**CSY 3024 Assigment 1**

**Aayush Shrestha (21422064)**

**Video link:** <https://video.northampton.ac.uk/media/21422064_csy3024_assignment1_video_demo/1_mbplqzx9>

**Report:**

Graph databases are the non-tabular database, stores, and present data in form graphical form of nodes, relationships, and properties. It is the powerful database, that demonstrates the relationship between nodes or data. Graph database makes easy to find the relationship between two data or objects which make complex query to easier, the pattern describes the flow of data.(Mark Needham, 2019) Here in assignment, EPL data set was observed carefully and created, which is the important step for analysis, design, and development of model.(Fernandes & Bernardino, 2018). According to (Fernandes & Bernardino, 2018) node can be treated as object, label as entity. Proper relationships and nodes can result in high performance and scalability of database. (Mark Needham, 2019) stated, while importing data, similar data can be grouped by label and relationship helps to design and create patterns and flow of database.

By observing “EPL\_matches.csv” dataset, I have created nodes and relationships. First, I created Team node with label separating home team and away team of matches. I have created all referee of matches. While creating node, I ensure that home team, away team, referee should be unique so I created unique constraint for name of each nodes;

**Code:**

// creating constraint

CREATE CONSTRAINT hometeam\_is\_unique FOR (team:HomeTeam) REQUIRE team.name IS UNIQUE;

// creating unique constraint for away team

CREATE CONSTRAINT awayteam\_is\_unique FOR (awayteam:AwayTeam) REQUIRE awayteam.name IS UNIQUE;

// for referee

CREATE CONSTRAINT referee\_is\_unique FOR (referee:Referee) REQUIRE referee.name IS UNIQUE;

// This code lods csv file from epl dataset

LOAD CSV WITH HEADERS FROM

"file:///EPL\_matches.csv" as row

// creates one Team node and HomeTeam and AwayTeam node

MERGE(team1:Team:HomeTeam{name: row.HomeTeam})

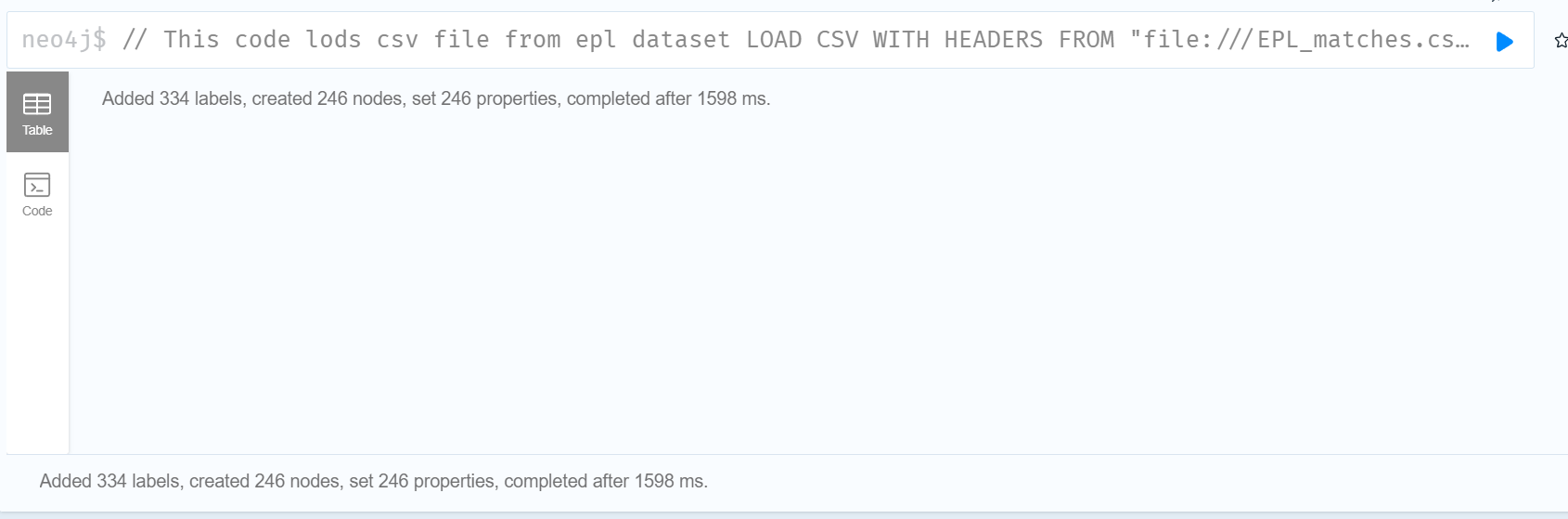
// Creates or find AwayTeam node with name from dataset

MERGE(team2:Team:AwayTeam{name: row.AwayTeam})

// Creates all referees of matches

MERGE(referee:Referee{name: row.Referee});

**Result:**



After that, I created all matches node and created the relationship “HANDLED\_BY” with referee. There is also relationship “PLAYED” with match and home team and away team, that stores full time and halftime goal of that team.

**Code:**

LOAD CSV WITH HEADERS FROM

"file:///EPL\_matches.csv" as row

// This find the HTR and FTR column from the dataset and

// set the values according to given condition

WITH row,

(CASE row.HTR

WHEN "H" THEN row.HomeTeam

WHEN "A" THEN row.AwayTeam

ELSE "DRAW"

END) AS half\_time,

// case for full time result

// If "H" in dataset then store home team name

(CASE row.FTR

WHEN "H" THEN row.HomeTeam

WHEN "A" THEN row.AwayTeam

ELSE "DRAW"

END) AS full\_time

// Find the homeTeam, awayTeam created before

MATCH(homeTeam:HomeTeam{name:row.HomeTeam})

MATCH(awayTeam:AwayTeam{name:row.AwayTeam})

// Find Referees

MATCH(referee:Referee{name:row.Referee})

// Create or find matches according to dataset storing following properties

// with relationship :HANDLED\_BY with referee

MERGE (m:Match{season: row.Season, dateTime: row.DateTime, halfTimeResult: half\_time, fullTimeResult: full\_time})-[:HANDLED\_BY]->(referee)

//Creates or find the relationship between homeTeam, awayteam and match

MERGE (homeTeam)-[:PLAYED{halfTimeGoal: toInteger(row.HTHG), fullTimeGoal: toInteger(row.FTHG)}]->(m)<-[:PLAYED{halfTimeGoal: toInteger(row.HTAG), fullTimeGoal: toInteger(row.FTAG)}]-(awayTeam);

**Result:**



Match results are stored in property halfTimeResult and fullTimeResult.

Loading all required data from dataset will create following model for this assignment, which can be viewed by following code.

**Code:**

// Finds homeTeam and away team

MATCH (t1:HomeTeam)-[:PLAYED]->(m:Match)<-[:PLAYED]-(t2:AwayTeam)

// find referee of matches

MATCH(m)-[:HANDLED\_BY]->(r:Referee)

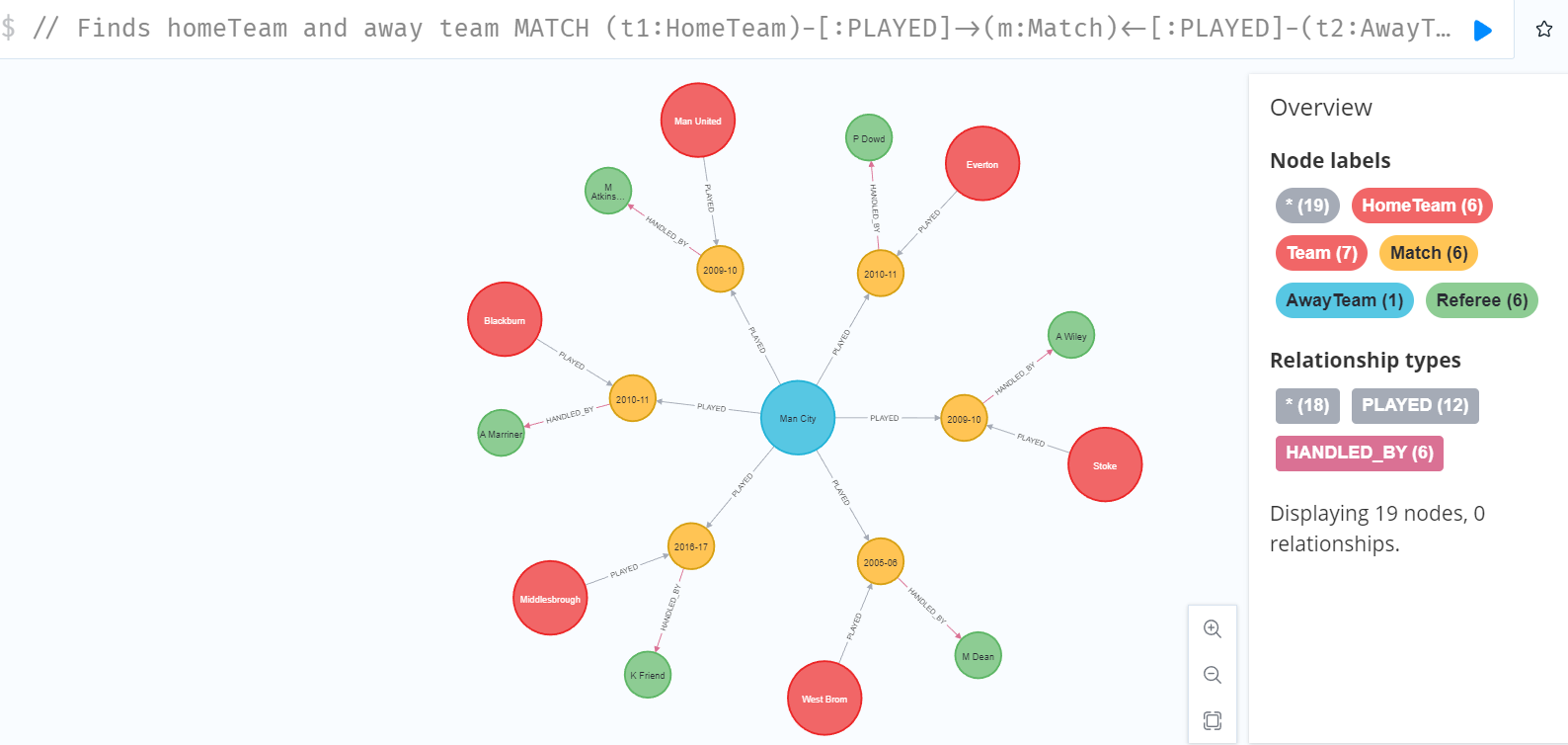
// return the nodes and relationships between them

RETURN t1,m,t2,r

LIMIT 6;

**Result:**

Here, home team are represented in red nodes which played match (i.e. in yellow node), with away team (blue node), The referee of matches are represented as green nodes.



**Queries:**

1. All Teams that have ever played EPL matches since 2000.

Code:

// Find the team that played matches

MATCH (team:Team)-[:PLAYED]->(match:Match)

// filter by season greater or equal to 2000-01

WHERE match.season >= "2000-01"

// return all teams

RETURN DISTINCT toUPPER(team.name) AS Teams

// order by team name in ascending order

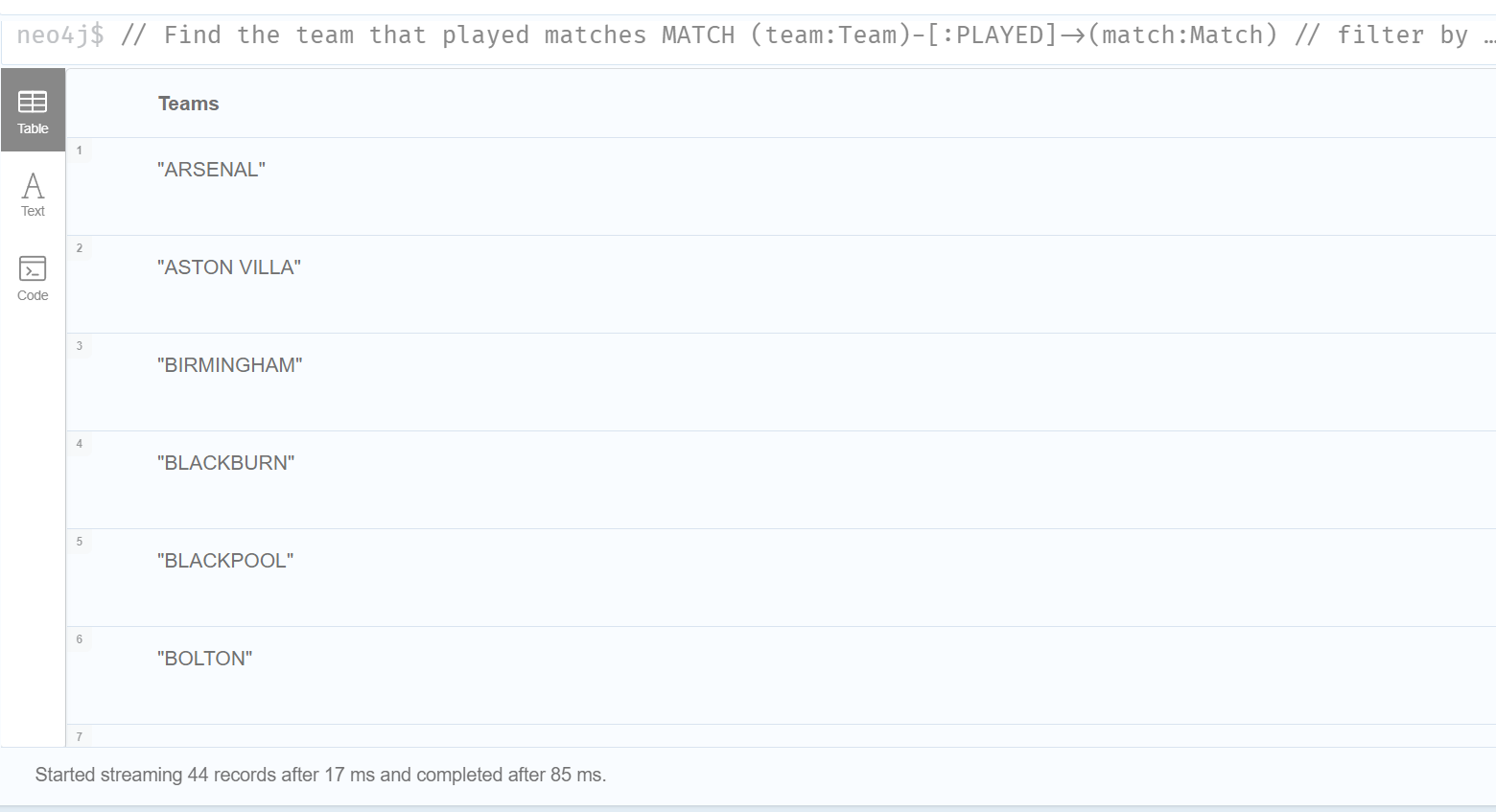
ORDER BY Teams;

Explanation:

This code finds all the team that have played matches since 2000 season and return the teams name in upper case ordering in ascending order.

Result:

Total 44 teams are available.



1. Since 2010, all matches that “Liverpool” won against “Man United”.

Code:

// Find the team liverpool that played matches with man united

MATCH (liverpool:Team {name: 'Liverpool'})-[:PLAYED]->(match:Match)<-[:PLAYED]-(united:Team {name: 'Man United'})

// filter data by sesason 2010

WHERE match.season >= "2010" AND match.fullTimeResult = 'Liverpool'

// return nodes and relationship

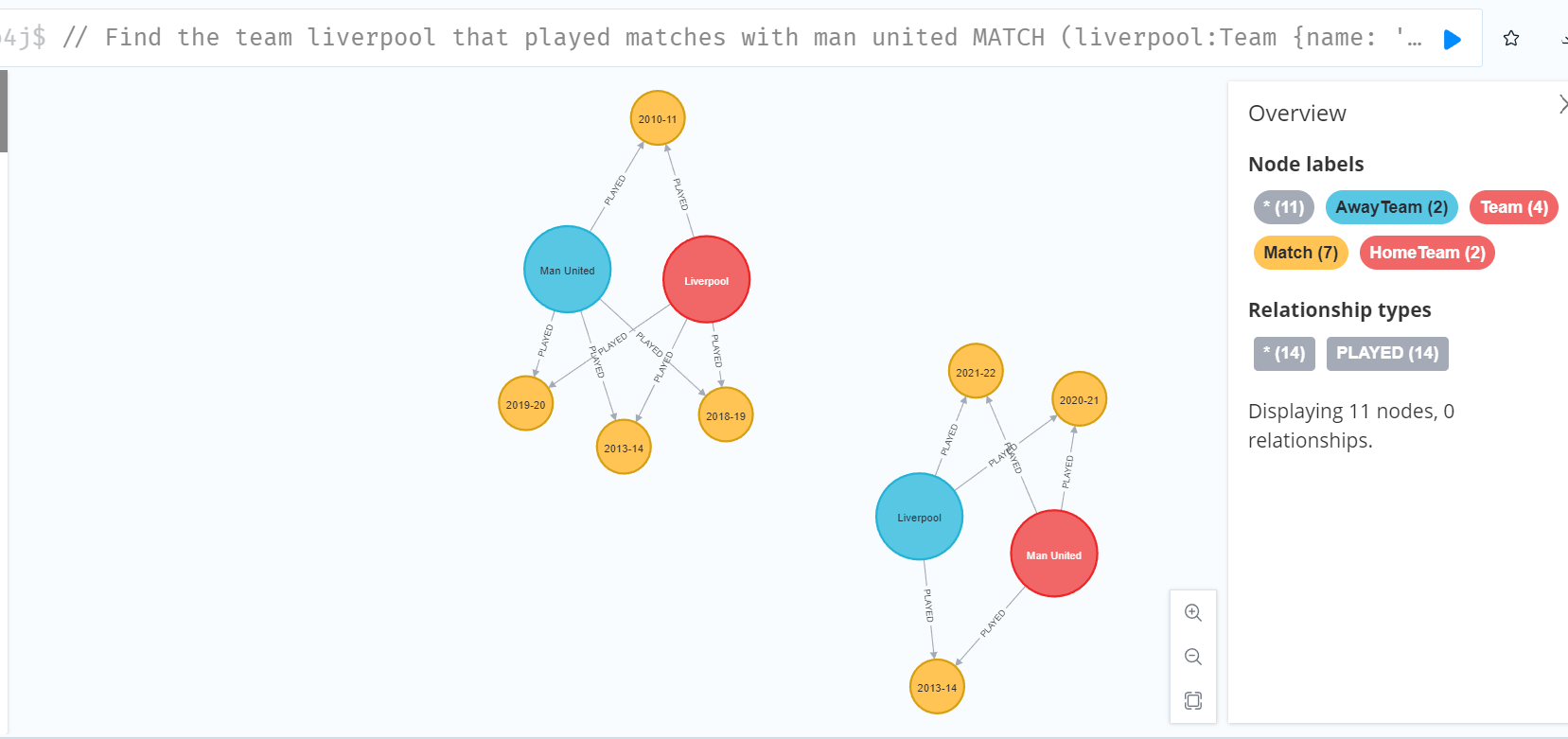
RETURN liverpool,match,united;

Explanation:

This code matches all matches that Liverpool played with man united, the records are filtered using season and all matches which winner are Liverpool are displayed. Winner name are stored in fullTimeResult property of match.

Result:

Total 7 matches found.



1. Top five referees and no of matches they refereed since 2000.

// Find all matches that handled or referred by referee

MATCH (m:Match)-[:HANDLED\_BY]->(referee:Referee)

// find matches since 2000 season

WHERE m.season >= "2000-01"

// counts the total matches

RETURN COUNT(m) AS total\_matches\_refereed, referee.name AS Referee

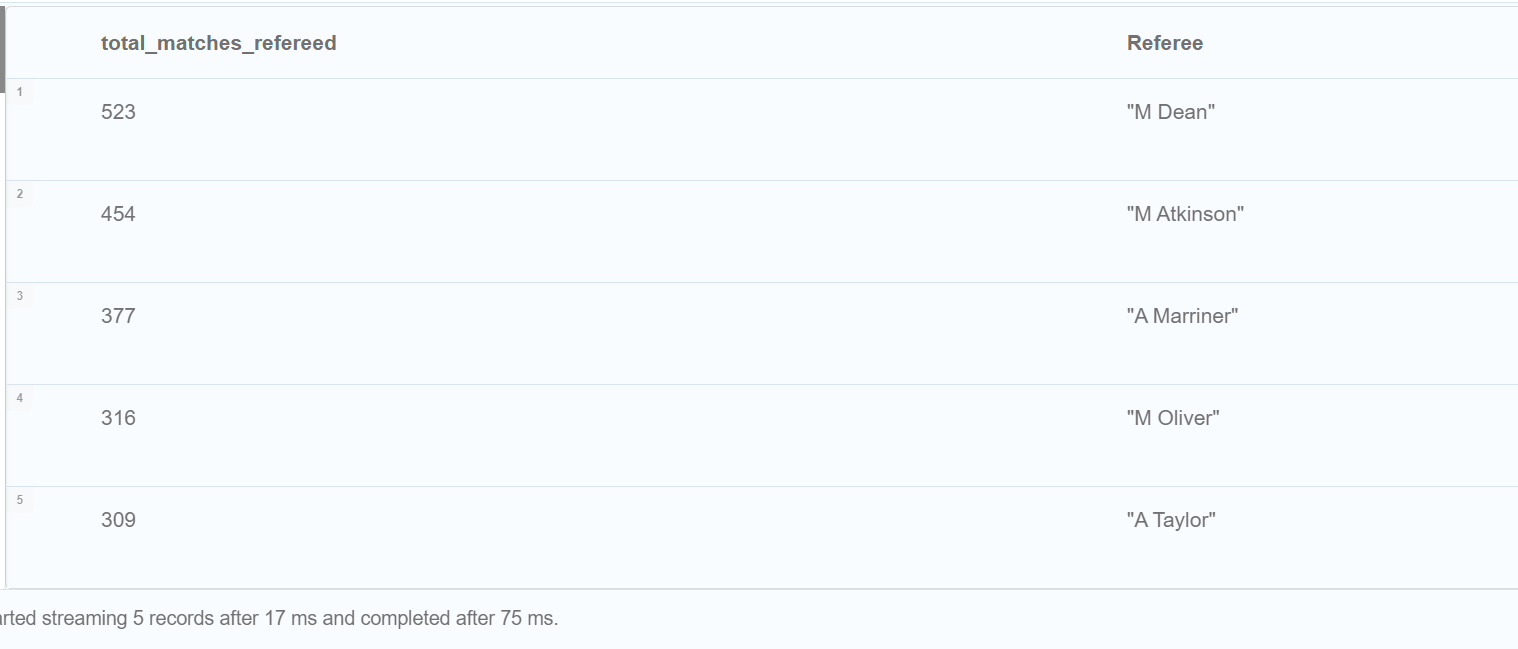
// order by total matches in descending order

ORDER BY total\_matches\_refereed DESC

// shows only five records

LIMIT 5;

Result:



1. All Teams and total no of goal they scored and conceded in the 2020/21 season.

//Find the matches between two teams in 2020/21 season

MATCH (t1:Team)-[homeTeam:PLAYED]->(m:Match)<-[awayTeam:PLAYED]-(t2:Team)

WHERE m.season = "2020-21"

// Return total teams and the goals they have scored and conceded

RETURN t1.name As teams, SUM(homeTeam.fullTimeGoal) AS scored, SUM(awayTeam.fullTimeGoal) As conceded

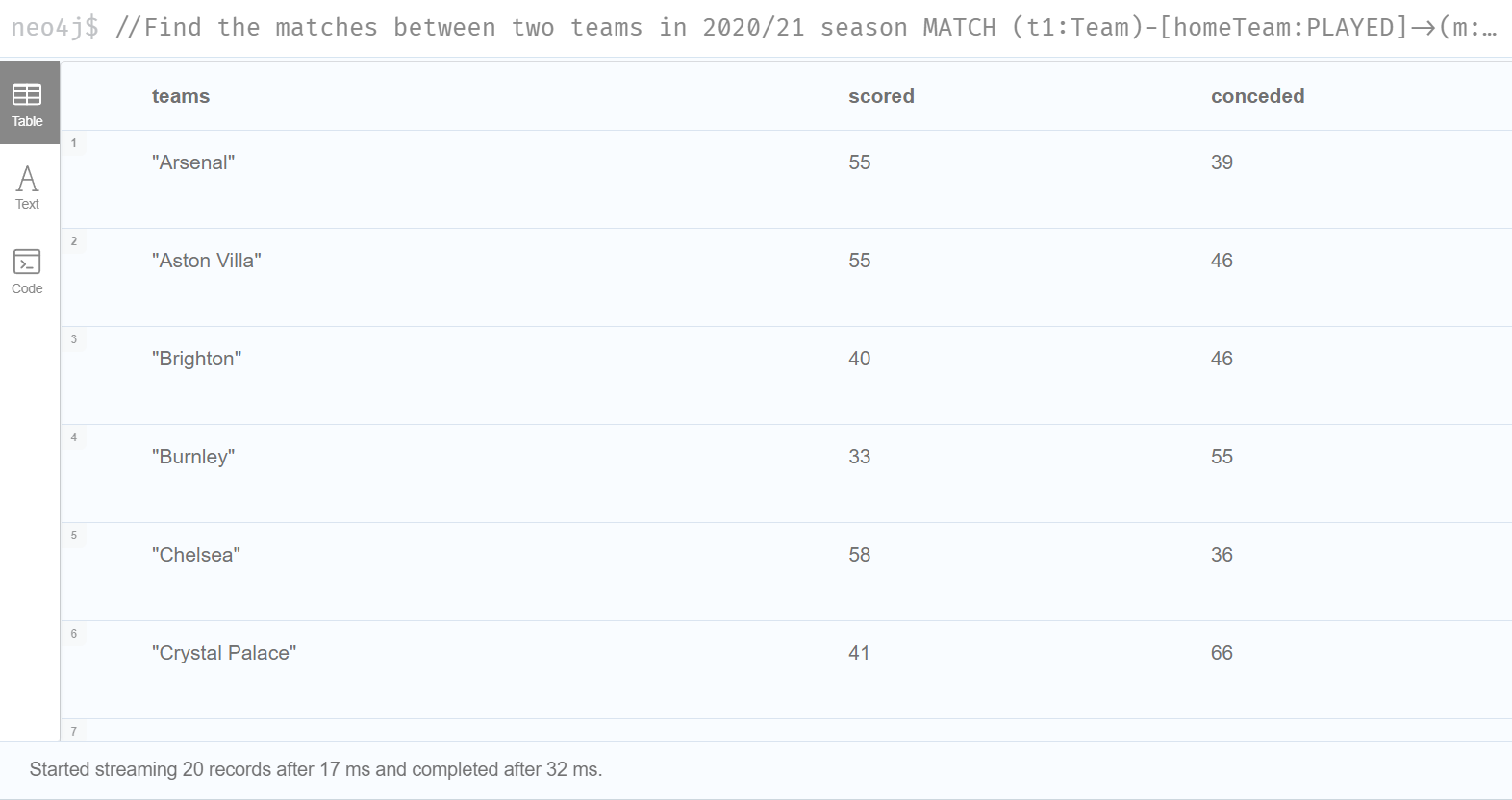
// order by team name

ORDER BY teams;

Explanation:

This code finds two team and match played between them in season 2020/21. The total goal scored by first team is total goal scored and the total goal scored by another is the conceded goal for first team.

Result:



**Testing:**

Let’s calculate the total goal scored and conceded by arsenal in season 2020-21 to check accuracy of above result.

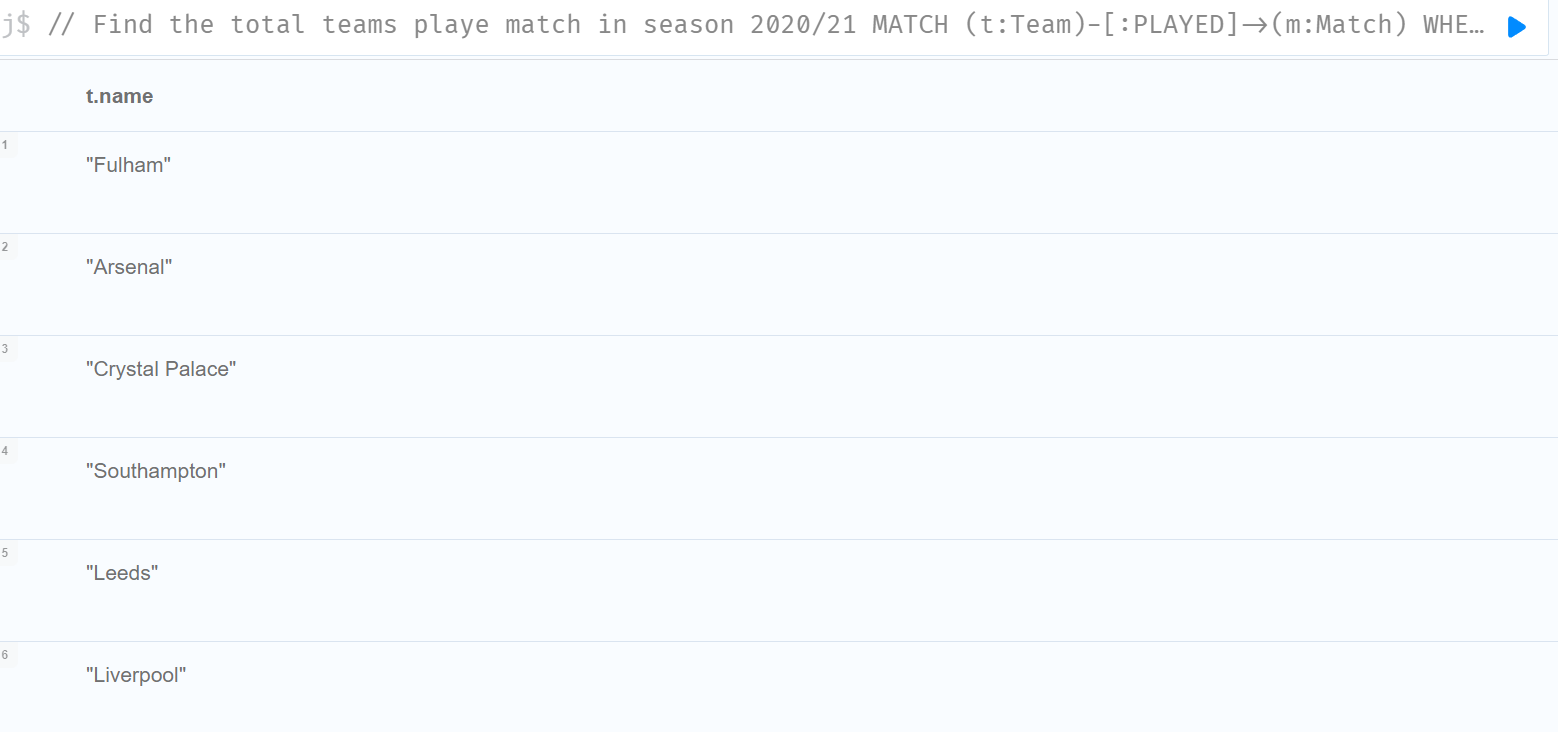
// Find the total teams playe match in season 2020/21

MATCH (t:Team)-[:PLAYED]->(m:Match)

WHERE m.season = "2020-21"

RETURN DISTINCT t.name;

* We found the team Arsenal has played in season 2020-21



Now, calculate the total goals and conceded goals.

Code:

// lets find the teams and match played by them

MATCH (t:Team)-[score1:PLAYED]->(m:Match)<-[score2:PLAYED]-(t1:Team)

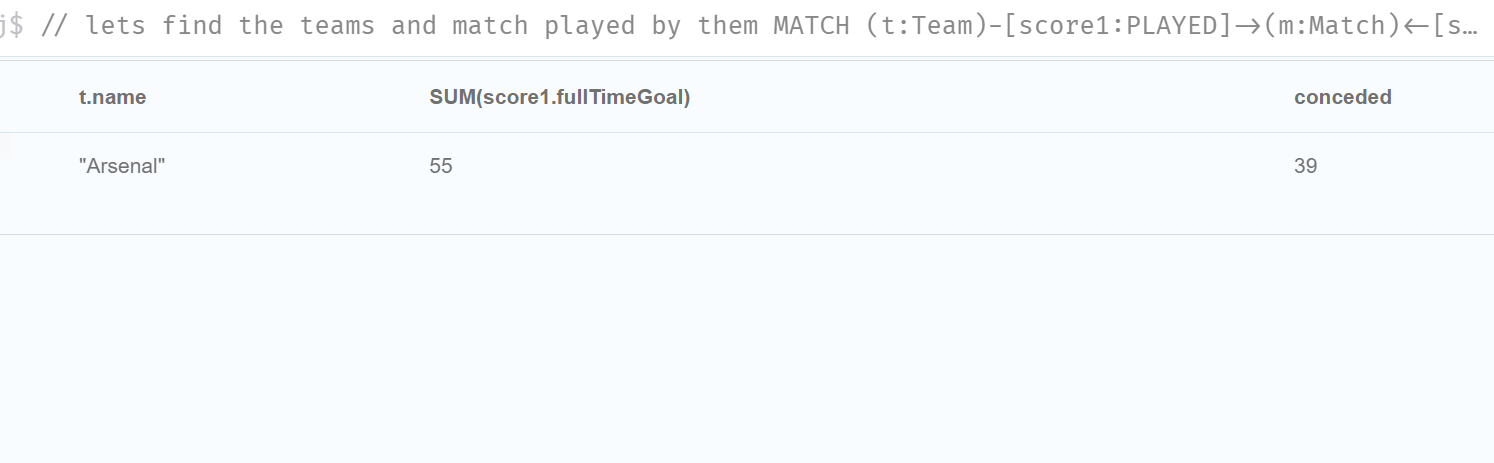
// filter season and take team Arsenal

WHERE m.season = "2020-21" AND t.name = "Arsenal"

// return team name and total score and conceded goal

RETURN DISTINCT t.name, SUM(score1.fullTimeGoal), SUM(score2.fullTimeGoal) As conceded;

**Result: The result is same as above.**



1. Team, who had the best home winning record since 2000.

// Find the home team and the match played by them

MATCH (t:HomeTeam)-[:PLAYED]->(m:Match)

// since 2000 season and winner is home team of that match

WHERE m.season >= "2000-01" AND m.fullTimeResult = t.name

// Return team name and total matches won by team as home team

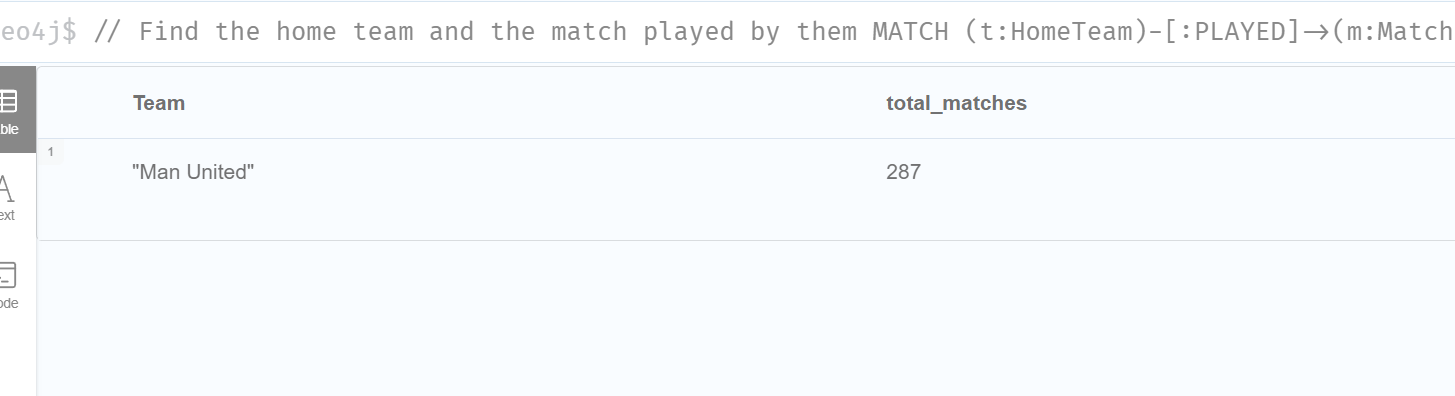
RETURN DISTINCT t.name AS Team, COUNT(t) AS total\_matches

ORDER BY total\_matches DESC

LIMIT 1;

Explanation: This code finds the home team who have won the game since 2000 by MATCH and WHERE clause. After finding records it counts the total matches won by each team and show the top team.

Result:



1. Team that lost most matches in 2020/21 season

Code:

// Find the team and match

MATCH (t:Team)-[:PLAYED]->(m:Match)

// matches where season is 2020/21 and team have losed that match

WHERE m.season = "2020-21" AND (m.fullTimeResult <> t.name AND m.fullTimeResult <> "DRAW")

// Total teams and total lost match

RETURN DISTINCT t.name AS Team, COUNT(m) AS lost\_matches

//ordering

ORDER BY lost\_matches DESC

// Top team

LIMIT 1;

Explanation:

Find the teams of season 2020/21 where that team is not the winner and not draw matches. Return team name and total matches lost, and display the top record.

Result:



1. Teams that lost 1st half but won the match in 2020/21 season

Code:

// Finds the teams and match

MATCH (t1:Team)-[:PLAYED]->(m:Match)<-[:PLAYED]-(t2:Team)

// where halfTimeResult of match is not won by first team

// but at full time won the match

WHERE m.season = "2020-21" AND ((m.halfTimeResult <> t1.name AND m.halfTimeResult <> "DRAW") AND m.fullTimeResult = t1.name)

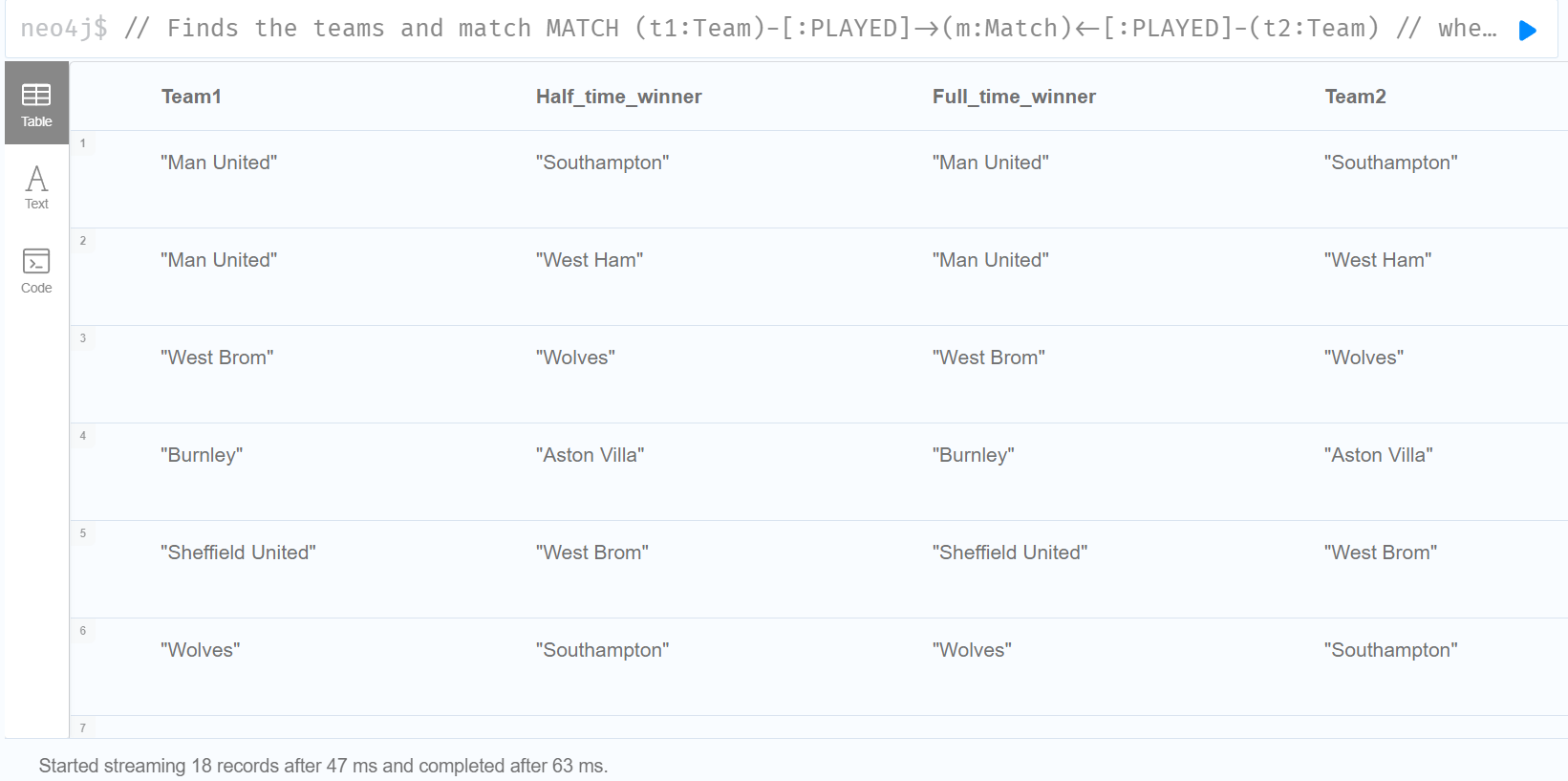
// return team name and half time and full time result

RETURN t1.name As Team1, m.halfTimeResult As Half\_time\_winner, m.fullTimeResult As Full\_time\_winner, t2.name As Team2;

Explanation:

Finds the two teams and match between them in 2020/21 season, WHERE clause filter the record by searching the record where halfTimeResult not having first team name but in fullTimeResult property of match node having the first team’s name.

Result:



1. Team that earned highest ever points in all season since 2000.

// Find eachTeam that played match

MATCH (eachTeam:Team)-[:PLAYED]->(m:Match)

// since 2000 season

WHERE m.season >= "2000-01"

// Return eachTeam and total point obtained

RETURN eachTeam.name As Team\_name, SUM(

CASE m.fullTimeResult

WHEN eachTeam.name THEN 3

WHEN "DRAW" THEN 1

ELSE 0

END) AS point\_obtained

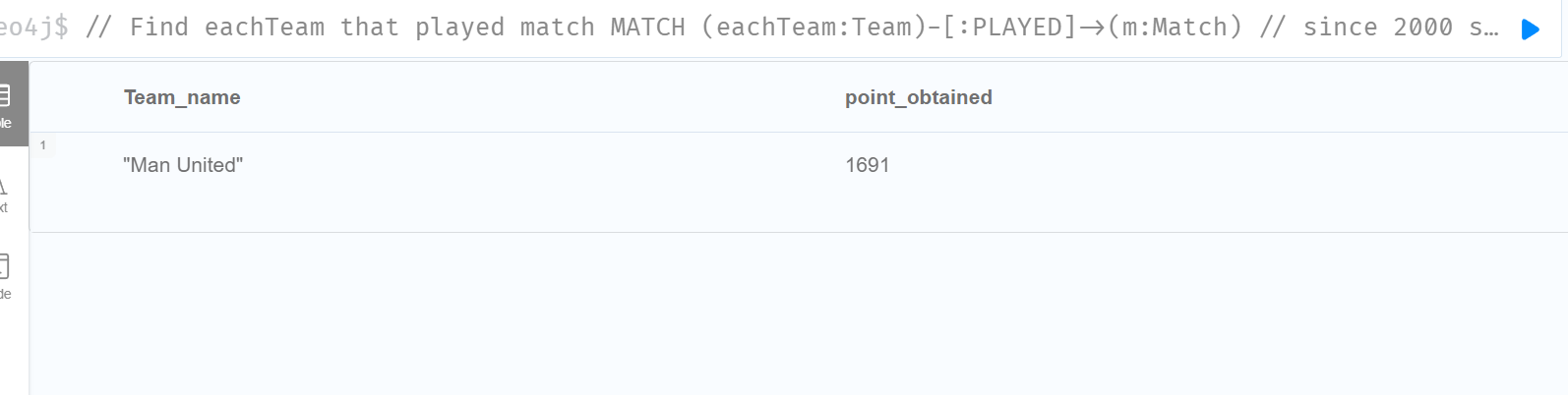
ORDER BY point\_obtained DESC

// show the highest point

LIMIT 1;

Explanation: Finds the teams that played matches since 2000 season and return the team name, Sum the points, if that team won the match then sum by 3 and if match is draw then sum by 1. Display the highest point gainer.

Result:



**Testing:**

Finding total draw matches of Man United, 176 matches obtained. Finding total won matches by Man United, 505 matches obtained. Now calculating, (505\*3) + 176 = 1691. Which is correct.

Code:

// Find draw matches of Man united

MATCH (t:Team)-[:PLAYED]->(n:Match)

// Where draw matches

WHERE n.season>= "2000-01" AND n.fullTimeResult= "DRAW" AND t.name="Man United"

// count the total matches

RETURN COUNT(n);

// 176 matches



// Find won matches

MATCH (t:Team)-[:PLAYED]->(n:Match)

// where wining matches

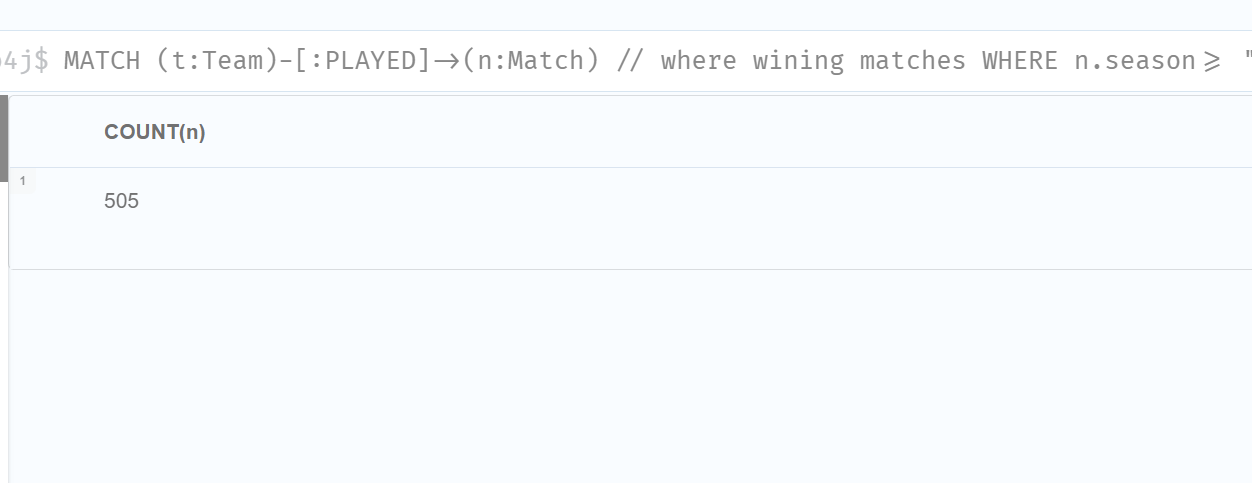
WHERE n.season>= "2000-01" AND n.fullTimeResult= t.name AND t.name="Man United"

// count total matches

RETURN COUNT(n);

// 505 matches

// Result = 505 \* 3 + 176



1. Final league table ranking of all teams in 2020/21 season based on total points.

// Find all teams and matches

MATCH (t:Team)-[:PLAYED]->(m:Match)

// in season 2020-21

WHERE m.season = "2020-21"

//Return team name and sum the points

RETURN DISTINCT t.name AS Ranking, SUM(

CASE m.fullTimeResult

WHEN t.name THEN 3

WHEN "DRAW" THEN 1

ELSE 0

END) AS Points

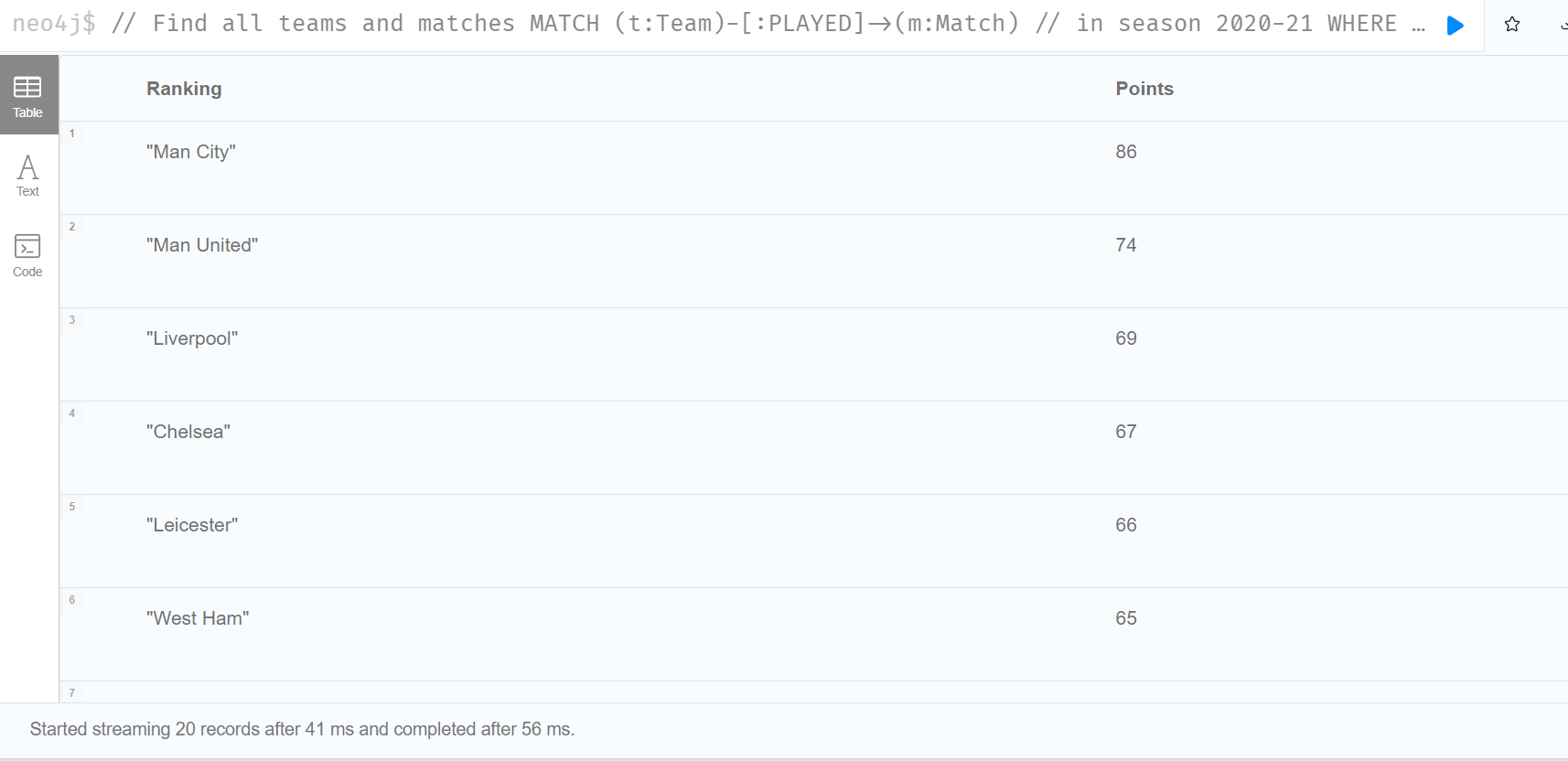
// order in descending order

ORDER BY Points DESC;

Explanation:

This code finds all teams and match played between them in season 2020-21. Returns the team’s name and SUM function with CASE would add the points by 3 if team had won the game. If match is draw, then point is added by 1. Finally displays the record by ordering by points in descending order.

Result:



1. Team with longest unbeaten record.

// Finds the each team and match

MATCH (eachTeam:Team)-[:PLAYED]->(match:Match)

// find the match since 2000 and the match result where team had not losed

WHERE match.season >= "2000-01" AND (match.fullTimeResult = eachTeam.name OR match.fullTimeResult = "DRAW")

// count the unbeaten records

WITH eachTeam.name as Team, COUNT (match) as undefeat\_record

// return name of team

RETURN Team

// show top record

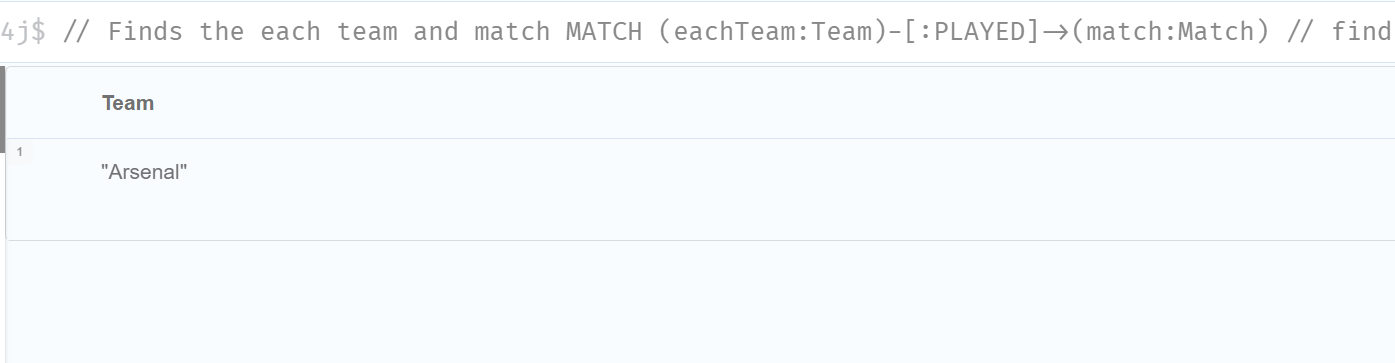
ORDER BY Team

LIMIT 1;

Explanation:

The code finds all matches and record that team had played. Filters the matches with unbeaten record. Passing with WITH clause and counting matches and return the top record having longest unbeaten record.

Result:



**Extra:**

**Finding highest point gainer in each season.**

// To get the highest point gainer in each season

MATCH (eachTeam:Team)-[:PLAYED]->(m:Match)

WHERE m.season >= "2000-01"

// pass unique team name and season, sum the total points

WITH DISTINCT eachTeam.name as Team, m.season As season,SUM(

CASE m.fullTimeResult

WHEN eachTeam.name THEN 3

WHEN "DRAW" THEN 1

ELSE 0

END) AS points

// order by season in ascending and points by descending order

ORDER BY season, points DESC

// after collect the team name and points obtained of each season

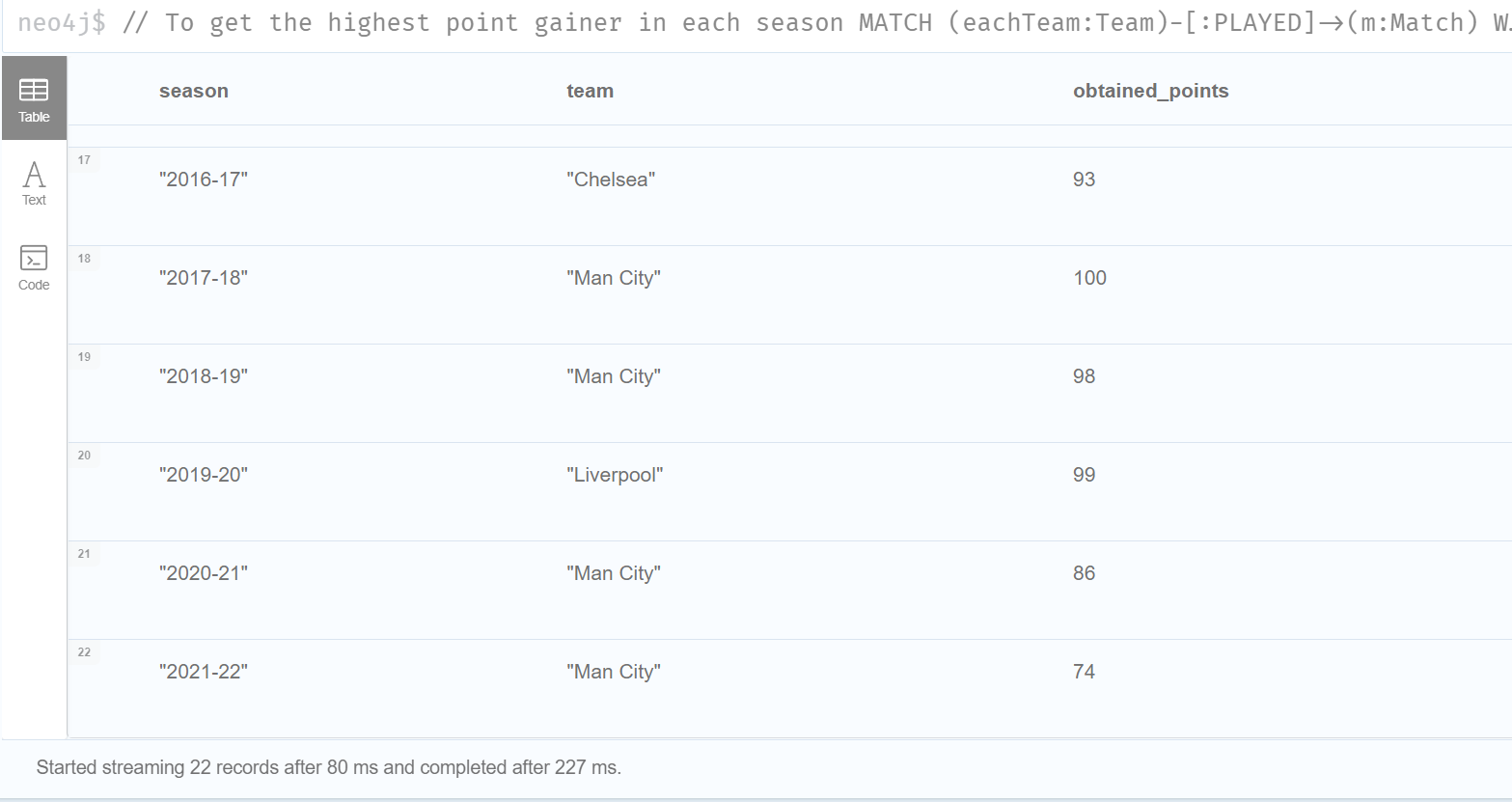
WITH season, collect({team:Team, point:points})[0] AS topHolder

// return the top point gainer of each season

RETURN season, topHolder.team as team, topHolder.point as obtained\_points ;

Explanation: Finds all teams and sum the match points of all seasons. After that list is passed with WITH clause with first record of every season. Return name and total points obtained.

Result: Getting top gainer of each season



For dropping Constraints:

// Dropping constriant

DROP CONSTRAINT hometeam\_is\_unique;

DROP CONSTRAINT awayteam\_is\_unique;

DROP CONSTRAINT referee\_is\_unique;

**References**

Fernandes, D., & Bernardino, J. (2018). Graph databases comparison: Allegrograph, arangoDB, infinitegraph, Neo4J, and orientDB. *DATA 2018 - Proceedings of the 7th International Conference on Data Science, Technology and Applications*, 373–380. https://doi.org/10.5220/0006910203730380

Mark Needham, A. E. H. (2019). *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*.