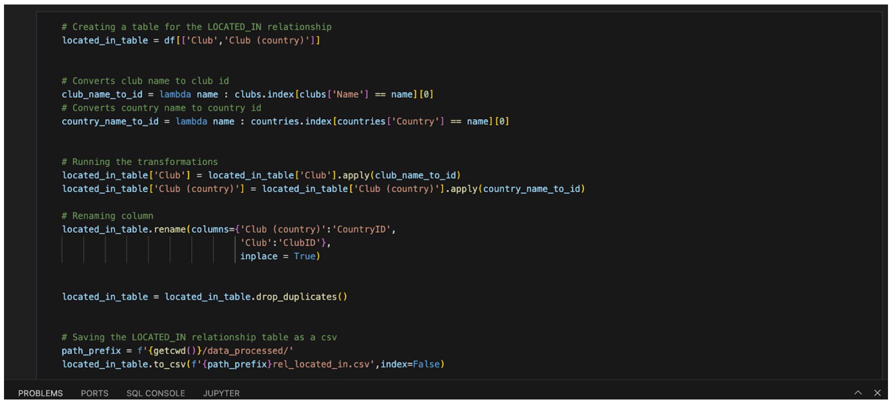
```
#club node
clubs = df[['Club']].drop_duplicates()
clubs.rename(columns={'Club': 'Name'}, inplace=True)
clubs['ClubID'] = range(1,len(clubs)+1)
clubs['Index'] = clubs['ClubID']
clubs.set_index('Index',inplace=True)
clubs
```

[11] Python

	Name	ClubID
Index		
1	Tigres UANL	1
2	Newcastle United Jets FC	2
3	Charlton Athletic FC	3
4	FC Barcelona	4
5	Galatasaray SK	5
...	...	...
293	VfR Aalen	293
294	Busan IPark FC	294
295	NK Lokomotiva Zagreb	295
296	Preston North End FC	296
297	HNK Rijeka	297

297 rows × 2 columns

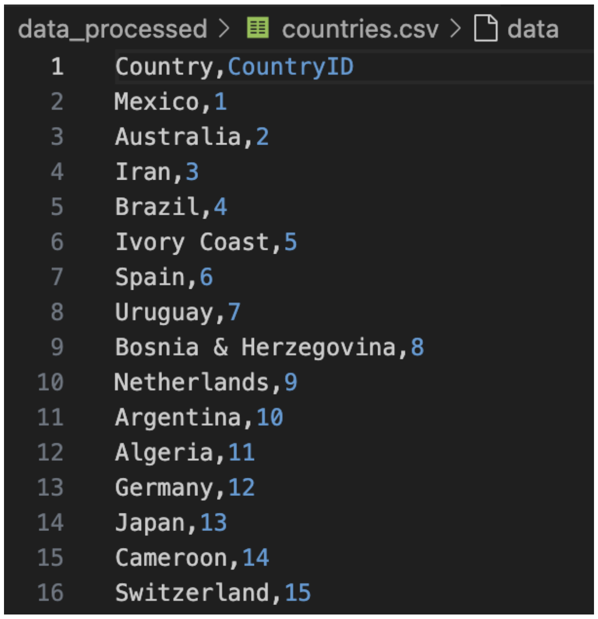
We create relationship tables by selecting columns from the original dataset that depict relationships between entities, such as clubs and their associated countries. We then convert entity names to their corresponding IDs from the combined DataFrame, and apply these functions to the relationship table to replace the names with IDs.



```
# Creating a table for the LOCATED_IN relationship
located_in_table = df[['Club','Club (country)']]


# Converts club name to club id
club_name_to_id = lambda name : clubs.index[clubs['Name'] == name][0]
# Converts country name to country id
country_name_to_id = lambda name : countries.index[countries['Country'] == name][0]


# Running the transformations
located_in_table['Club'] = located_in_table['Club'].apply(club_name_to_id)
located_in_table['Club (country)'] = located_in_table['Club (country)'].apply(country_name_to_id)

# Renaming column
located_in_table.rename(columns={'Club (country)':'CountryID',
                                 'Club':'ClubID'},
                                 inplace = True)


located_in_table = located_in_table.drop_duplicates()


# Saving the LOCATED_IN relationship table as a csv
path_prefix = f'{getcwd()}/data_processed/'
located_in_table.to_csv(f'{path_prefix}rel_located_in.csv',index=False)
```

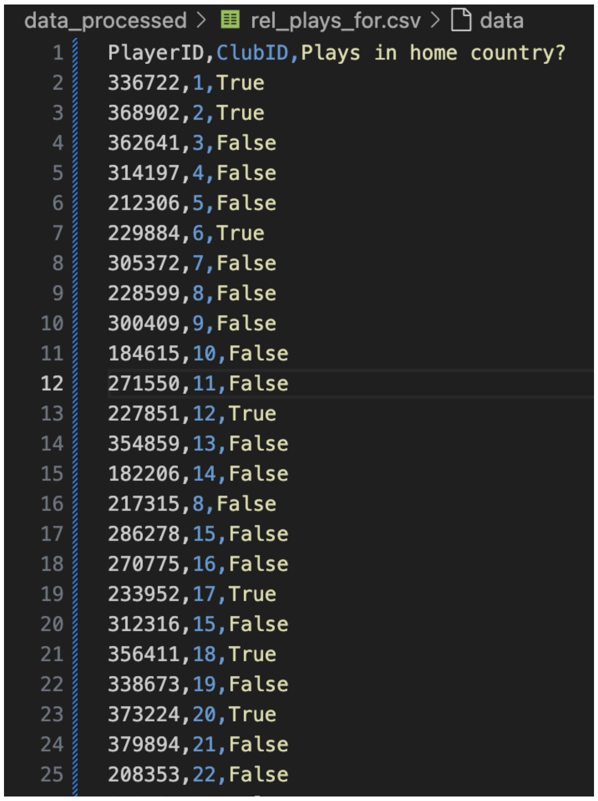
3.2 Preparing the CSV data

After all the files are extracted, we are going to load them into Neo4j. We need 2 types of files to do this: data and relationship files. Data files: there are 3 tables representing 3 nodes: player, country and club. Each column will correspond to a node property. All the tables have an ID column to make each row an unique value. For example, each row in the country table represents a country with its properties (CountryID, Name).


data_processed > countries.csv > data

```
Country,CountryID
Mexico,1
Australia,2
Iran,3
Brazil,4
Ivory Coast,5
Spain,6
Uruguay,7
Bosnia & Herzegovina,8
Netherlands,9
Argentina,10
Algeria,11
Germany,12
Japan,13
Cameroon,14
Switzerland,15
```

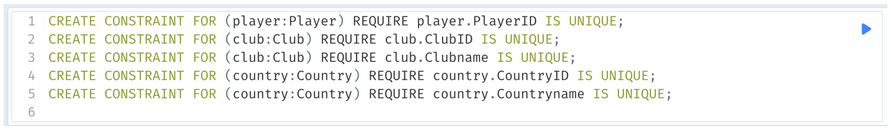
Relationship Files: These files will be loaded into Neo4j using Cypher queries to establish relationships between the Player node and the Club and Country nodes. Each relationship file contains two main columns: one for PlayerID and another for the ID of the related node, along with columns for any properties of the relationship. For example, the relationship file for the PLAYS_FOR relationship includes PlayerID, ClubID, and a property such as whether the player plays in their home country.



```
PlayerID,ClubID,Plays in home country?
336722,1,True
368902,2,True
362641,3,False
314197,4,False
212306,5,False
229884,6,True
305372,7,False
228599,8,False
300409,9,False
184615,10,False
271550,11,False
227851,12,True
354859,13,False
182206,14,False
217315,8,False
286278,15,False
270775,16,False
233952,17,True
312316,15,False
356411,18,True
338673,19,False
373224,20,True
379894,21,False
208353,22,False
```

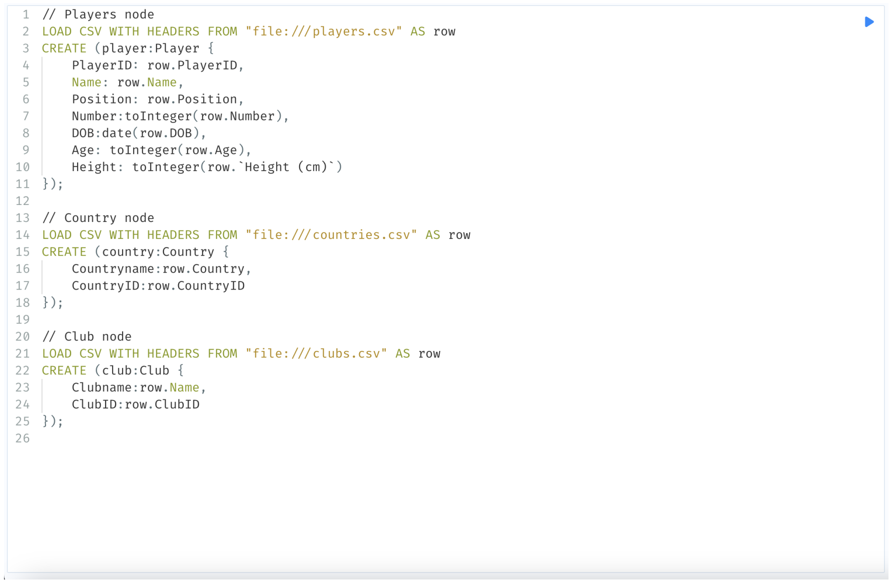
3.3 Importing data into Neoj4

We start off by defining uniqueness constraints for player IDs, club names, and country names to ensure data integrity and consistency.

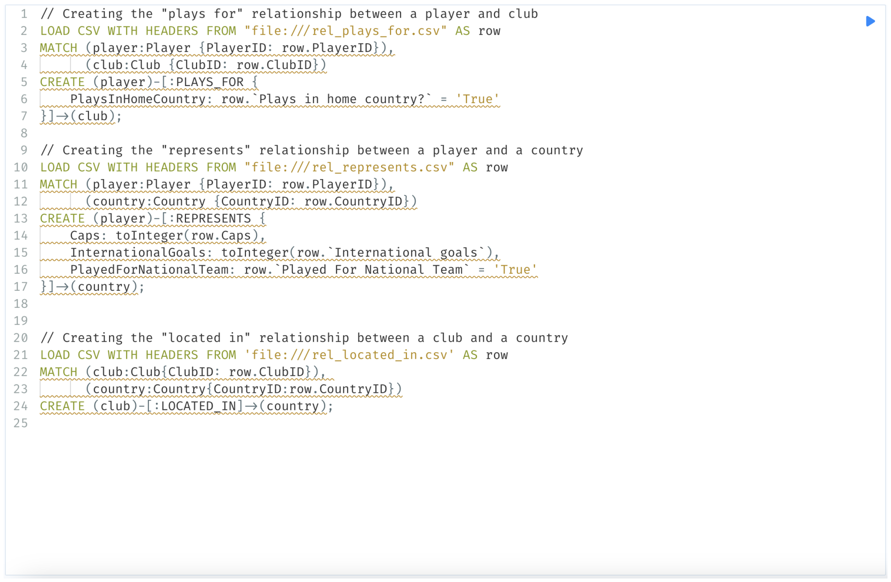


```
CREATE CONSTRAINT FOR (player:Player) REQUIRE player.PlayerID IS UNIQUE;
CREATE CONSTRAINT FOR (club:Club) REQUIRE club.ClubID IS UNIQUE;
CREATE CONSTRAINT FOR (club:Club) REQUIRE club.Clubname IS UNIQUE;
CREATE CONSTRAINT FOR (country:Country) REQUIRE country.CountryID IS UNIQUE;
CREATE CONSTRAINT FOR (country:Country) REQUIRE country.Countryname IS UNIQUE;
```

Next, nodes are loaded into the graph database from CSV files. This includes importing player node, club node and country node . Following the node import, relationships between these nodes are established. For example, the PLAYS_FOR relationship connects players to clubs, the REPRESENTS relationship links players to their national teams, and the LOCATED_IN relationship ties clubs to countries.



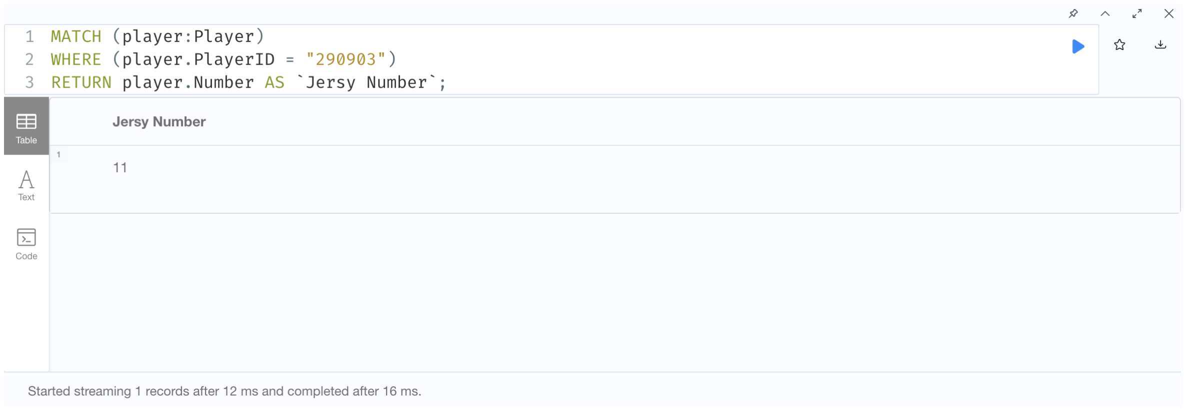
```
// Players node
LOAD CSV WITH HEADERS FROM "file:///players.csv" AS row
CREATE (player:Player {
    PlayerID: row.PlayerID,
    Name: row.Name,
    Position: row.Position,
    Number:toInteger(row.Number),
    DOB:date(row.DOB),
    Age: toInteger(row.Age),
    Height: toInteger(row.`Height (cm)`)
});

// Country node
LOAD CSV WITH HEADERS FROM "file:///countries.csv" AS row
CREATE (country:Country {
    Countryname:row.Country,
    CountryID:row.CountryID
});

// Club node
LOAD CSV WITH HEADERS FROM "file:///clubs.csv" AS row
CREATE (club:Club {
    Clubname:row.Name,
    ClubID:row.ClubID
});
```



```
// Creating the "plays for" relationship between a player and club
LOAD CSV WITH HEADERS FROM "file:///rel_plays_for.csv" AS row
MATCH (player:Player {PlayerID: row.PlayerID}),
      (club:Club {ClubID: row.ClubID})
CREATE (player)-[:PLAYS_FOR {
    PlaysInHomeCountry: row.`Plays in home country?` = 'True'
}]->(club);

// Creating the "represents" relationship between a player and a country
LOAD CSV WITH HEADERS FROM "file:///rel_represents.csv" AS row
MATCH (player:Player {PlayerID: row.PlayerID}),
      (country:Country {CountryID: row.CountryID})
CREATE (player)-[:REPRESENTS {
    Caps: toInteger(row.Caps),
    InternationalGoals: toInteger(row.`International goals`),
    PlayedForNationalTeam: row.`Played For National Team` = 'True'
}]->(country);


// Creating the "located in" relationship between a club and a country
LOAD CSV WITH HEADERS FROM 'file:///rel_located_in.csv' AS row
MATCH (club:Club{ClubID: row.ClubID}),
      (country:Country{CountryID:row.CountryID})
CREATE (club)-[:LOCATED_IN]->(country);
```

4 Query results

Q1 What is the jersey number of the player with a player id of 290903?

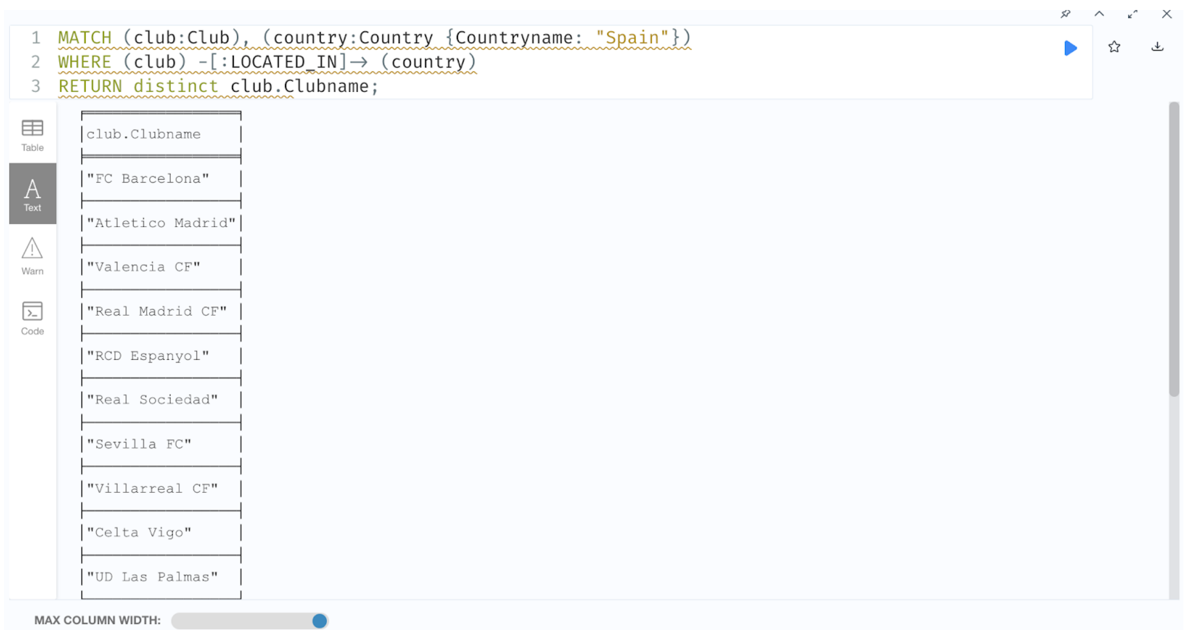


```
MATCH (player:Player)
WHERE (player.PlayerID = "290903")
RETURN player.Number AS `Jersy Number`;
```

```
Jersy Number
11
```

Started streaming 1 records after 12 ms and completed after 16 ms.

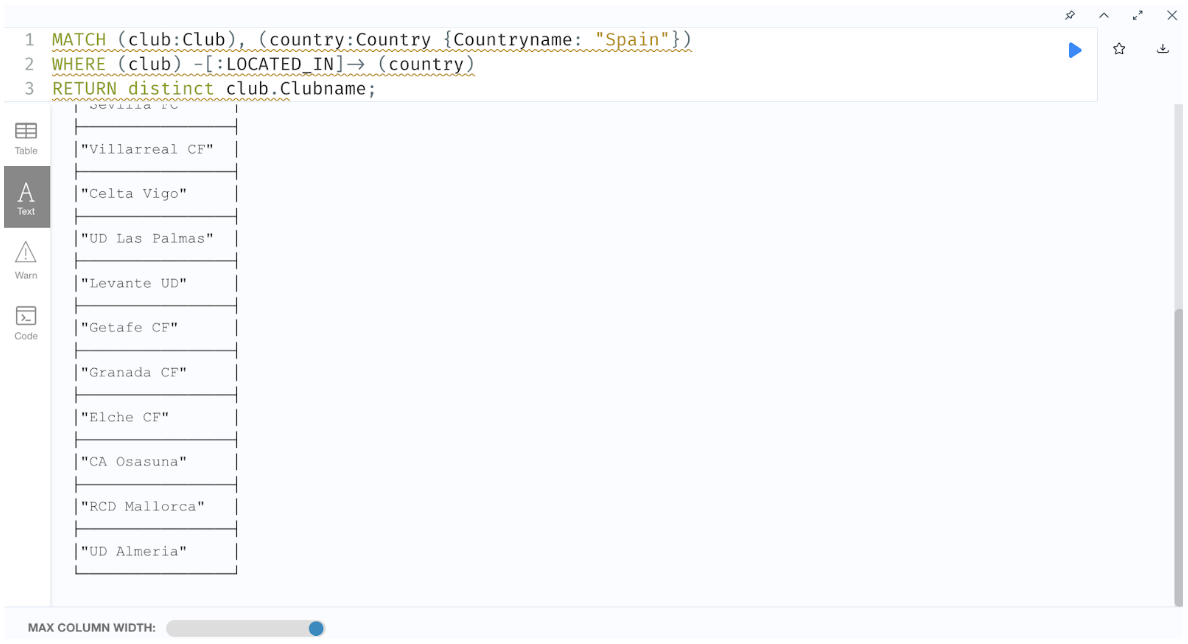
Q2 Which clubs are based in Spain?



```
MATCH (club:Club), (country:Country {Countryname: "Spain"})
WHERE (club) -[:LOCATED_IN]→ (country)
RETURN distinct club.Clubname;
```

```
╒════════════════╕
│club.Clubname   │
╞════════════════╡
│"FC Barcelona"  │
├────────────────┤
│"Atletico Madrid"│
├────────────────┤
│"Valencia CF"   │
├────────────────┤
│"Real Madrid CF"│
├────────────────┤
│"RCD Espanyol"  │
├────────────────┤
│"Real Sociedad" │
├────────────────┤
│"Sevilla FC"    │
├────────────────┤
│"Villarreal CF" │
├────────────────┤
│"Celta Vigo"    │
├────────────────┤
│"UD Las Palmas" │
```

MAX COLUMN WIDTH:

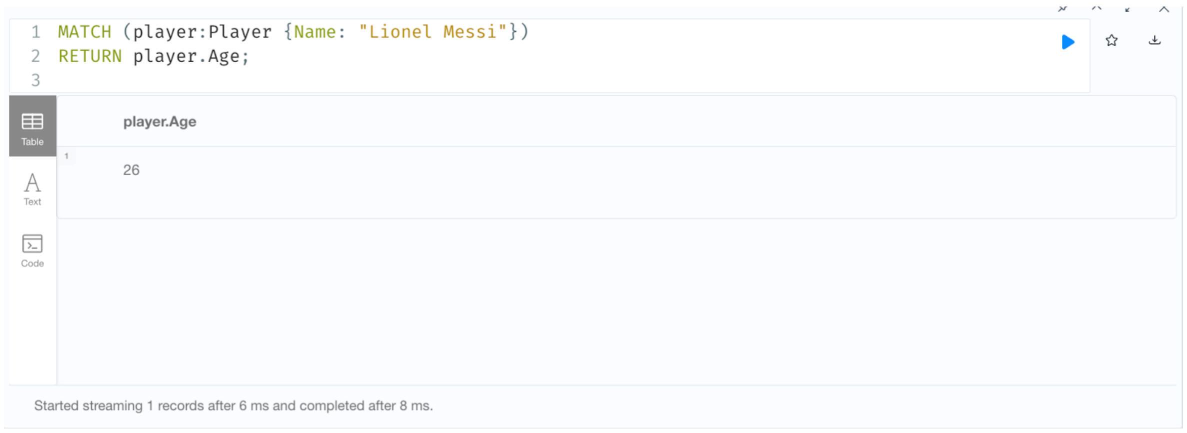


```
MATCH (club:Club), (country:Country {Countryname: "Spain"})
WHERE (club) -[:LOCATED_IN]→ (country)
RETURN distinct club.Clubname;
```

```
│"Sevilla FC"    │
├────────────────┤
│"Villarreal CF" │
├────────────────┤
│"Celta Vigo"    │
├────────────────┤
│"UD Las Palmas" │
├────────────────┤
│"Levante UD"    │
├────────────────┤
│"Getafe CF"     │
├────────────────┤
│"Granada CF"    │
├────────────────┤
│"Elche CF"      │
├────────────────┤
│"CA Osasuna"    │
├────────────────┤
│"RCD Mallorca"  │
├────────────────┤
│"UD Almeria"    │
└────────────────┘
```

MAX COLUMN WIDTH:

Q3 How old is Lionel Messi?

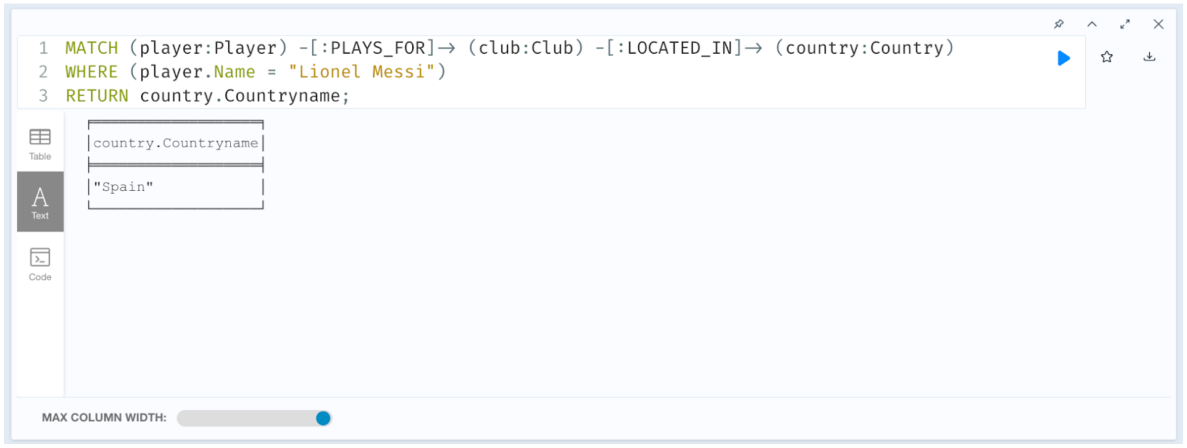


```
MATCH (player:Player {Name: "Lionel Messi"})
RETURN player.Age;
```

```
player.Age
1  26
```

Started streaming 1 records after 6 ms and completed after 8 ms.

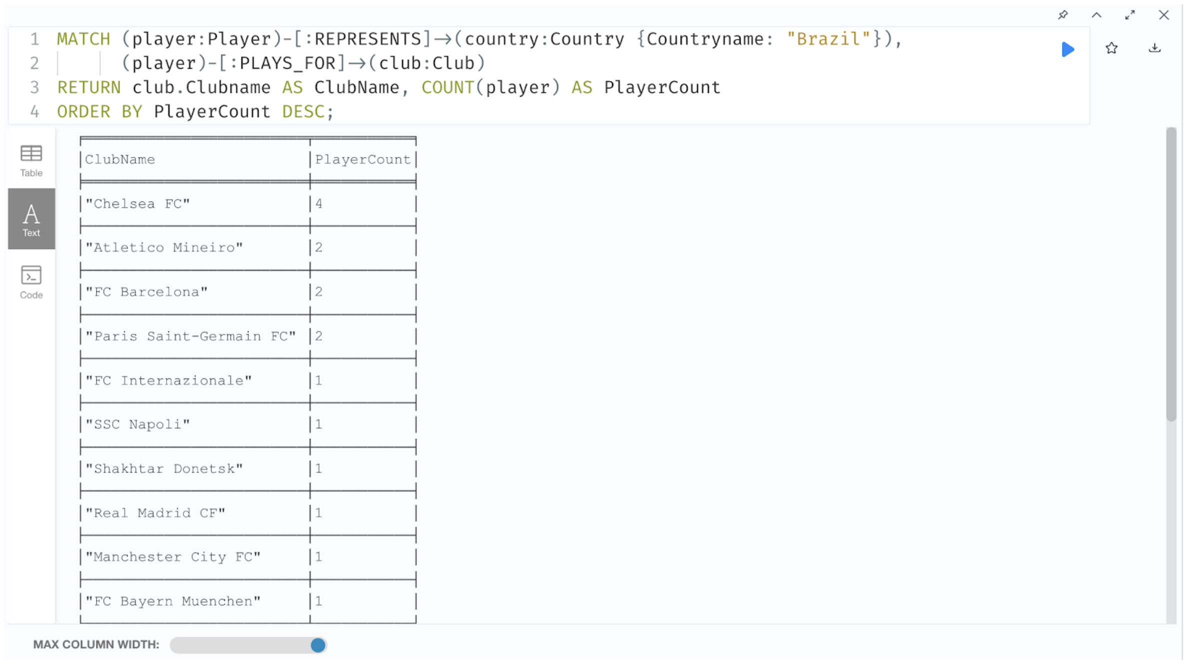
Q4 In which country is the club that Lionel Messi plays for?



```
MATCH (player:Player) -[:PLAYS_FOR]→ (club:Club) -[:LOCATED_IN]→ (country:Country)
WHERE (player.Name = "Lionel Messi")
RETURN country.Countryname;
```

```
╒═══════════════════╕
│country.Countryname│
╞═══════════════════╡
│"Spain"            │
└───────────────────┘
```

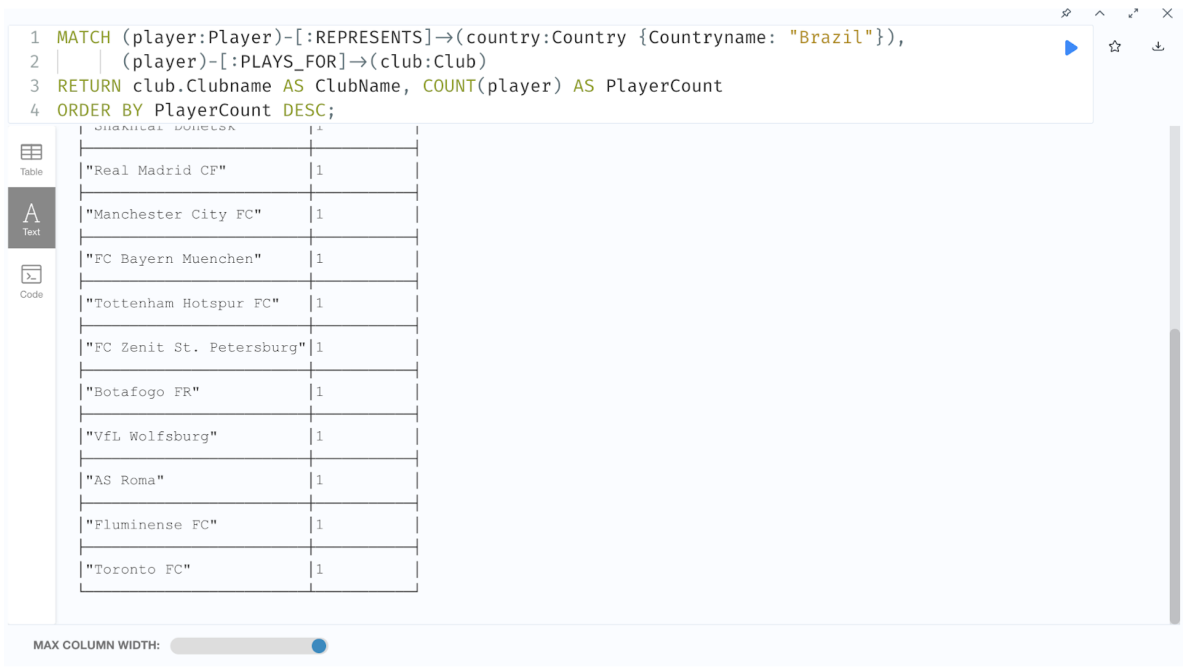
Q5 Find a club that has players from Brazil



```
MATCH (player:Player)-[:REPRESENTS]→(country:Country {Countryname: "Brazil"}),
      (player)-[:PLAYS_FOR]→(club:Club)
RETURN club.Clubname AS ClubName, COUNT(player) AS PlayerCount
ORDER BY PlayerCount DESC;
```

```
╒═════════════════════════╤═══════════╕
│ClubName                 │PlayerCount│
╞═════════════════════════╪═══════════╡
│"Chelsea FC"             │4          │
├─────────────────────────┼───────────┤
│"Atletico Mineiro"       │2          │
├─────────────────────────┼───────────┤
│"FC Barcelona"           │2          │
├─────────────────────────┼───────────┤
│"Paris Saint-Germain FC" │2          │
├─────────────────────────┼───────────┤
│"FC Internazionale"      │1          │
├─────────────────────────┼───────────┤
│"SSC Napoli"             │1          │
├─────────────────────────┼───────────┤
│"Shakhtar Donetsk"       │1          │
├─────────────────────────┼───────────┤
│"Real Madrid CF"         │1          │
├─────────────────────────┼───────────┤
│"Manchester City FC"     │1          │
├─────────────────────────┼───────────┤
│"FC Bayern Muenchen"     │1          │
```

MAX COLUMN WIDTH:

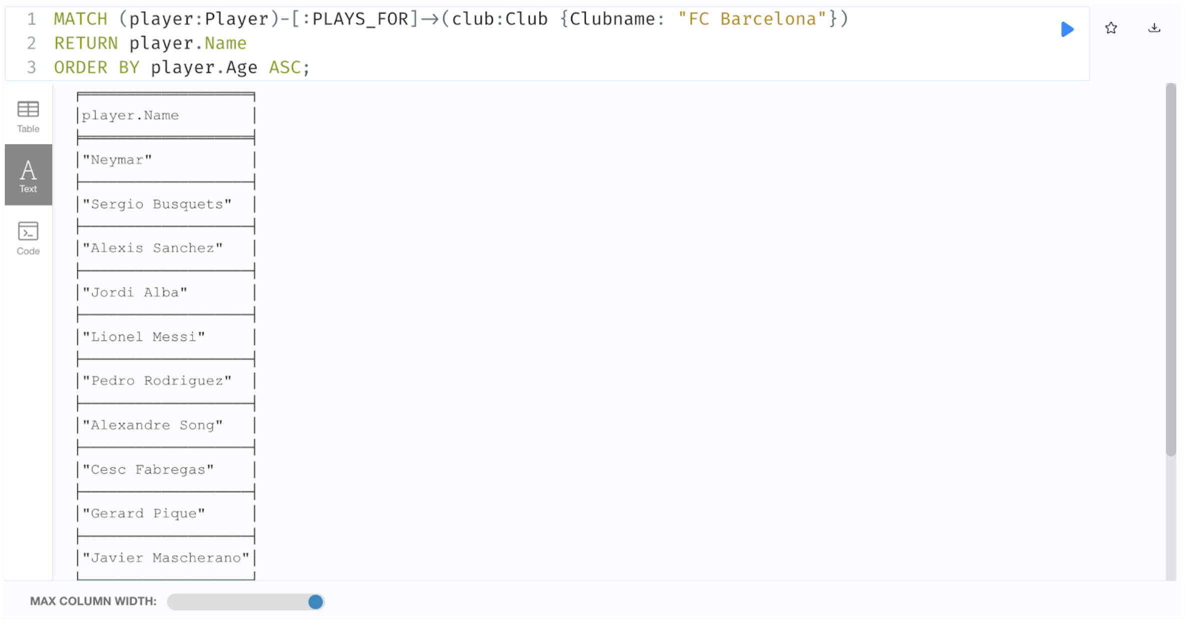


```
MATCH (player:Player)-[:REPRESENTS]→(country:Country {Countryname: "Brazil"}),
      (player)-[:PLAYS_FOR]→(club:Club)
RETURN club.Clubname AS ClubName, COUNT(player) AS PlayerCount
ORDER BY PlayerCount DESC;
```

```
│"Shakhtar Donetsk"       │1          │
├─────────────────────────┼───────────┤
│"Real Madrid CF"         │1          │
├─────────────────────────┼───────────┤
│"Manchester City FC"     │1          │
├─────────────────────────┼───────────┤
│"FC Bayern Muenchen"     │1          │
├─────────────────────────┼───────────┤
│"Tottenham Hotspur FC"   │1          │
├─────────────────────────┼───────────┤
│"FC Zenit St. Petersburg"│1          │
├─────────────────────────┼───────────┤
│"Botafogo FR"            │1          │
├─────────────────────────┼───────────┤
│"VfL Wolfsburg"          │1          │
├─────────────────────────┼───────────┤
│"AS Roma"                │1          │
├─────────────────────────┼───────────┤
│"Fluminense FC"          │1          │
├─────────────────────────┼───────────┤
│"Toronto FC"             │1          │
└─────────────────────────┴───────────┘
```

MAX COLUMN WIDTH:

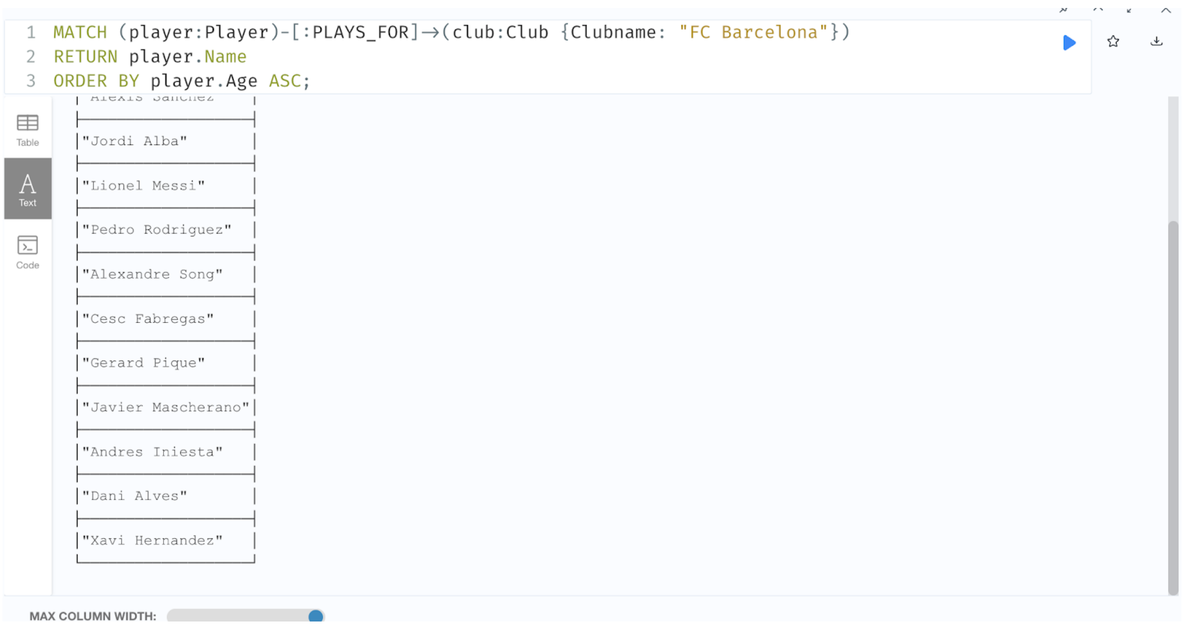
Q6 Find all players play at FC Barcelona, returning in ascending orders of age.



```
MATCH (player:Player)-[:PLAYS_FOR]→(club:Club {Clubname: "FC Barcelona"})
RETURN player.Name
ORDER BY player.Age ASC;
```

```
╒════════════════════╕
│player.Name         │
╞════════════════════╡
│"Neymar"            │
├────────────────────┤
│"Sergio Busquets"   │
├────────────────────┤
│"Alexis Sanchez"    │
├────────────────────┤
│"Jordi Alba"        │
├────────────────────┤
│"Lionel Messi"      │
├────────────────────┤
│"Pedro Rodriguez"   │
├────────────────────┤
│"Alexandre Song"    │
├────────────────────┤
│"Cesc Fabregas"     │
├────────────────────┤
│"Gerard Pique"      │
├────────────────────┤
│"Javier Mascherano" │
└────────────────────┘
```

MAX COLUMN WIDTH:

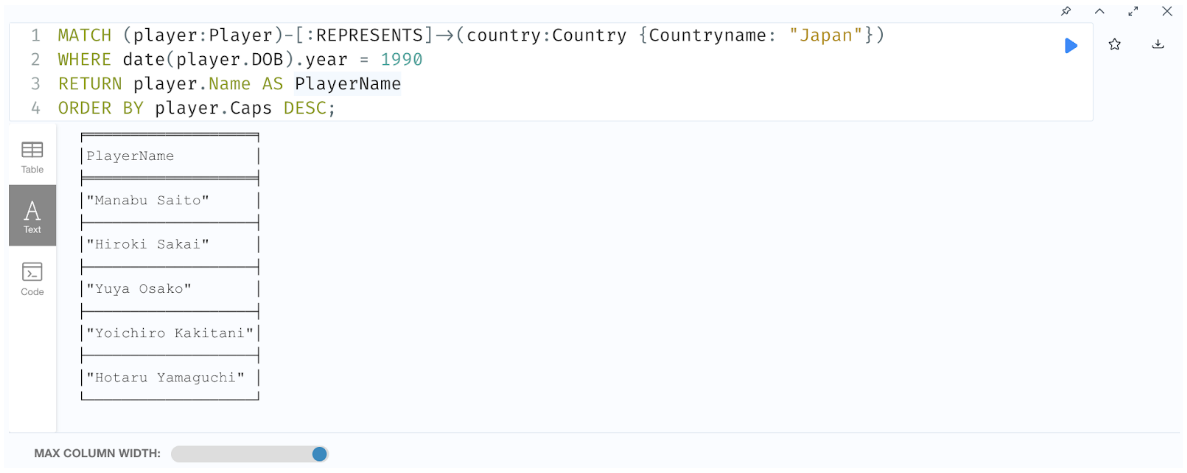


```
MATCH (player:Player)-[:PLAYS_FOR]→(club:Club {Clubname: "FC Barcelona"})
RETURN player.Name
ORDER BY player.Age ASC;
```

```
│"Alexis Sanchez"    │
├────────────────────┤
│"Jordi Alba"        │
├────────────────────┤
│"Lionel Messi"      │
├────────────────────┤
│"Pedro Rodriguez"   │
├────────────────────┤
│"Alexandre Song"    │
├────────────────────┤
│"Cesc Fabregas"     │
├────────────────────┤
│"Gerard Pique"      │
├────────────────────┤
│"Javier Mascherano" │
├────────────────────┤
│"Andres Iniesta"    │
├────────────────────┤
│"Dani Alves"        │
├────────────────────┤
│"Xavi Hernandez"    │
└────────────────────┘
```

MAX COLUMN WIDTH:

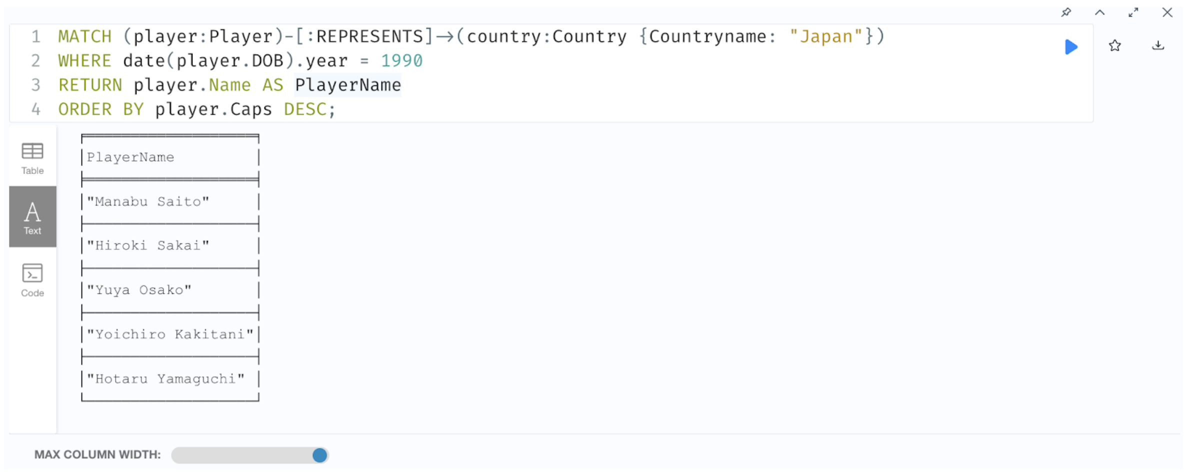
Q7 Find all defender players the national team of Spain, returning in descending order of caps.



```
MATCH (player:Player)-[:REPRESENTS]→(country:Country {Countryname: "Japan"})
WHERE date(player.DOB).year = 1990
RETURN player.Name AS PlayerName
ORDER BY player.Caps DESC;
```

```
╒═══════════════════╕
│PlayerName         │
╞═══════════════════╡
│"Manabu Saito"     │
├───────────────────┤
│"Hiroki Sakai"     │
├───────────────────┤
│"Yuya Osako"       │
├───────────────────┤
│"Yoichiro Kakitani"│
├───────────────────┤
│"Hotaru Yamaguchi" │
└───────────────────┘
```

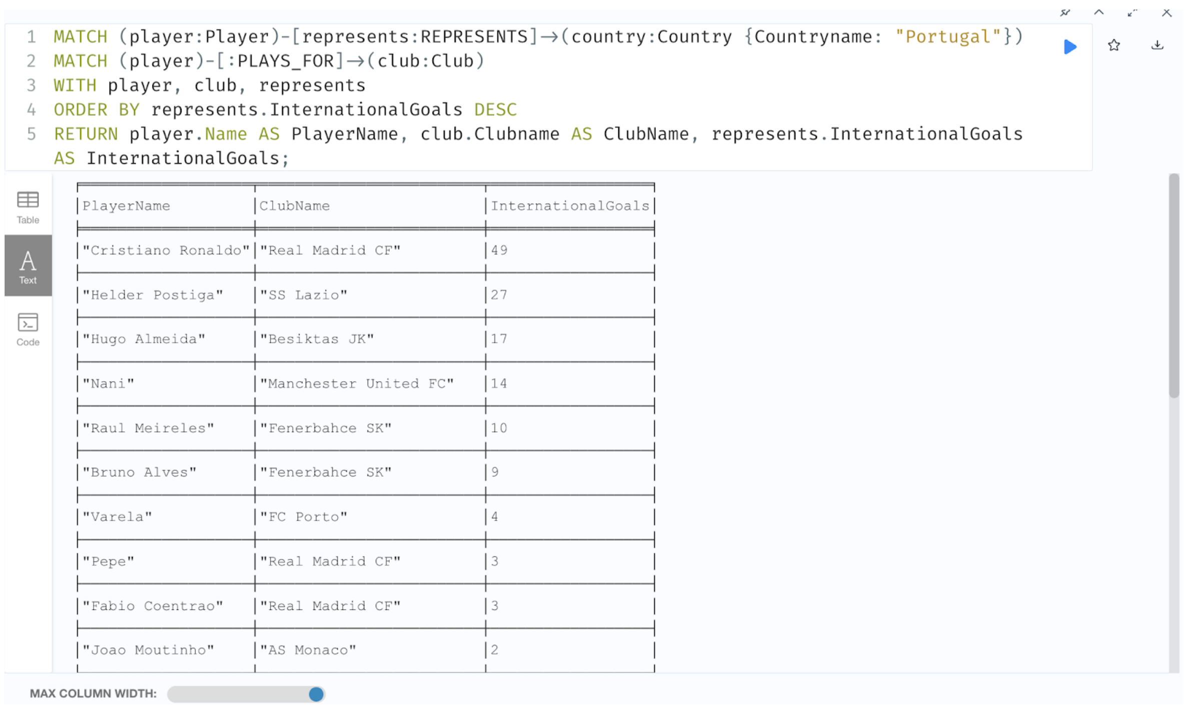
Q8 Find all players born in 1990 and in the national team of Japan, returning in descending order of caps.



```
MATCH (player:Player)-[:REPRESENTS]→(country:Country {Countryname: "Japan"})
WHERE date(player.DOB).year = 1990
RETURN player.Name AS PlayerName
ORDER BY player.Caps DESC;
```

```
╒══════════════════╕
│PlayerName        │
╞══════════════════╡
│"Manabu Saito"    │
├──────────────────┤
│"Hiroki Sakai"    │
├──────────────────┤
│"Yuya Osako"      │
├──────────────────┤
│"Yoichiro Kakitani"│
├──────────────────┤
│"Hotaru Yamaguchi"│
└──────────────────┘
```

Q9 Find the players that belongs to the same club in national team of Portugal, returning in descending order of international goals



```
MATCH (player:Player)-[represents:REPRESENTS]→(country:Country {Countryname: "Portugal"})
MATCH (player)-[:PLAYS_FOR]→(club:Club)
WITH player, club, represents
ORDER BY represents.InternationalGoals DESC
RETURN player.Name AS PlayerName, club.Clubname AS ClubName, represents.InternationalGoals
AS InternationalGoals;
```

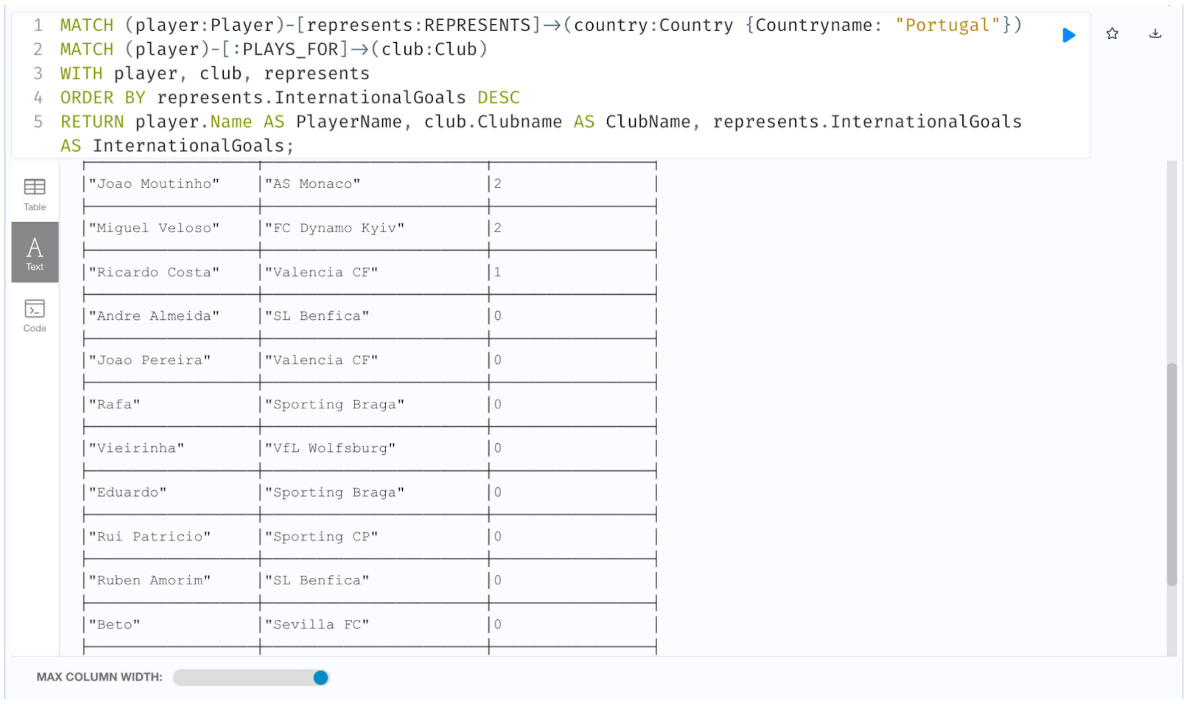
Table

Text

Code

```
|PlayerName         |ClubName                  |InternationalGoals|
|"Cristiano Ronaldo"|"Real Madrid CF"          |49                |
|"Helder Postiga"   |"SS Lazio"                |27                |
|"Hugo Almeida"     |"Besiktas JK"             |17                |
|"Nani"             |"Manchester United FC"    |14                |
|"Raul Meireles"    |"Fenerbahce SK"           |10                |
|"Bruno Alves"      |"Fenerbahce SK"           |9                 |
|"Varela"           |"FC Porto"                |4                 |
|"Pepe"             |"Real Madrid CF"          |3                 |
|"Fabio Coentrao"   |"Real Madrid CF"          |3                 |
|"Joao Moutinho"    |"AS Monaco"               |2                 |
```

MAX COLUMN WIDTH:



```
MATCH (player:Player)-[represents:REPRESENTS]→(country:Country {Countryname: "Portugal"})
MATCH (player)-[:PLAYS_FOR]→(club:Club)
WITH player, club, represents
ORDER BY represents.InternationalGoals DESC
RETURN player.Name AS PlayerName, club.Clubname AS ClubName, represents.InternationalGoals
AS InternationalGoals;
```

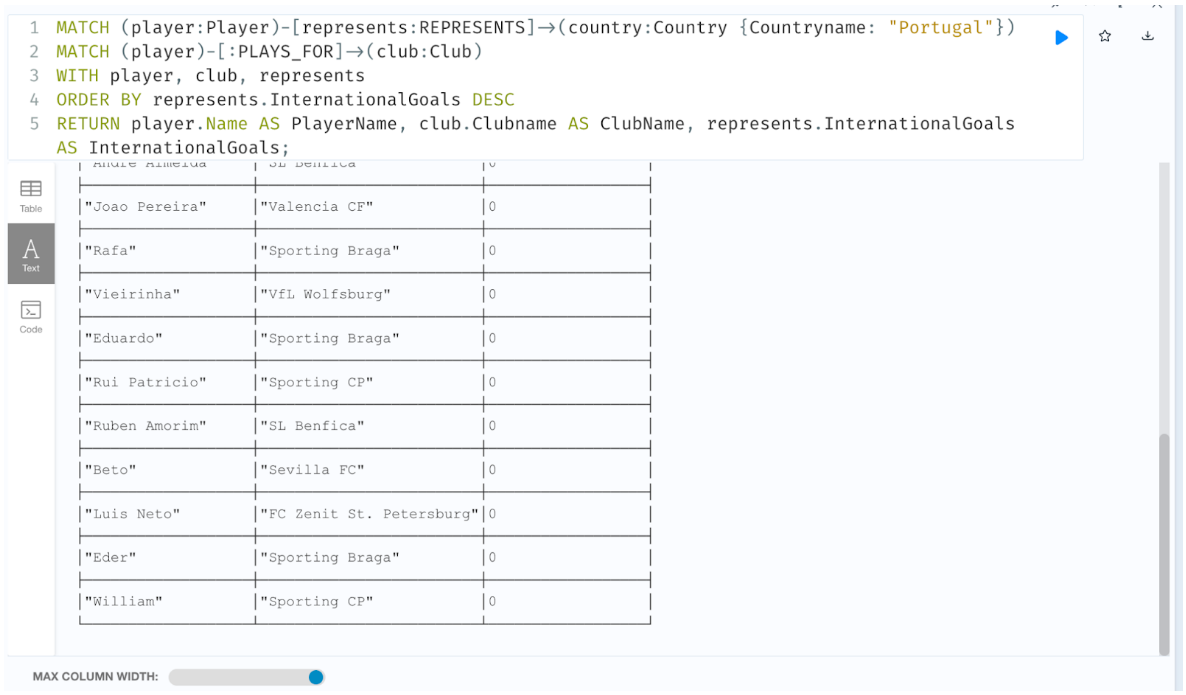
Table

Text

Code

```
|"Joao Moutinho"    |"AS Monaco"               |2                 |
|"Miguel Veloso"    |"FC Dynamo Kyiv"          |2                 |
|"Ricardo Costa"    |"Valencia CF"             |1                 |
|"Andre Almeida"    |"SL Benfica"              |0                 |
|"Joao Pereira"     |"Valencia CF"             |0                 |
|"Rafa"             |"Sporting Braga"          |0                 |
|"Vieirinha"        |"VfL Wolfsburg"           |0                 |
|"Eduardo"          |"Sporting Braga"          |0                 |
|"Rui Patricio"     |"Sporting CP"             |0                 |
|"Ruben Amorim"     |"SL Benfica"              |0                 |
|"Beto"             |"Sevilla FC"              |0                 |
```

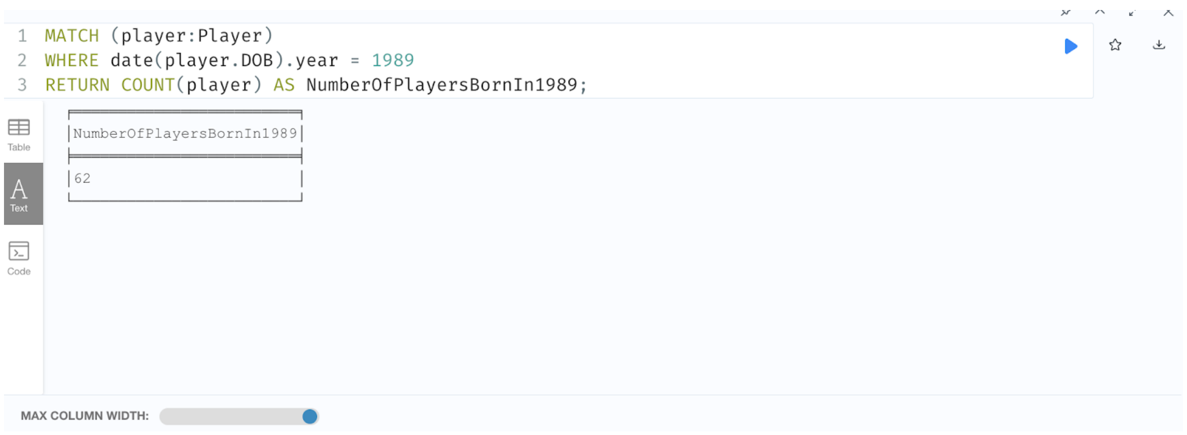
MAX COLUMN WIDTH:



```
MATCH (player:Player)-[represents:REPRESENTS]→(country:Country {Countryname: "Portugal"})
MATCH (player)-[:PLAYS_FOR]→(club:Club)
WITH player, club, represents
ORDER BY represents.InternationalGoals DESC
RETURN player.Name AS PlayerName, club.Clubname AS ClubName, represents.InternationalGoals
AS InternationalGoals;
```

```
│"Andre Almeida"     │"SL Benfica"              │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Joao Pereira"      │"Valencia CF"             │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Rafa"              │"Sporting Braga"          │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Vieirinha"         │"VfL Wolfsburg"           │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Eduardo"           │"Sporting Braga"          │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Rui Patricio"      │"Sporting CP"             │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Ruben Amorim"      │"SL Benfica"              │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Beto"              │"Sevilla FC"              │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Luis Neto"         │"FC Zenit St. Petersburg"│0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"Eder"              │"Sporting Braga"          │0                  │
├────────────────────┼──────────────────────────┼───────────────────┤
│"William"           │"Sporting CP"             │0                  │
└────────────────────┴──────────────────────────┴───────────────────┘
```

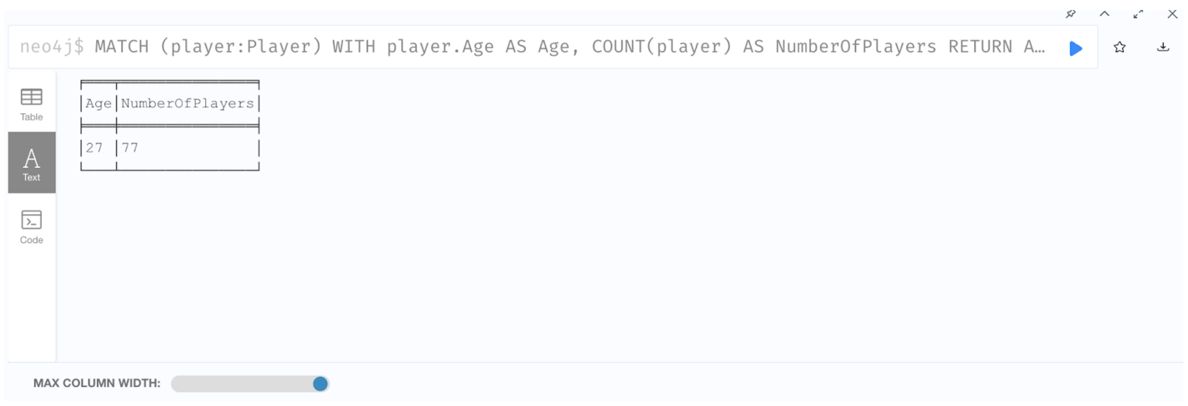
Q10 Count how many players are born in 1989



```
MATCH (player:Player)
WHERE date(player.DOB).year = 1989
RETURN COUNT(player) AS NumberOfPlayersBornIn1989;
```

```
╒═════════════════════════╕
│NumberOfPlayersBornIn1989│
╞═════════════════════════╡
│62                       │
└─────────────────────────┘
```

Q11 Which age has the highest participation in the 2014 FIFA World Cup?

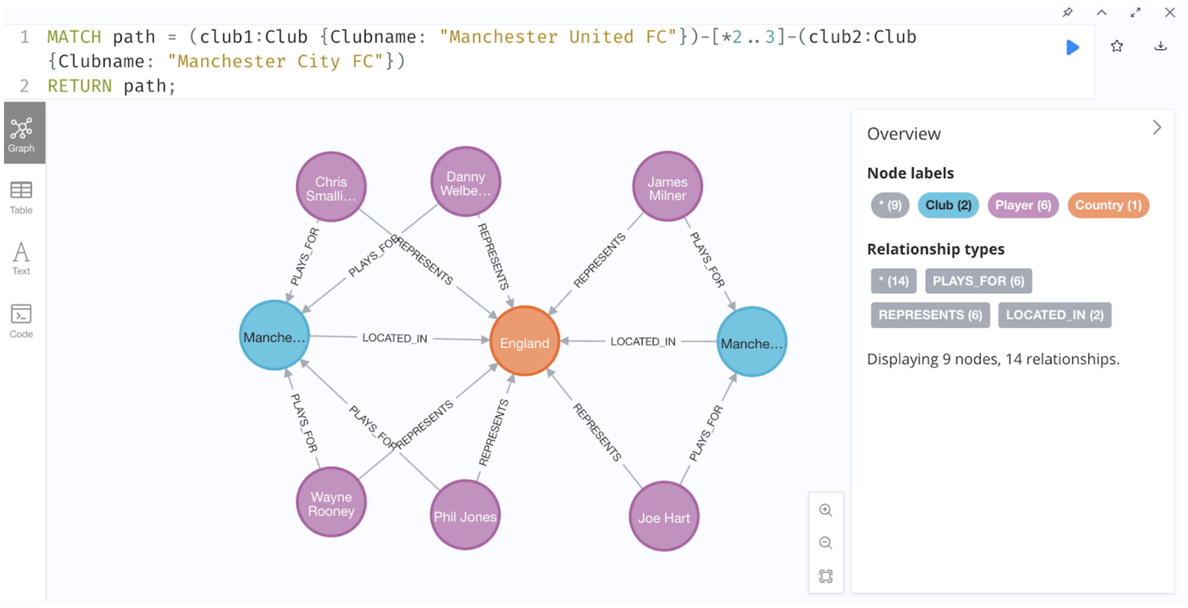


```
neo4j$ MATCH (player:Player) WITH player.Age AS Age, COUNT(player) AS NumberOfPlayers RETURN A…
```

```
╒═══╤═══════════════╕
│Age│NumberOfPlayers│
╞═══╪═══════════════╡
│27 │77             │
└───┴───────────────┘
```

Q12 Find the path with a length of 2 or 3 between Manchester United FC and Manchester City FC.

Manchester United FC: right node Manchester City FC: left node



```
MATCH path = (club1:Club {Clubname: "Manchester United FC"})-[*2..3]-(club2:Club
{Clubname: "Manchester City FC"})
RETURN path;
```

Overview

Node labels

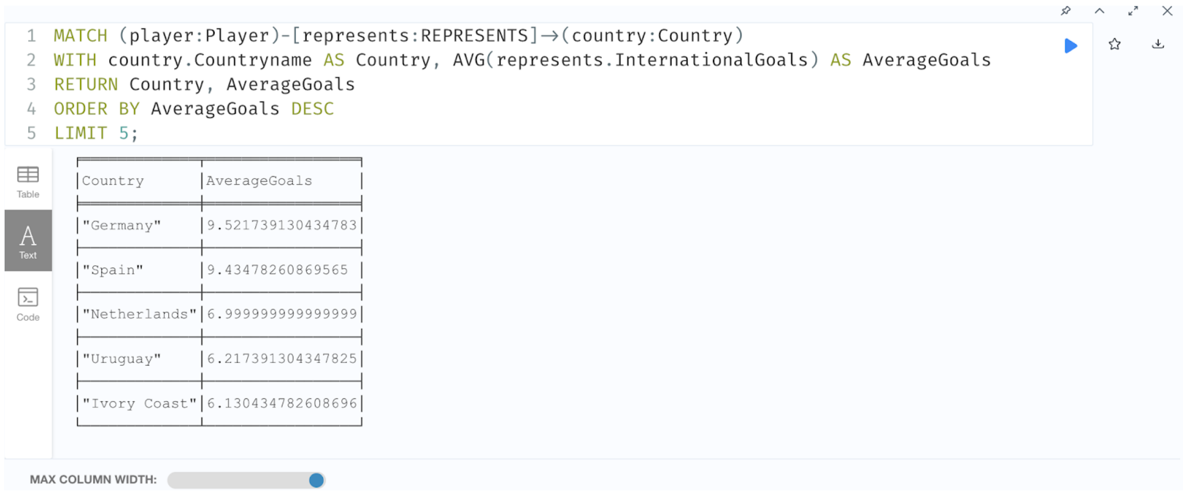
* (9) Club (2) Player (6) Country (1)

Relationship types

* (14) PLAYS_FOR (6) REPRESENTS (6) LOCATED_IN (2)

Displaying 9 nodes, 14 relationships.

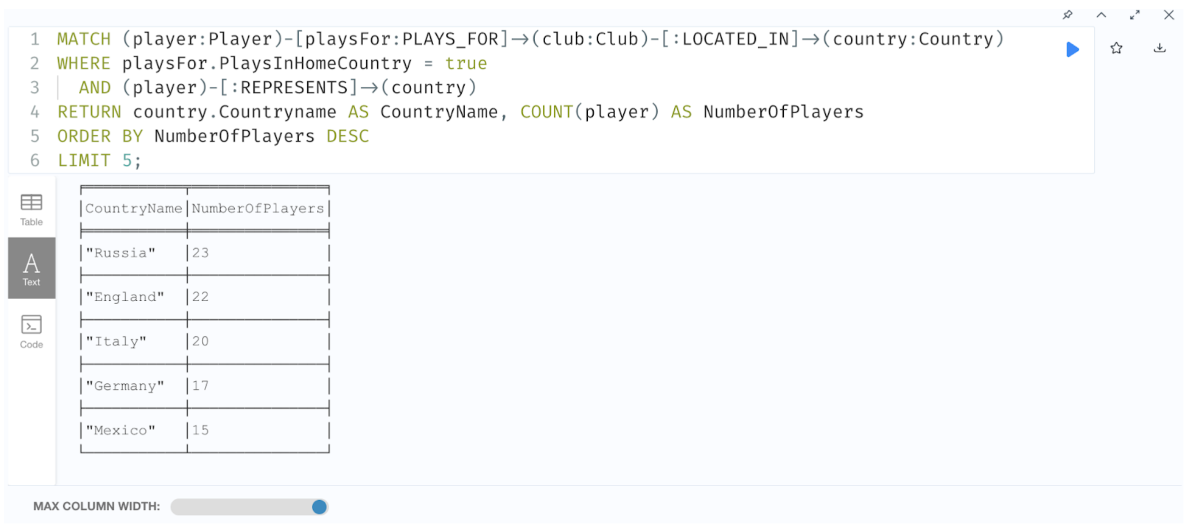
Q13 Find the top 5 countries with players who have the highest average number of international goals. Return the countries and their average international goals in descending order.



```
MATCH (player:Player)-[represents:REPRESENTS]→(country:Country)
WITH country.Countryname AS Country, AVG(represents.InternationalGoals) AS AverageGoals
RETURN Country, AverageGoals
ORDER BY AverageGoals DESC
LIMIT 5;
```

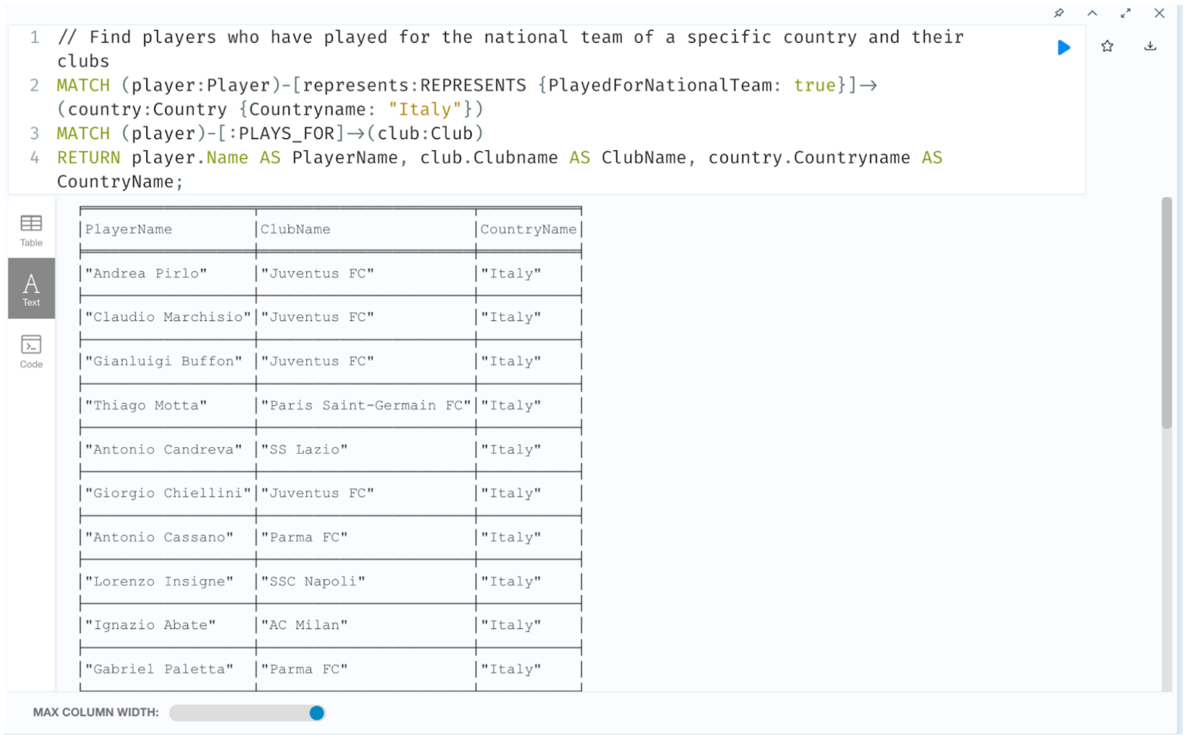
```
╒═════════════╤═════════════════╕
│Country      │AverageGoals     │
╞═════════════╪═════════════════╡
│"Germany"    │9.521739130434783│
├─────────────┼─────────────────┤
│"Spain"      │9.43478260869565 │
├─────────────┼─────────────────┤
│"Netherlands"│6.999999999999999│
├─────────────┼─────────────────┤
│"Uruguay"    │6.217391304347825│
├─────────────┼─────────────────┤
│"Ivory Coast"│6.130434782608696│
└─────────────┴─────────────────┘
```

Q14 Find the top 5 countries with the most players who play for their home country's club



```
MATCH (player:Player)-[playsFor:PLAYS_FOR]→(club:Club)-[:LOCATED_IN]→(country:Country)
WHERE playsFor.PlaysInHomeCountry = true
  AND (player)-[:REPRESENTS]→(country)
RETURN country.Countryname AS CountryName, COUNT(player) AS NumberOfPlayers
ORDER BY NumberOfPlayers DESC
LIMIT 5;
```

```
╒═══════════╤═══════════════╕
│CountryName│NumberOfPlayers│
╞═══════════╪═══════════════╡
│"Russia"   │23             │
├───────────┼───────────────┤
│"England"  │22             │
├───────────┼───────────────┤
│"Italy"    │20             │
├───────────┼───────────────┤
│"Germany"  │17             │
├───────────┼───────────────┤
│"Mexico"   │15             │
└───────────┴───────────────┘
```



```
// Find players who have played for the national team of a specific country and their clubs
MATCH (player:Player)-[represents:REPRESENTS {PlayedForNationalTeam: true}]→(country:Country {Countryname: "Italy"})
MATCH (player)-[:PLAYS_FOR]→(club:Club)
RETURN player.Name AS PlayerName, club.Clubname AS ClubName, country.Countryname AS CountryName;
```

```
│PlayerName         │ClubName               │CountryName│
│"Andrea Pirlo"     │"Juventus FC"          │"Italy"    │
│"Claudio Marchisio"│"Juventus FC"          │"Italy"    │
│"Gianluigi Buffon" │"Juventus FC"          │"Italy"    │
│"Thiago Motta"     │"Paris Saint-Germain FC"│"Italy"    │
│"Antonio Candreva" │"SS Lazio"             │"Italy"    │
│"Giorgio Chiellini"│"Juventus FC"          │"Italy"    │
│"Antonio Cassano"  │"Parma FC"             │"Italy"    │
│"Lorenzo Insigne"  │"SSC Napoli"           │"Italy"    │
│"Ignazio Abate"    │"AC Milan"             │"Italy"    │
│"Gabriel Paletta"  │"Parma FC"             │"Italy"    │
```

Figure 3: Enter Caption

Q15 Find players who have played for the national team of Italy and their clubs

5 Comparing Graph and Relational Databases

A discussion on graph databases' capabilities as compared to relational databases, highlighting what's achievable in graph databases that's challenging or not feasible in relational databases.

RDBMSs have long been used for data storage and retrieval, but the complexity of modern applications has revealed their limitations in managing sophisticated relationships at-scale. This has resulted in a shift in attention towards graph database management systems (GDMS), which use graph structures - vertices, edges, and properties - to represent and store data (Cheng et al., 2019).

Relational databases use a tabular structure and rely on defining unique IDs (foreign keys) to relate data across different tables, making complex queries inefficient (Timón-Reina, 2021). In densely connected data dominated by many-to-many relationships, querying the database may require chaining together many expensive, performance-hampering JOIN operations (Hsu et al., 2014). In contrast, graph databases like Neo4j offer a more efficient way to model and retrieve relationship data. Relationships can be easily queried directly through connections, making the process faster and less resource-intensive when compared with a relational model (Medhi & Baruah, 2017). Because of this graph databases are preferred over relational databases when it comes to fraud detection. SQL table joins become computationally expensive where various data types like accounts, individuals and transactions are analyzed (Henderson, 2020).

Graph databases also outperform relational databases when storing and retrieving highly connected data. For instance, updating data in a relational database requires visiting two records each time, while in a graph database, only one record needs to be visited, making the process more efficient (Lazarska & Siedlecka-Lamch, 2019).

In terms of query performance, Kotiranta and colleagues (2022) suggest that, as a baseline, Neo4J is often at least three times faster than modern relational databases. In instances where an aggregated value was calculated for the given entity however, Neo4J could be as much as 200 times faster than MySQL.

The schema-less nature of GDBMSs provide greater flexibility, allowing for more dynamic approaches to data modeling (Timón-Reina, 2021). This flexibility is advantageous in scenarios where the data structure is not static or predefined. Relational databases, on the other hand, require a fixed schema, making them less adaptable to changes in data structure. This is beneficial especially in the field of biomedical research, where there are large volumes of densely interconnected data across various domains.

Lazarska & Siedlecka-Lamch (2019) conducted a comparative analysis of relational and graph databases using a telephone connection database. Their study found that query execution times in the graph database were significantly faster than in the relational database. Specifically, Neo4j completed a query on 100,000 records in just 2.2 seconds, whereas Oracle took 111.3 seconds for the same task. Additionally, Neo4j scaled efficiently, increasing the number of visited objects tenfold in line with a tenfold increase in the number of result records.

Despite the increased efficiency they may provide when representing highly connected systems, graph databases do currently come with a number of unique security concerns. One such issue is a lack of built-in multiuser support - a staple of relational databases (Vicknair et al., 2010). While Neo4j does provide basic ACL security, MySQL and other relational databases offer extensive ACL-based security (Batra and Tyag, 2012).

As an emerging technology, graph databases offer performance and scalability advantages over traditional relational databases, especially with data representing highly interrelated systems. While graph databases still have a way to go in matching the established security infrastructures found in older RDMS implementations, every indication is that this approach to database design will continue to grow in relevance.

6 Application of Graph Data Science

The field of graph analytics has been around for a long time. The general idea is to create a database of things connecting to other things. Graphs are regularly used to enhance search capabilities, recommend products to shoppers on e-commerce sites, detect fraud, or to navigate the shortest route from point A to point B (Sullivan, 2020).

One such application of graph data science (GDS) is fraud detection, an ever-present concern of the finance industry. Nowadays, in order to not be detected by bank's fraud algorithms, malicious actors may perform many typical, non-suspicious transactions with multiple accounts in an attempt to have their account classified favorably prior to opening accounts nefarious actions. Such behavior can be hard to be detected with conventional methods, but graph database-centered approaches have shown promise in detecting these cases. While machine learning methods prove effective in fraud detection, they often overlook interconnections between data-sets. Graph databases address these issues by allowing entity-link analysis, uncovering connections between data points (Magomedov et al., 2018). Graph databases are proven to excel in handling complex fraud schemes such as money laundering, and can analyze high volumes of relationship-defined data in real time, enabling the detection of suspicious activities across multiple accounts and transactions (Henderson, 2020).

More recently, graph databases have also seen use tracking the spread of infection during the COVID-19 pandemic (Alguliyev et al., 2020). The ability to model complex relationships and dynamic factors provided valuable insights for managing the public health crisis by aiding decision-making.

Alquaisi et al. (2020) developed a graph-based ML model for the detection COVID-19 based on symptoms. GDS was utilized to enhance the performance of machine learning models in predicting COVID-19 infection. The logistic regression (LR) and random forest (RF) models were trained on 5,434 nodes representing instances in the COVID-19 dataset. The enhanced LR and RF models demonstrated superior performance, highlighting the effectiveness of integrating graph-based features.

Graph analytics applications are not just for interpreting data; they can also predict how data will change shortly. A paper by Pandat, et al. (2023) uses graph analytics to forecast future changes in the centrality and community of bird data.

References

Alguliyev, R., Aliguliyev, R., & Yusifov, F. (2020). Graph modelling for tracking the COVID-19 pandemic spread. *Infectious Disease Modelling*, *6*, 112–122. https://doi.org/10.1016/j.idm.2020.12.002

Alqaissi, E., Alotaibi, F., & Ramzan, M. S. (2023). Graph data science and machine learning for the detection of COVID-19 infection from symptoms. *PeerJ. Computer science*, *9*, e1333. https://doi.org/10.7717/peerj-cs.1333

Batra, S., Tyagi, C. (2012). Comparative Analysis of Relational and Graph
Databases. *International Journal of Soft Computing and Engineering (IJSCE),* 2(2).

Henderson, R. (2020). Using graph databases to detect financial fraud. *Computer Fraud & Security*, *2020*(7), 6-10.

Hsu, J. C., Hsu, C. H., Chen, S. C., & Chung, Y. C. (2014, July). Correlation aware technique for SQL to NoSQL transformation. In *2014 7th international conference on Ubi-media computing and workshops* (pp. 43-46). IEEE.

Kotiranta, P., Junkkari, M., & Nummenmaa, J. (2022). Performance of graph and relational databases in complex queries. *Applied sciences*, *12*(13), 6490.

Lazarska, M., & Siedlecka-Lamch, O. (2019, November). Comparative study of relational and graph databases. In *2019 IEEE 15th International Scientific Conference on Informatics* (pp. 000363-000370). IEEE.

Magomedov, S., Pavelyev, S., Ivanova, I., Dobrotvorsky, A., Khrestina, M., & Yusubaliev, T. (2018). Anomaly detection with machine learning and graph databases in fraud management. *International Journal of Advanced Computer Science and Applications*, *9*(11).

Medhi, S., & Baruah, H. K. (2017). Relational database and graph database: A comparative analysis. *Journal of process management and new technologies*, *5*(2), 1-9.

Pandat, A., Bhise, M., & Srivastava, S. (2023). Graph Analytics for Avian Science Data.

Patil, A., Mahajan, S., Menpara, J., Wagle, S., Pareek, P., & Kotecha, K. (2024). Enhancing fraud detection in banking by integration of graph databases with machine learning. *MethodsX*, *12*, 102683. https://doi.org/10.1016/j.mex.2024.102683

Sandell, J., Asplund, E., Ayele, W. Y., & Duneld, M. (2024). Performance Comparison Analysis of ArangoDB, MySQL, and Neo4j: An Experimental Study of Querying Connected Data. *arXiv preprint arXiv:2401.17482*.

Timón-Reina, S., Rincón, M., & Martínez-Tomás, R. (2021). An overview of graph databases and their applications in the biomedical domain. *Database*, *2021*, baab026.