**ADT Lab 7 (10pts)**

**Neo4j**

**Q1**

**Q2**

**Q3**

**Q4**

**Q5**

**Highlight the questions you are done with green color before you submit**

*What is Neo4j?*

Neo4j is a graph database management system that allows users to store, manage, and query graph data. Unlike traditional relational databases, which are organized around tables and columns, graph databases store data in nodes and edges, which represent entities and relationships between entities, respectively.

In Neo4j, nodes represent entities such as people, places, or things, and edges represent the relationships between them. For example, in a social network graph, nodes might represent individual users, and edges might represent friendships between users. The relationships between nodes can be queried and analyzed to extract insights and patterns from the data.

Neo4j supports the Cypher query language, which is a declarative language used to query graph data. Cypher allows users to write complex queries that traverse the graph, filter data, and perform calculations. It also supports features such as indexing, clustering, and data replication for high availability and scalability.

Neo4j is used in a wide range of applications, including social networks, recommendation engines, fraud detection, and more. It is popular among developers and data scientists due to its flexibility and ease of use, and its ability to handle large-scale connected data.

*What is Cypher Query language?*

Cypher is a query language used to interact with graph databases like Neo4j. It is a declarative language, which means that you describe what you want to retrieve from the database, rather than how to retrieve it.

List of all clauses in Cypher Query Language- <https://neo4j.com/docs/cypher-manual/current/clauses/>

Some of important Clauses-

Here are some important keywords and their uses in the Cypher query language:

MATCH: Used to define patterns to match in the graph data. The pattern is defined using nodes and edges, and can be filtered using properties and labels.

WHERE: Used to filter the results of a MATCH clause based on property values or other conditions.

RETURN: Used to specify what data to retrieve from the graph. You can specify node and edge properties, labels, and other calculations or aggregations.

CREATE: Used to create nodes and edges in the graph.

MERGE: Used to either create a node and edge or match an existing node and edge based on a set of properties.

SET: Used to update the properties of nodes and edges.

DELETE: Used to delete nodes and edges from the graph.

WITH: Used to chain multiple clauses together, passing data between them.

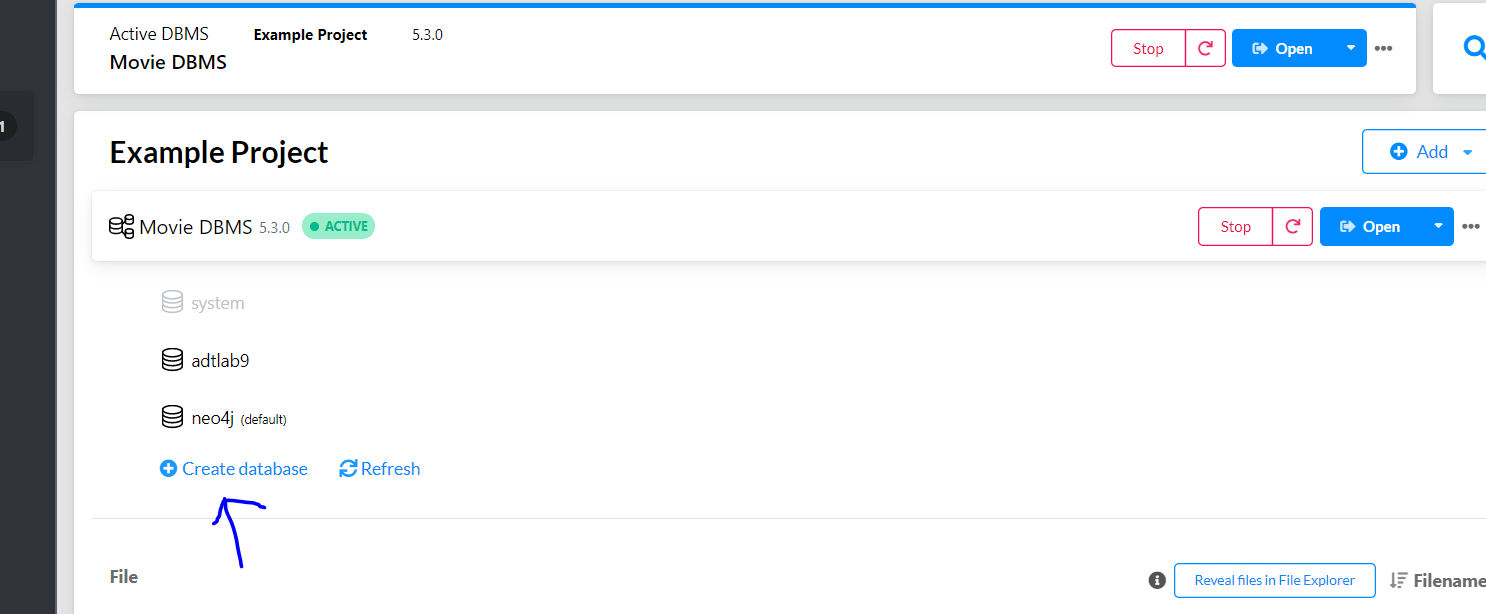
UNION: Used to combine multiple results into a single result set.

ORDER BY: Used to sort the results of a query based on one or more properties.

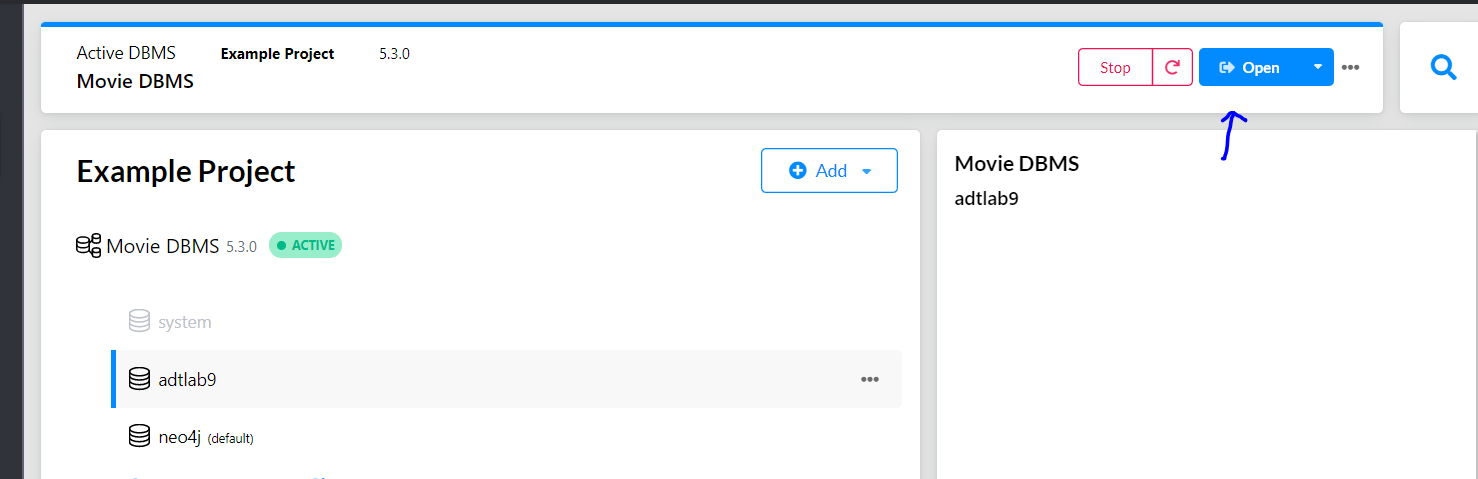
LIMIT: Used to limit the number of results returned by a query.

**Setup Instructions:**

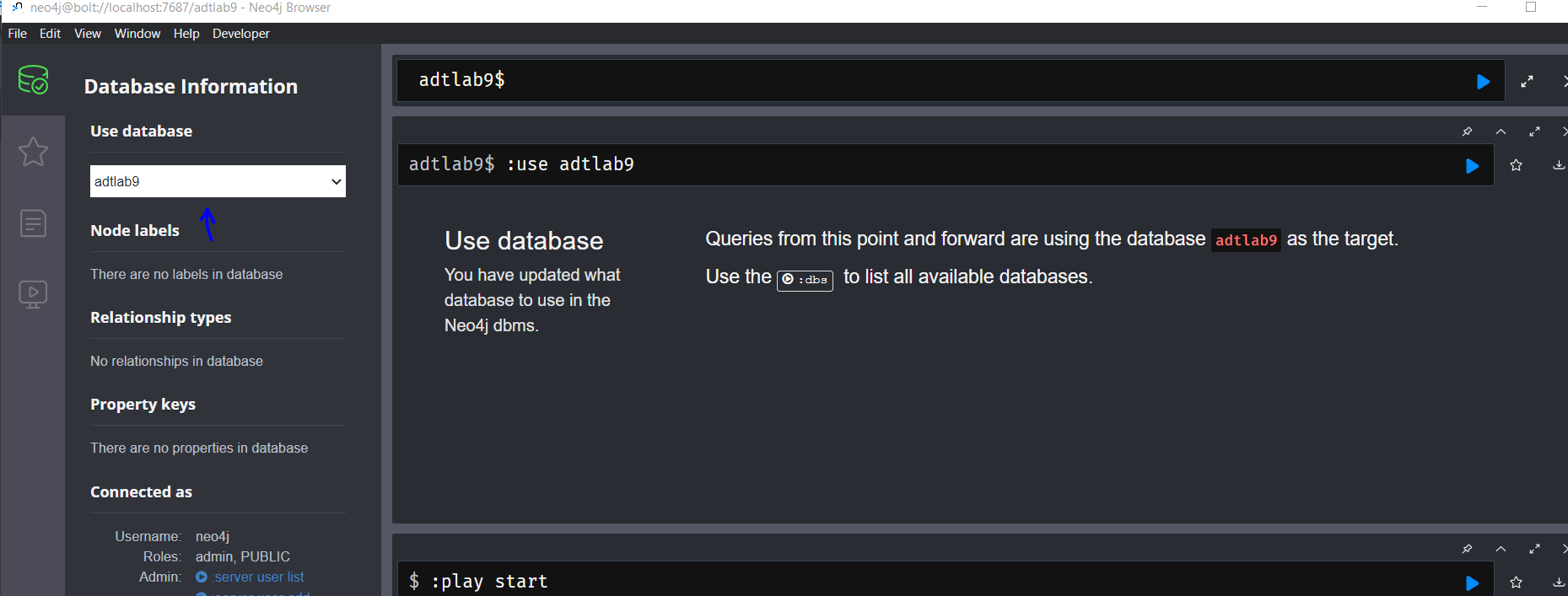
1. Install Neo4j
2. Open Neo4j and click on create database



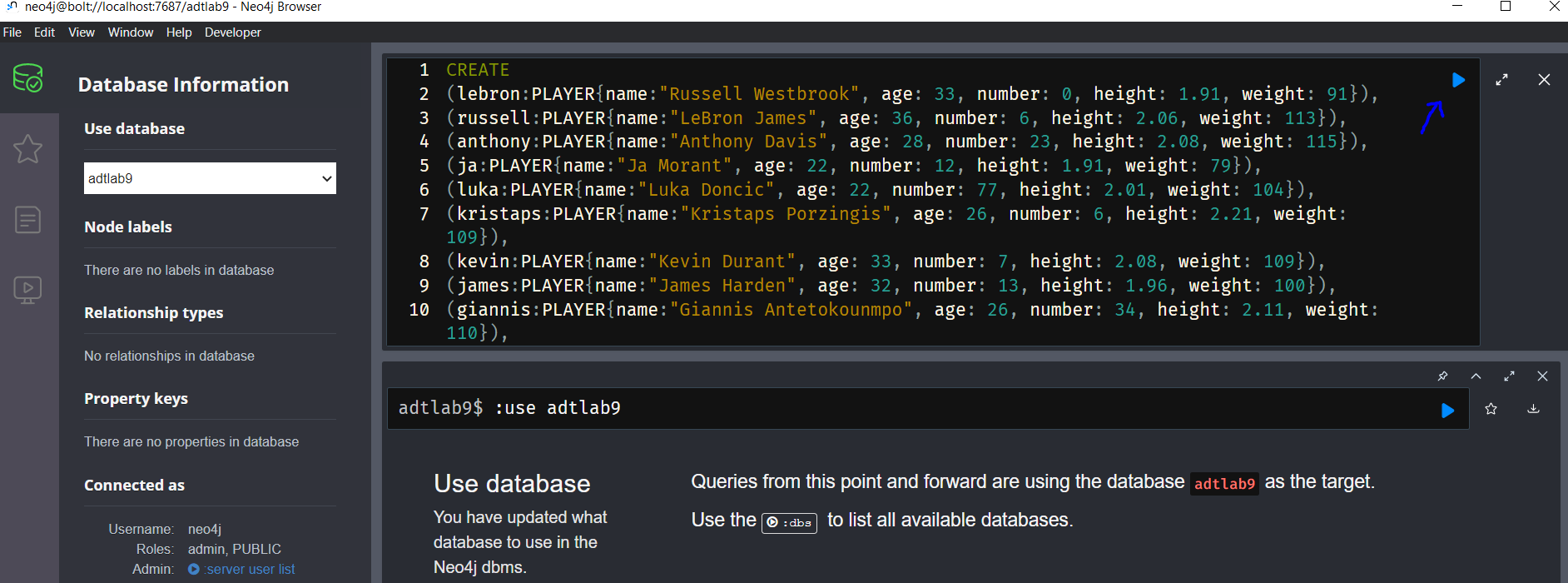
1. Create a new database, you can select any name for it



1. Then click on Open, it will open the Neo4j broswer. As shown below. Make sure you have selected the newly created database.



1. Now copy the contents of basketball.txt and paste in the cell. Then run it



1. To see if the data has been loaded correctly run the below query

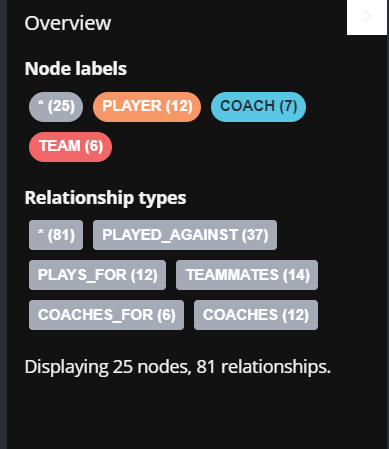
Example Queries-

MATCH (n) RETURN n

Expected output-



If you see to the right, it shows all the newly created nodes and their relationship types.

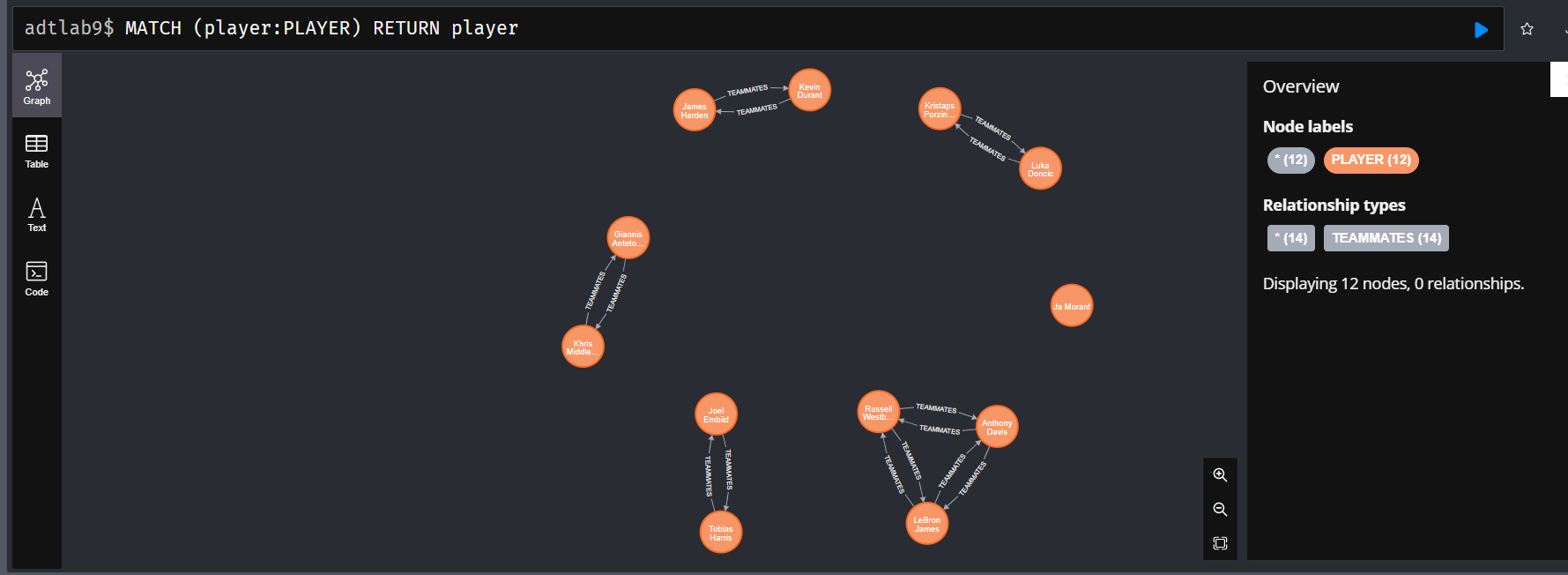


Here we have 12 node of players, 7 node of coach and 6 node of team. In total we have 25 nodes and 81 relationships

Query-

MATCH (player:PLAYER) RETURN player

Expected output-



Query-

MATCH (player:PLAYER) RETURN player.name, player.height

Expected output-



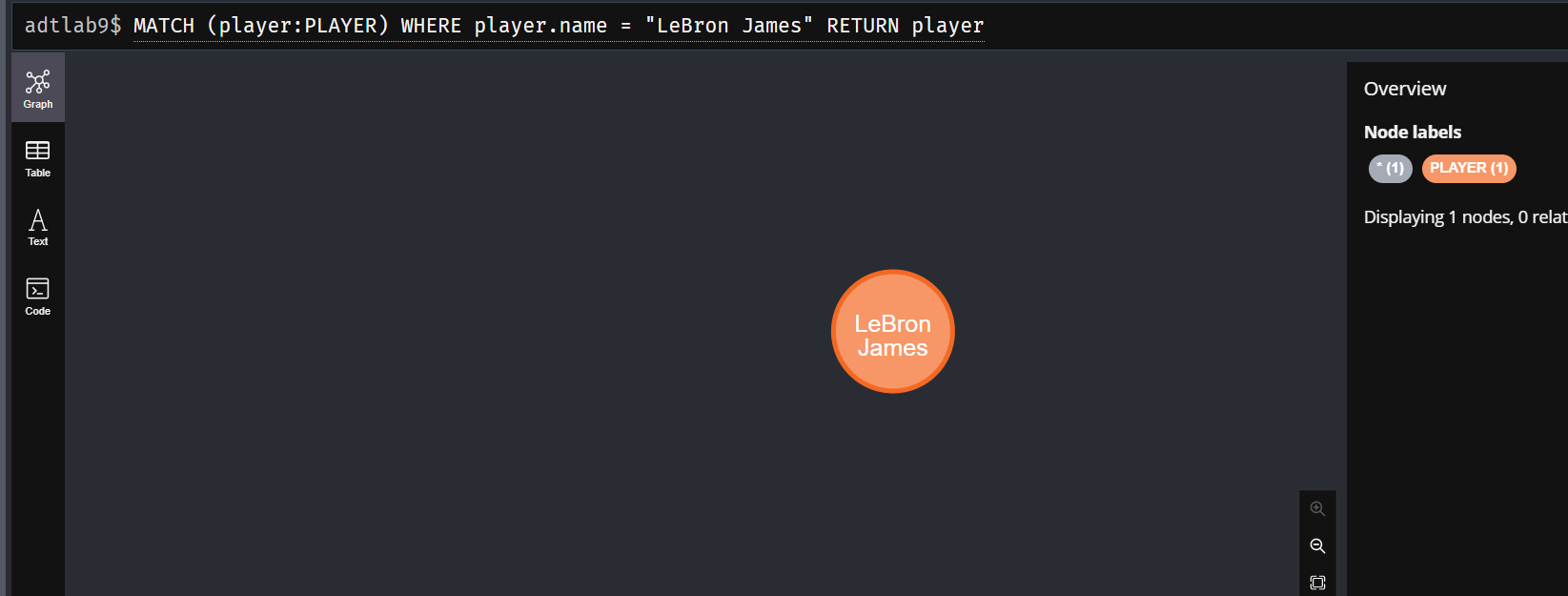
Query-

MATCH (player:PLAYER)

WHERE player.name = "LeBron James"

RETURN player

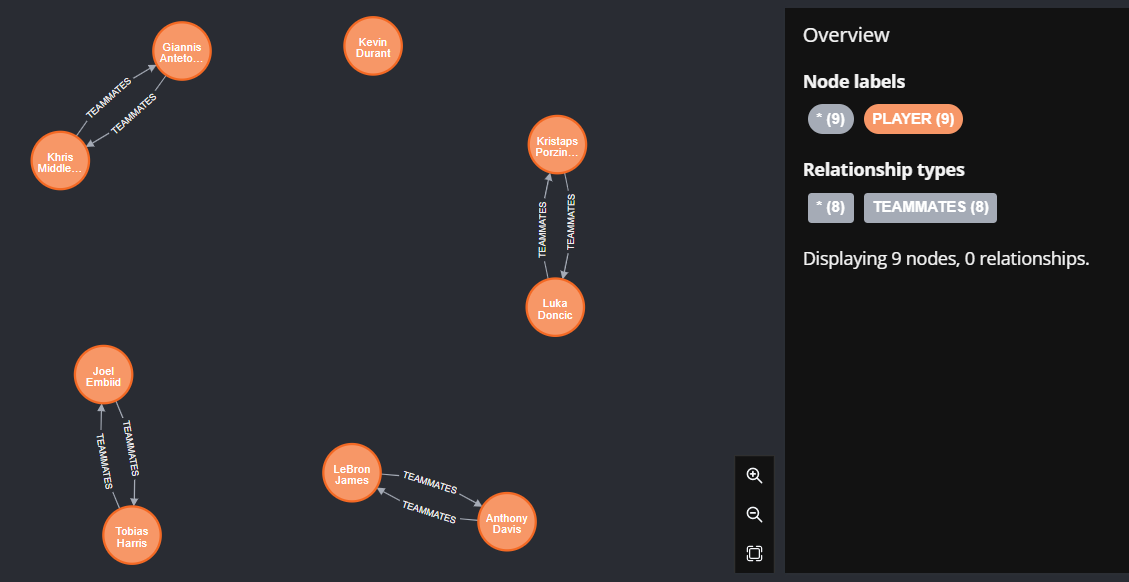
Expected output-



**Question 1**

Player where height is greater than or equal to 2

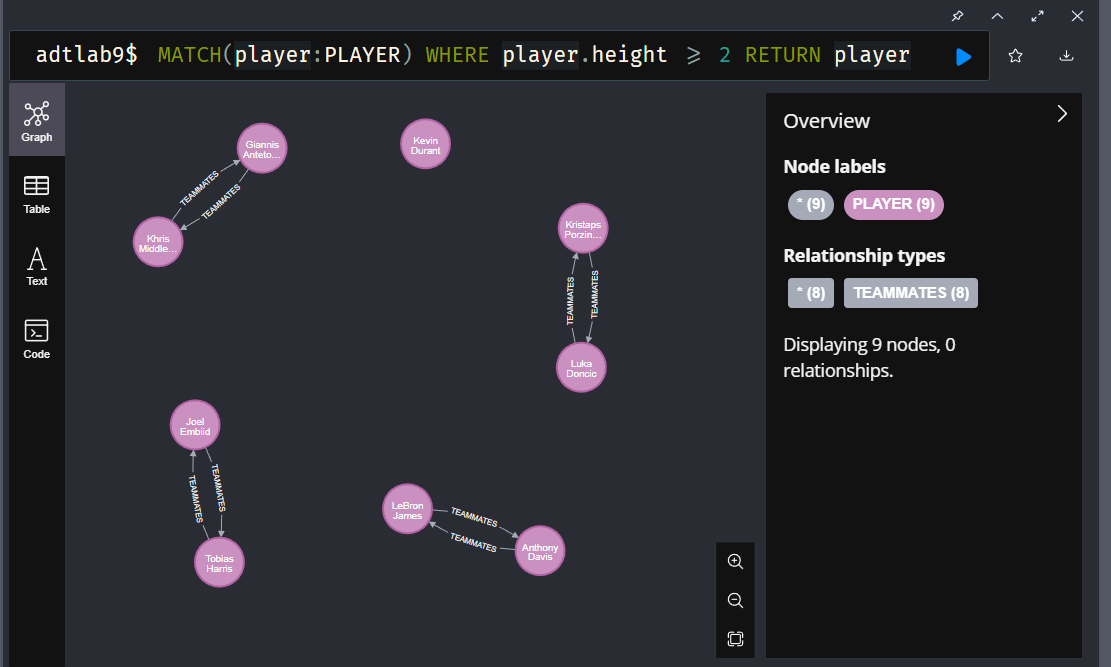
Expected output-



# Your query

MATCH(player:PLAYER) WHERE player.height >= 2 RETURN player

# Your Output

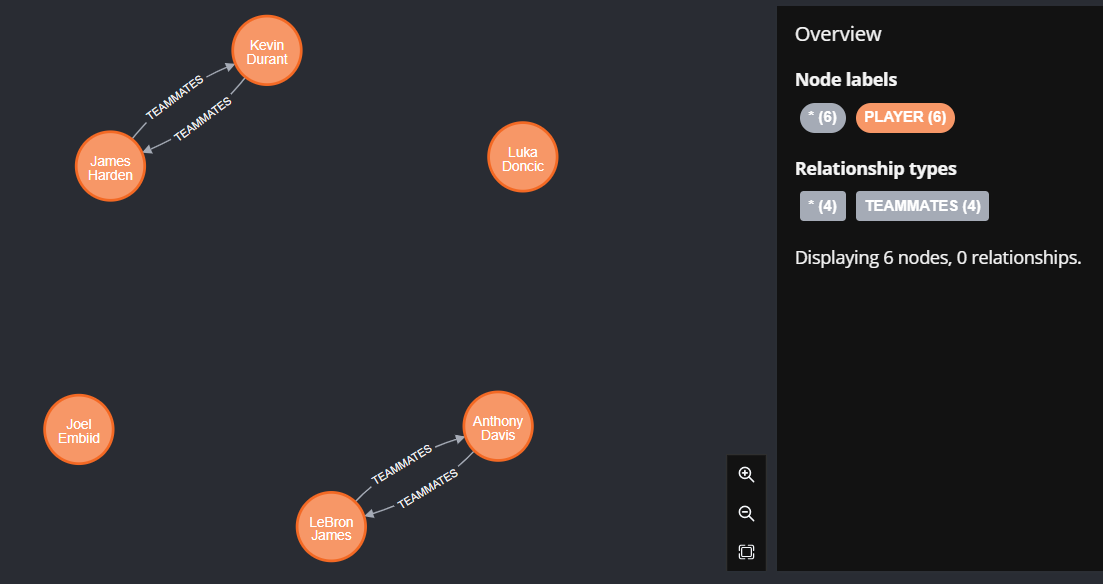


**Question 2**

Players with a BMI larger than 25

BMI = Weight/(Height)^2

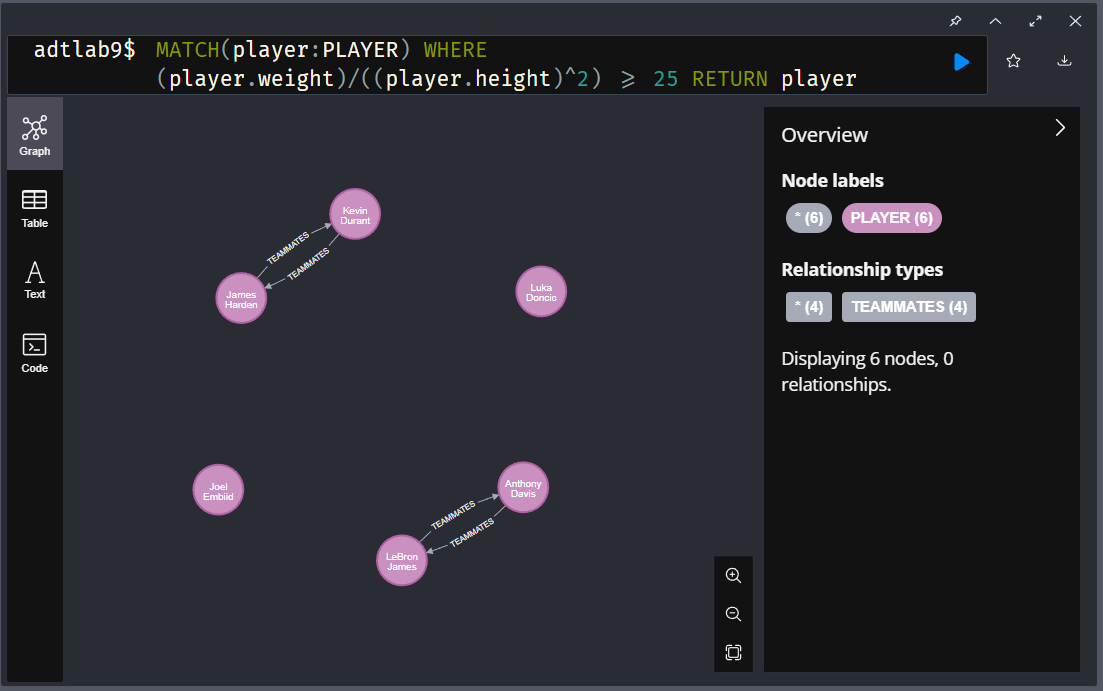
Expected output-



# Your query

MATCH(player:PLAYER) WHERE (player.weight)/((player.height)^2) >= 25 RETURN player

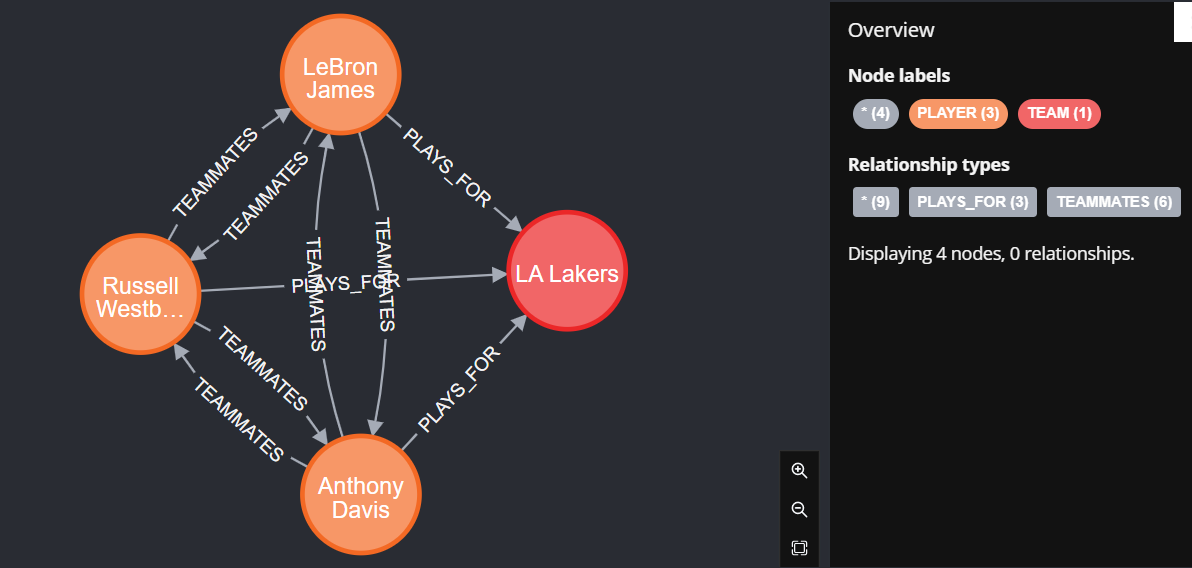
# Your Output



**Question 3**

Select all Players and Team where Team Name is ‘LA Lakers’

Expected output-



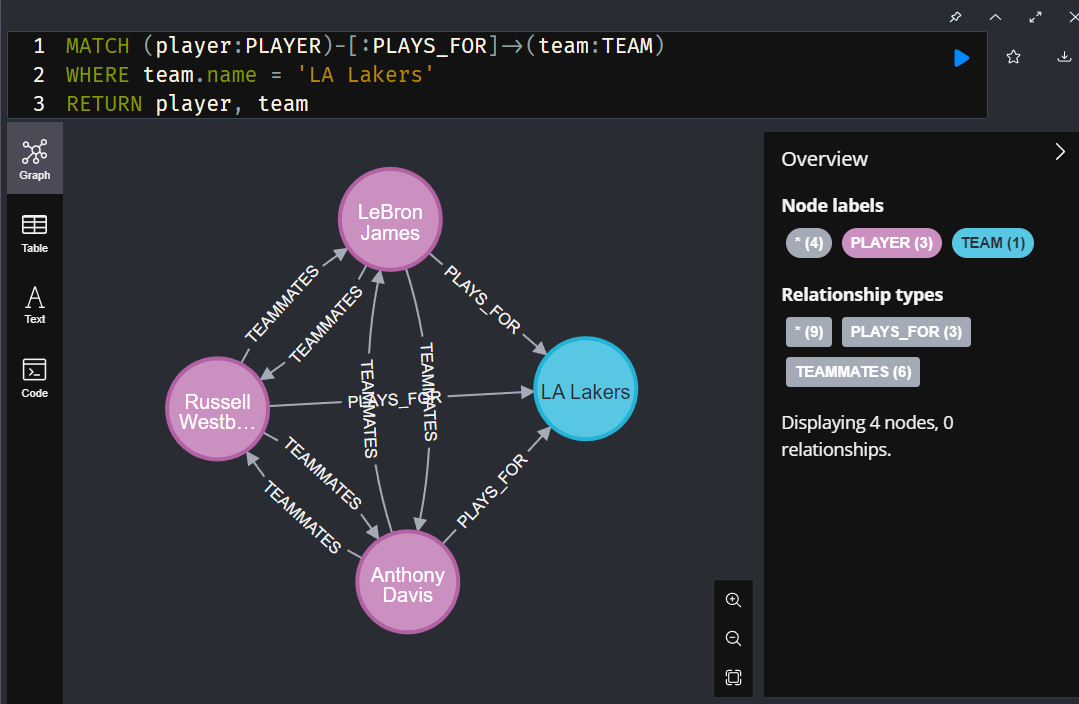
# Your query

MATCH (player:PLAYER)-[:PLAYS\_FOR]->(team:TEAM)

WHERE team.name = 'LA Lakers'

RETURN player, team

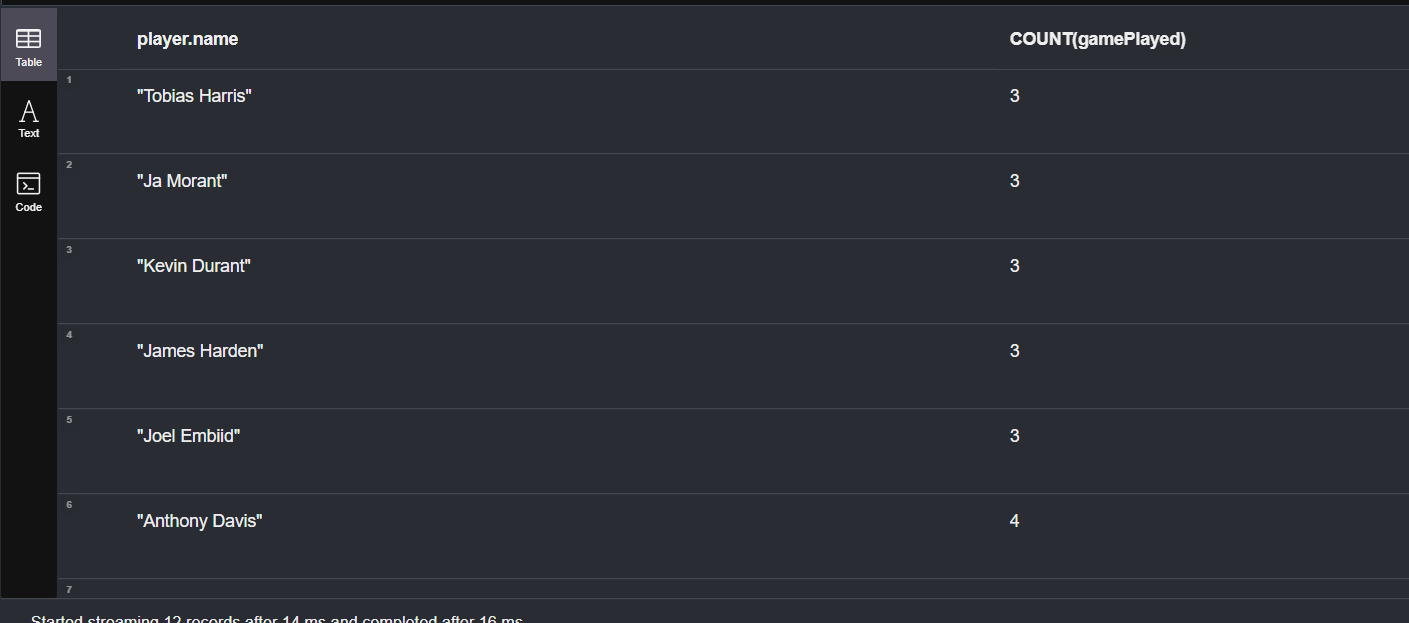
# Your Output



**Question 4**

Get players and number of games played

Expected output-



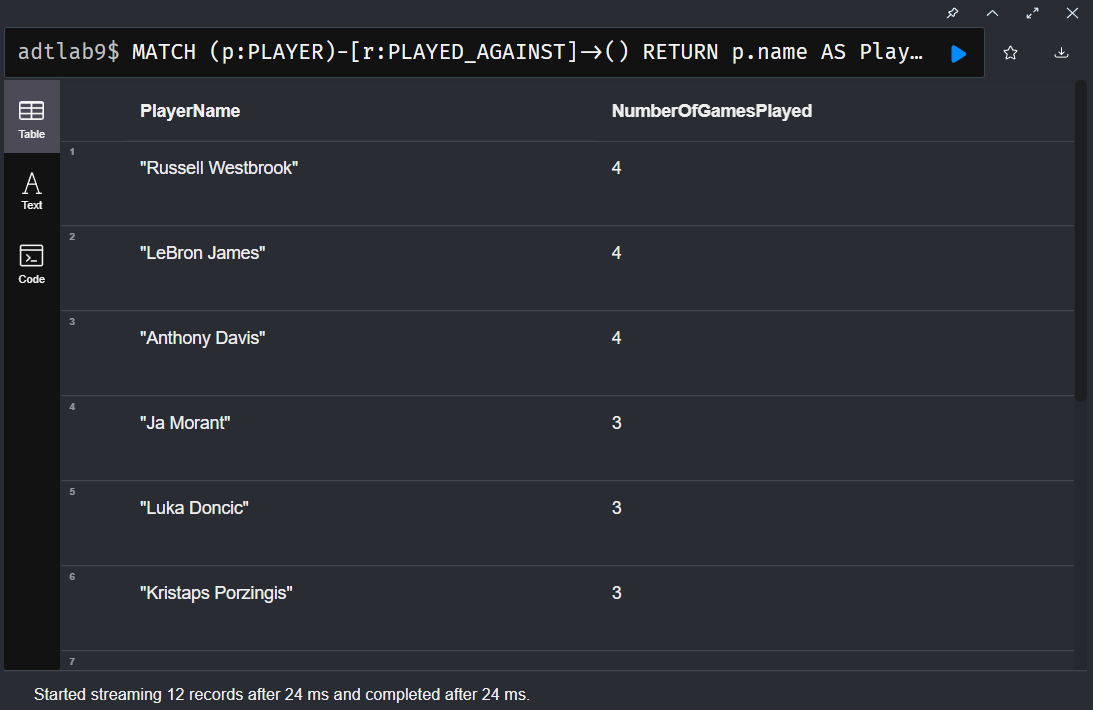
# Your query

MATCH (p:PLAYER)-[r:PLAYED\_AGAINST]->()

RETURN p.name AS PlayerName, COUNT(r) AS NumberOfGamesPlayed

ORDER BY NumberOfGamesPlayed DESC;

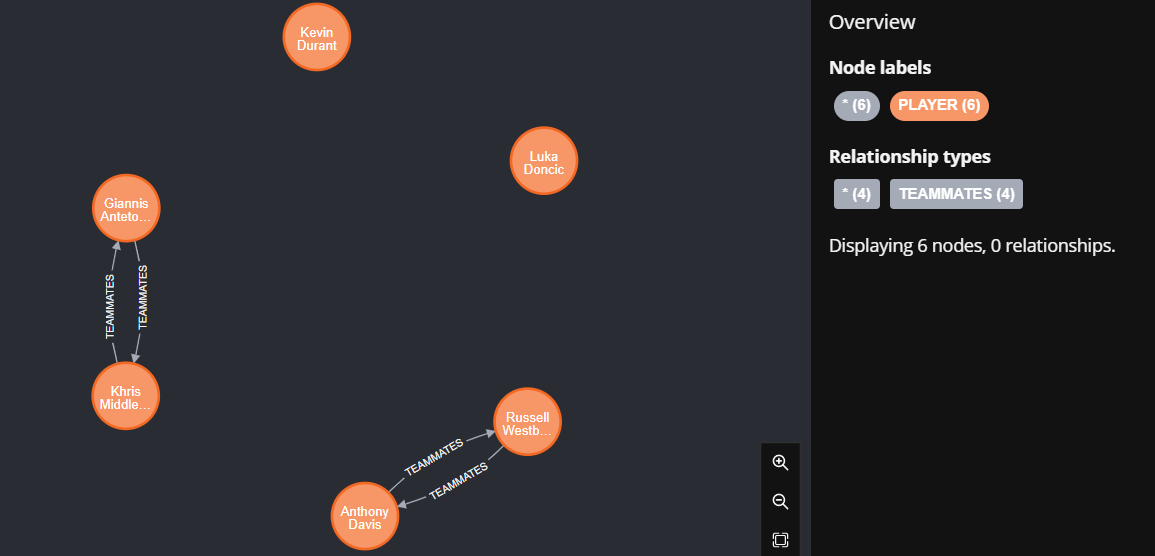
# Your Output



**Question 5**

Get all players that make more than 35M

Expected output-



# Your query

MATCH(p:PLAYER)-[pf:PLAYS\_FOR]->(x) WHERE pf.salary > 35000000 RETURN p

# Your Output

