Mithibai College

Department of Computer Science

M.Sc. (Data Sci and AI)

Practical 09: CRUD operations using Neo4j

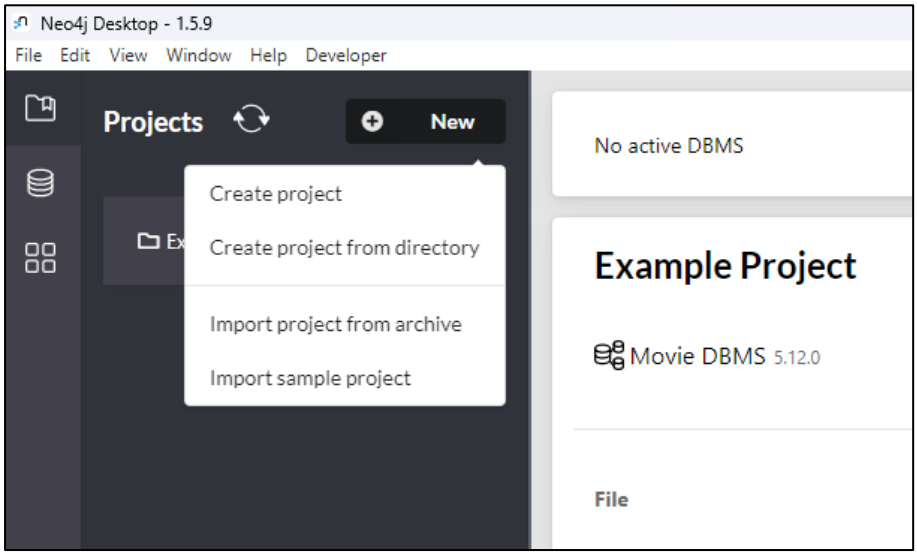
Date: 18/03/2025 **Submission Date: 21/03/2025**

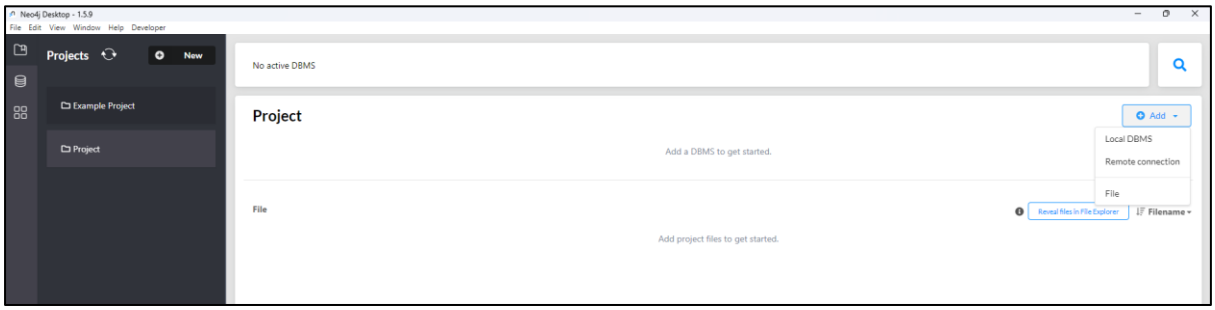
Write-up: -

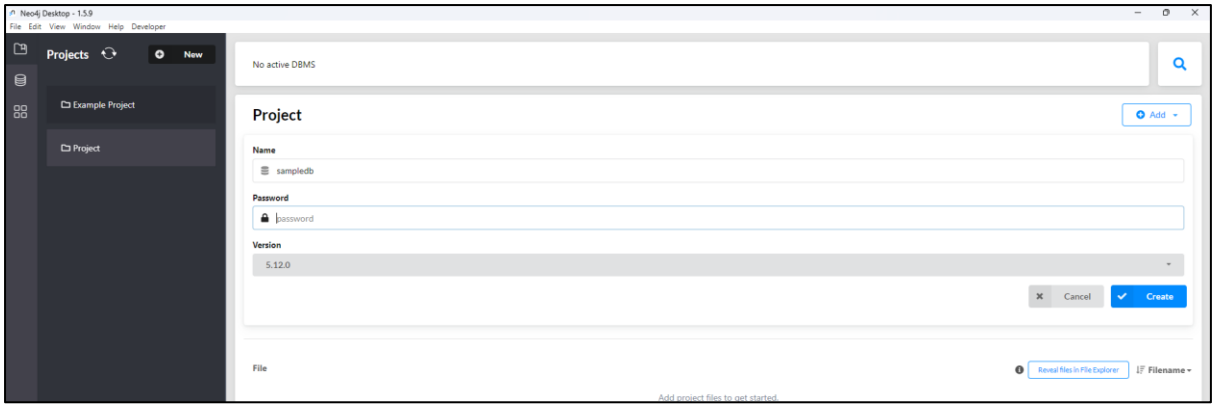
- Graph databases
- Graph databases vs Relational DBMS
- Neo4j
- Cypher query language
- Any 5 CQL constructs

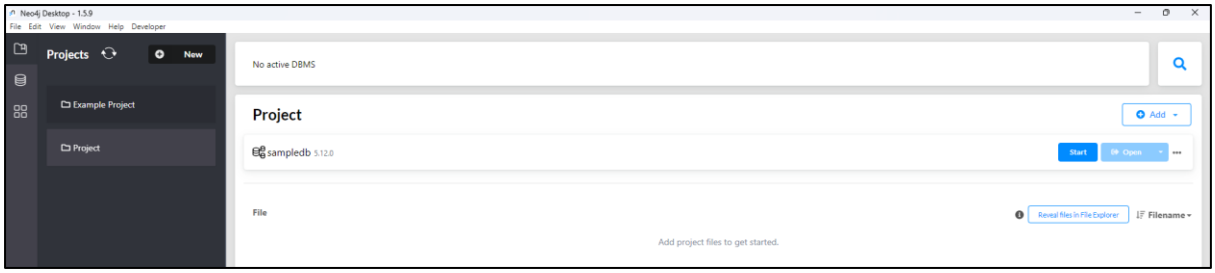
Implement CRUD operations:

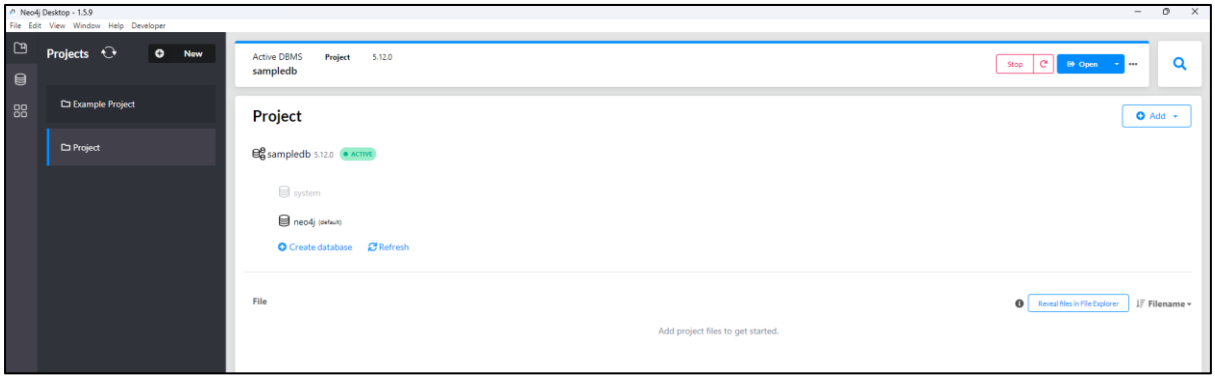
Open the Neo4j desktop App and start the Neo4j Server as shown in the following screenshot

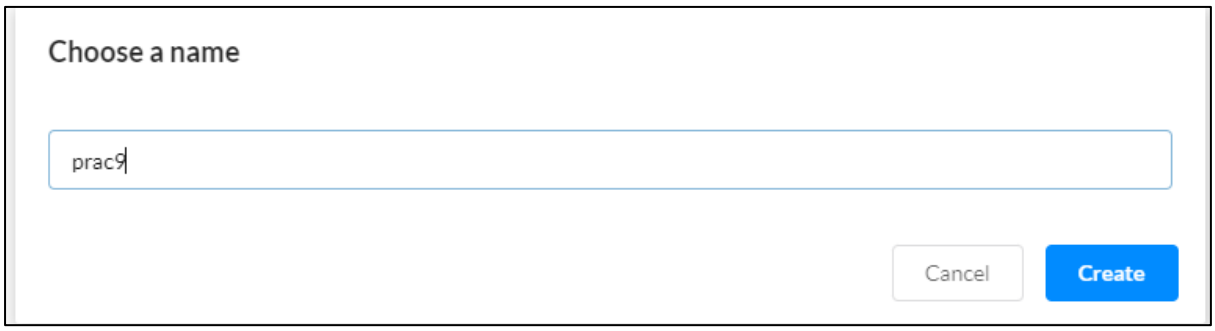


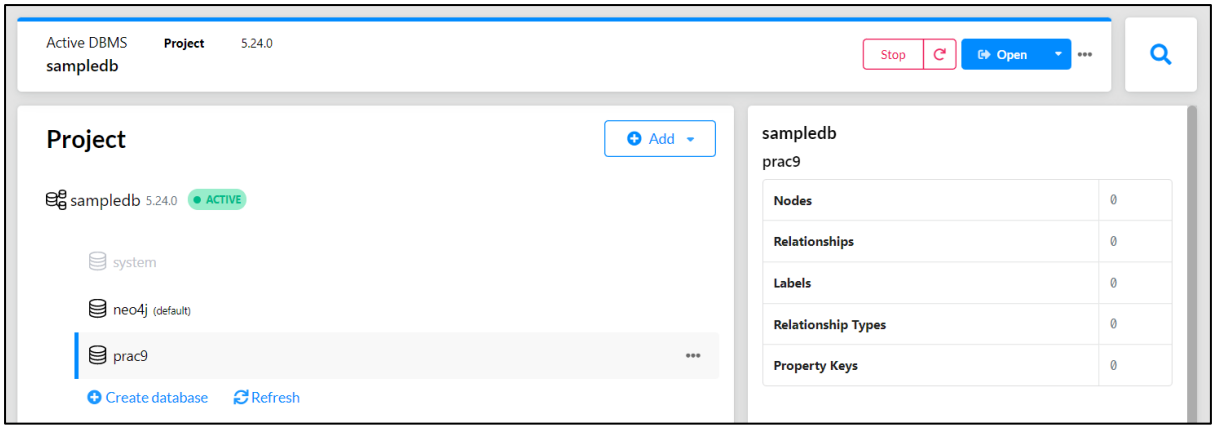


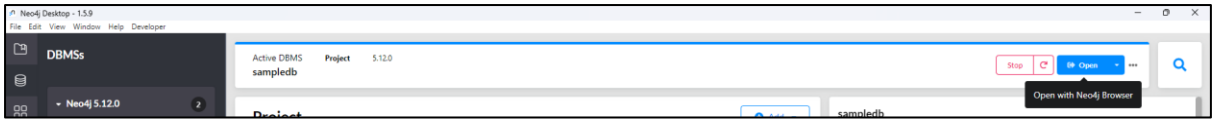

Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
Projects
New
Example Project
Project
No active DBMS
Project
Add
Name
sampledb
Password
password
Version
5.12.0
Cancel
Create
File
Reveal files in File Explorer
Filename
Add project files to get started.


Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
Projects
New
Example Project
Project
No active DBMS
Project
Add
sampledb 5.12.0
Start
Open
File
Reveal files in File Explorer
Filename
Add project files to get started.

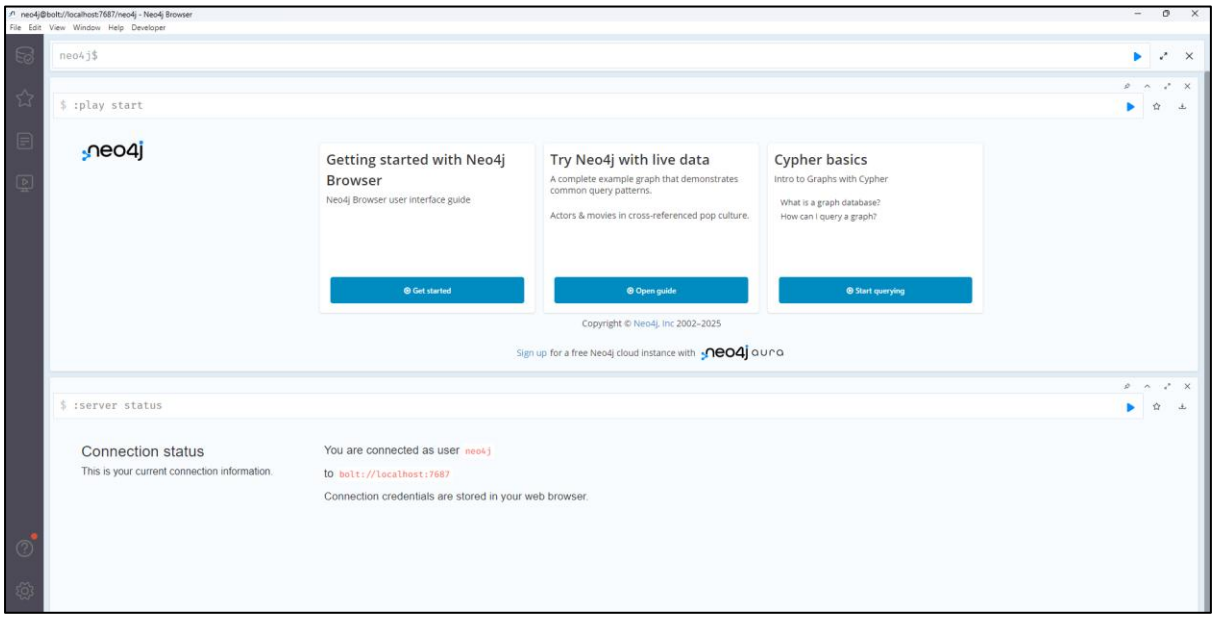

Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
Projects
New
Example Project
Project
Active DBMS
sampledb
Project
5.12.0
Stop
Open
Project
Add
sampledb 5.12.0
ACTIVE
system
neo4j (default)
Create database
Refresh
File
Reveal files in File Explorer
Filename
Add project files to get started.


Choose a name
prac9
Cancel
Create

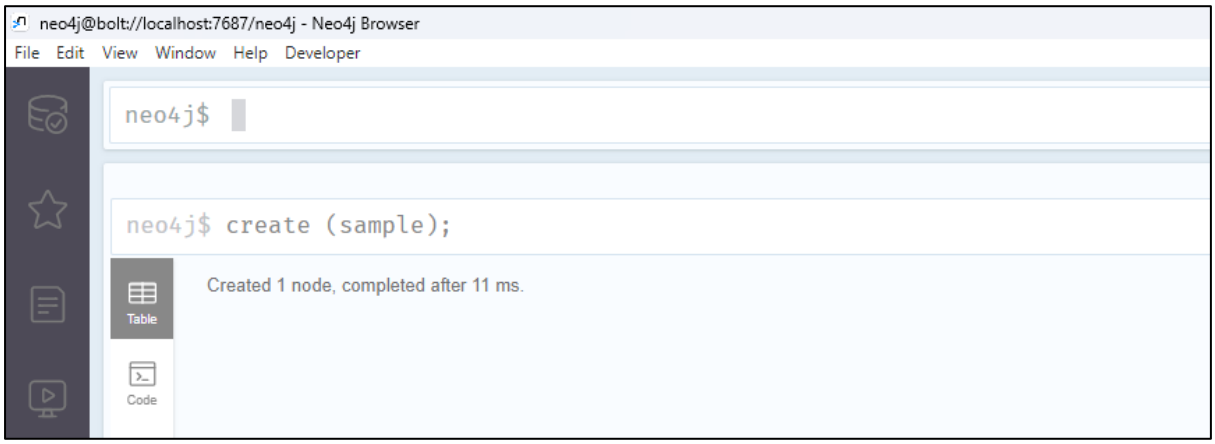




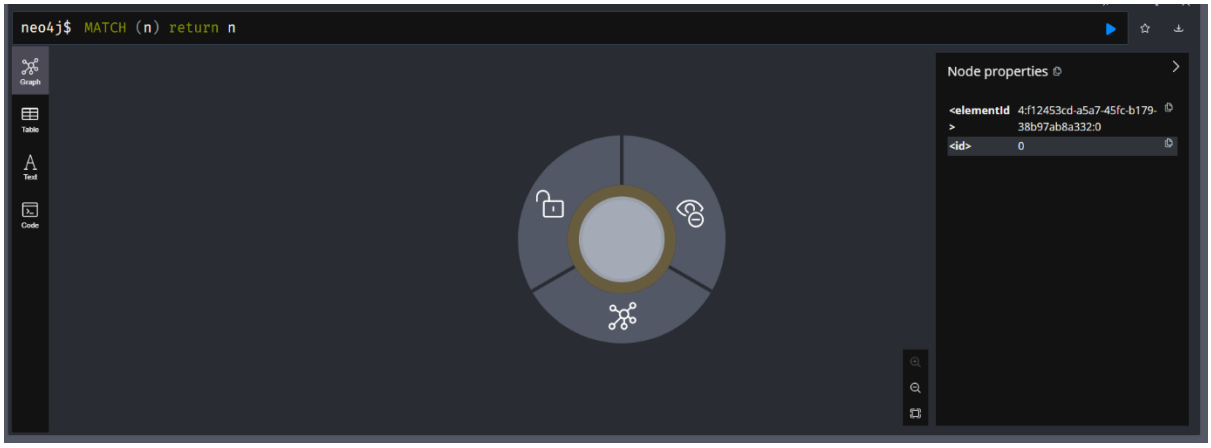
You will be getting the following interface:



1] Creating a node

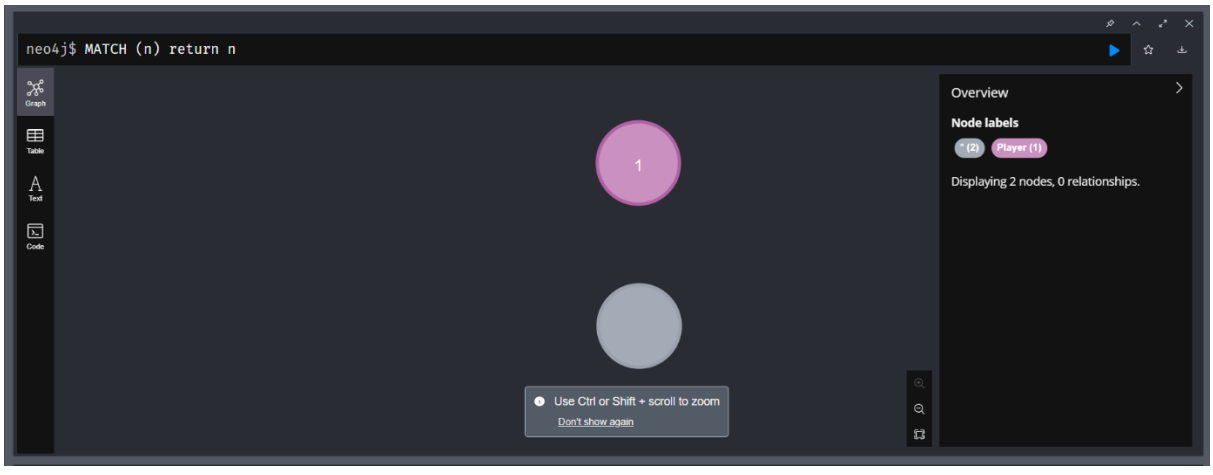


To verify the creation of the node, type and execute the following query in the dollar prompt.

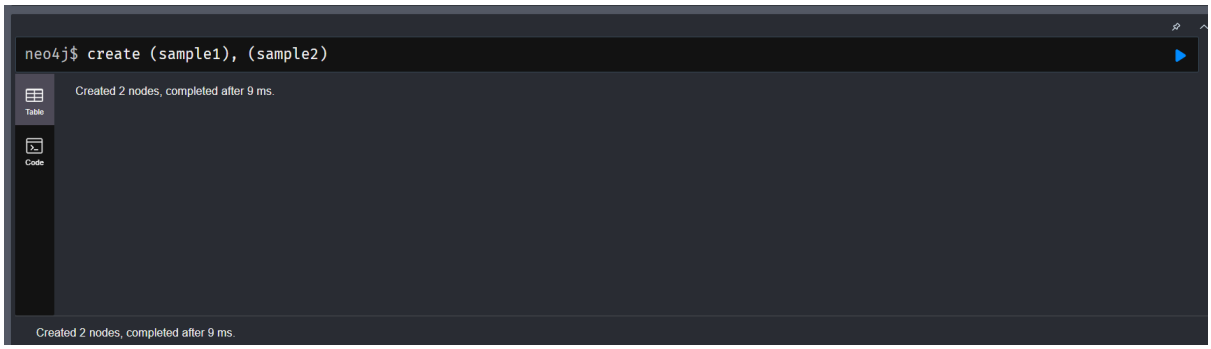


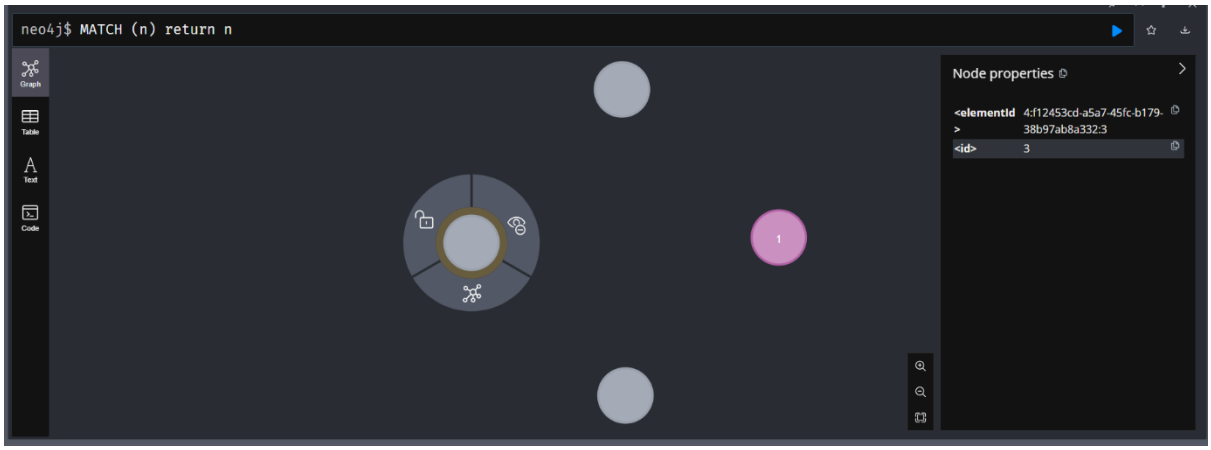
2] Creating a node with a label

Syntax: CREATE (node:label)

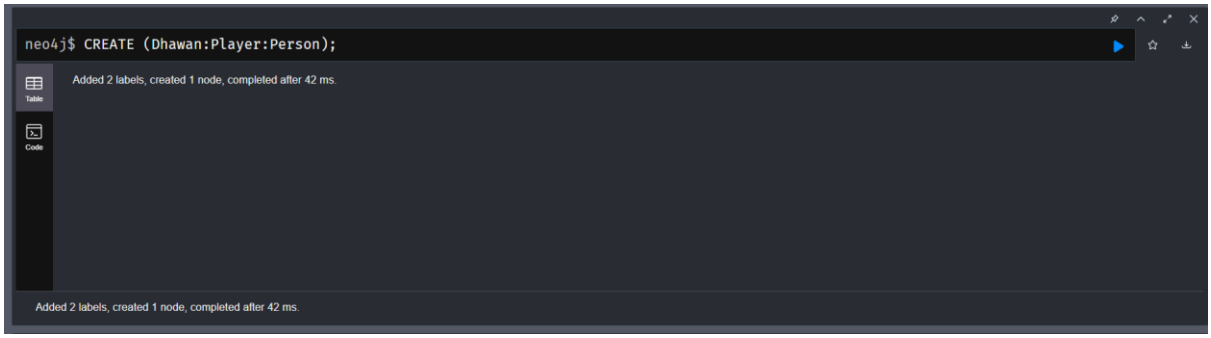


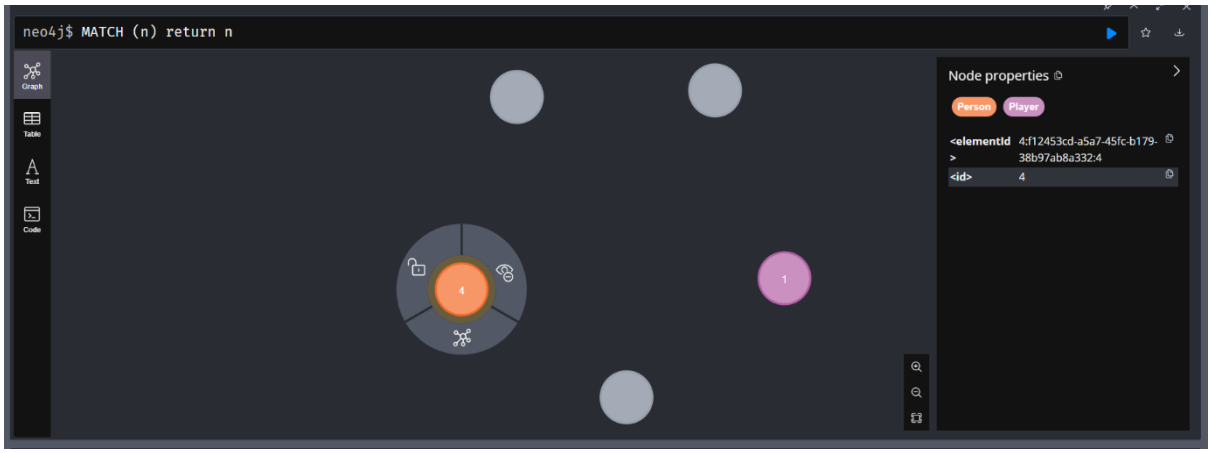
3] Create multiple nodes





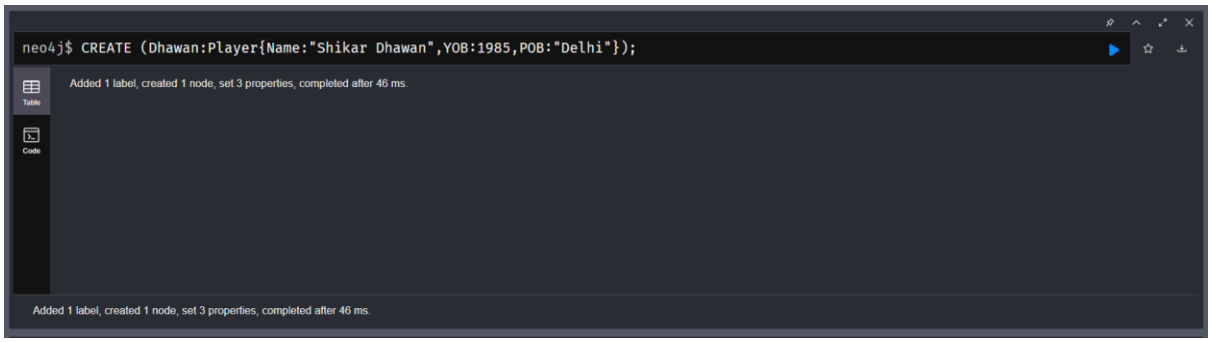
4] Creating a node with multiple labels

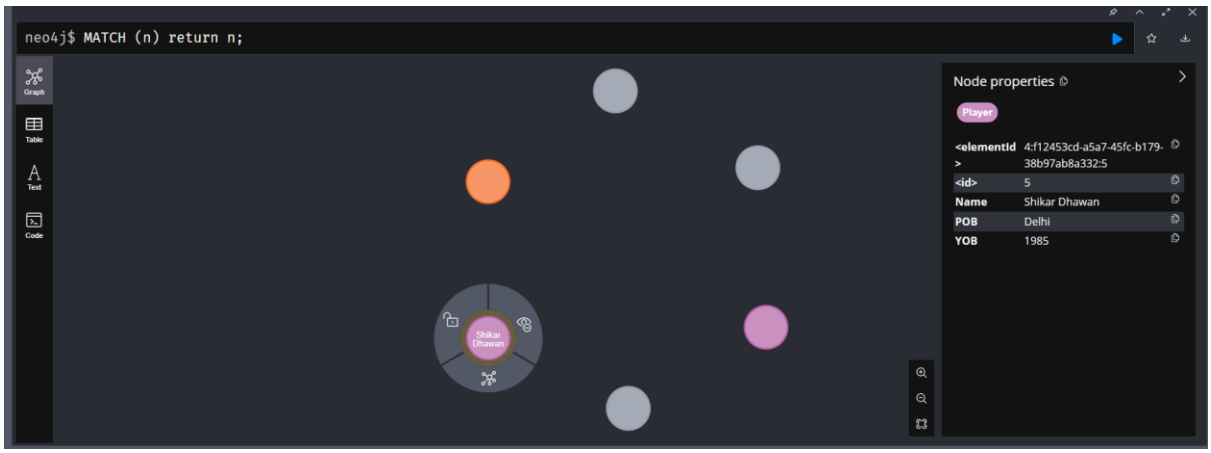




5] Creating a node with properties

Syntax: CREATE (node:label { key1: value, key2: value, ... }

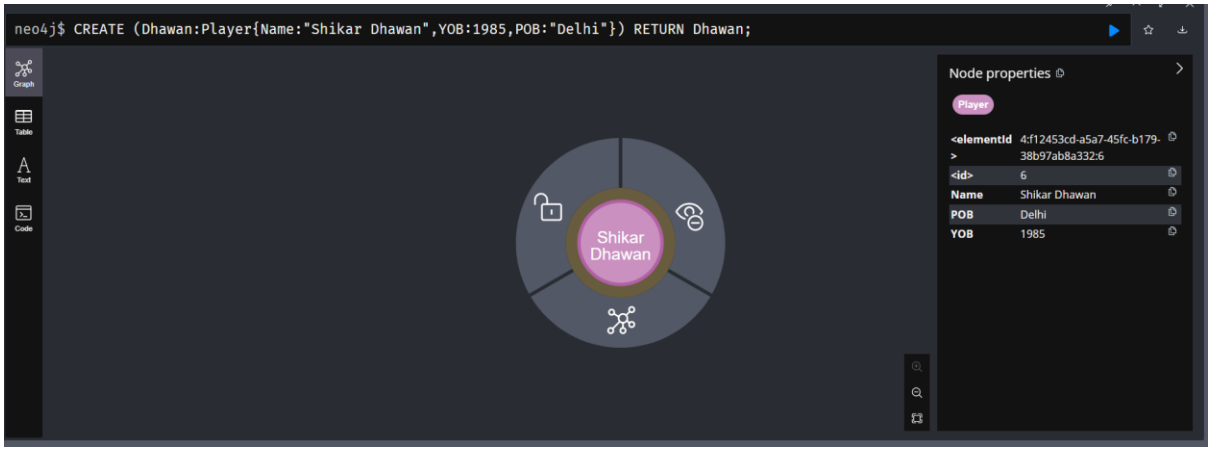




6] Returning the created node

Throughout the chapter, we used the **MATCH (n) RETURN n** query to view the created nodes. This query returns all the existing nodes in the database. Instead of this, we can use the **RETURN** clause with **CREATE** to view the newly created node.

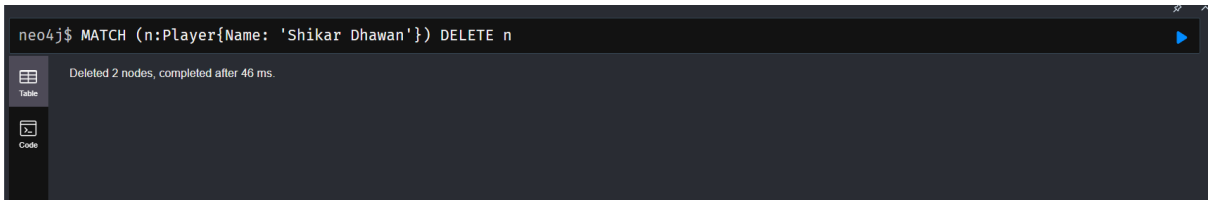
Syntax: CREATE (Node:Label{properties. . . . }) RETURN Node

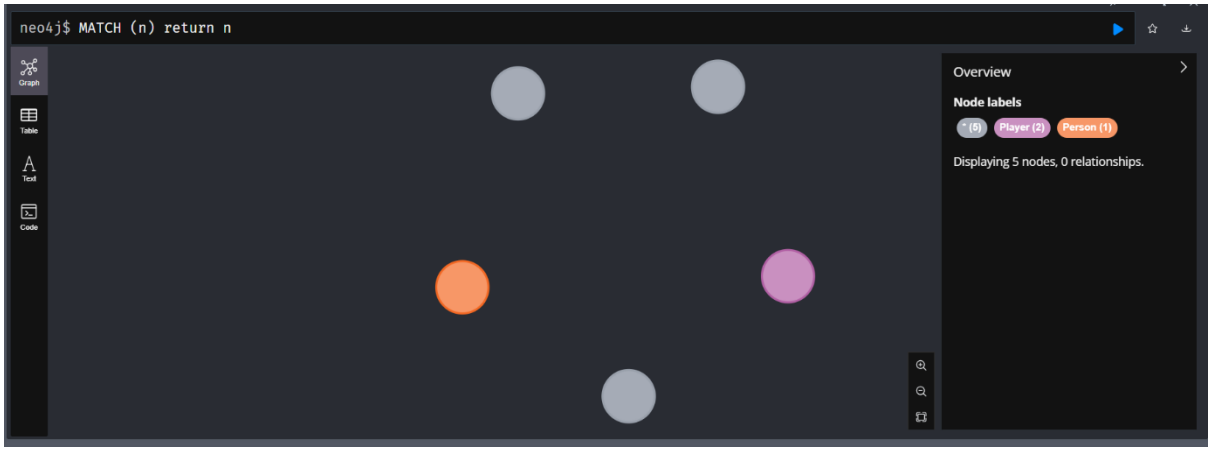


7] Deleting a node

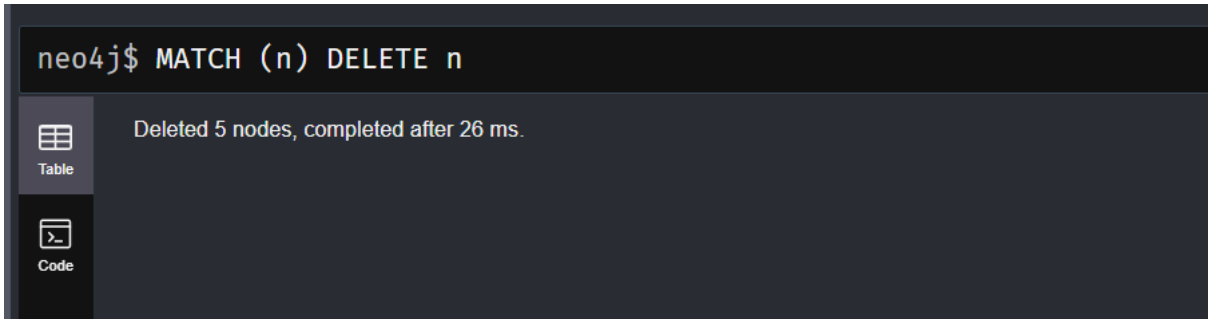
This query is only possible to run on nodes without any relationships connected to them.

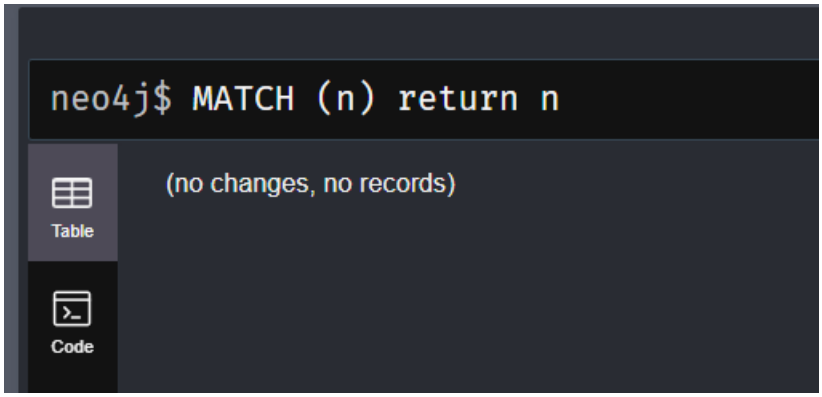
MATCH (n:Player{Name: 'Shikar Dhawan'}) DELETE n;





8] Deleting all nodes



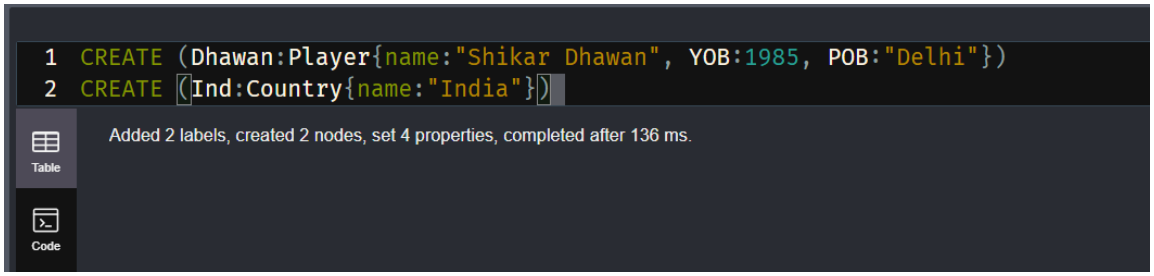


9] Creating a relationship

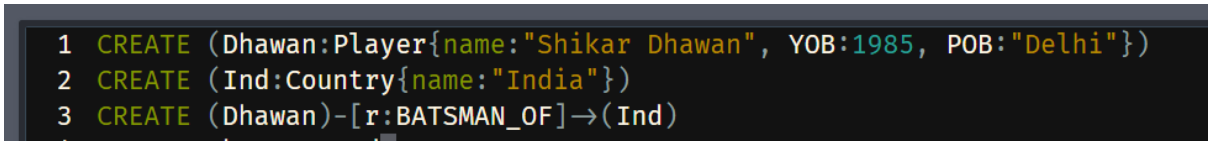
Following is the syntax to create a relationship using the CREATE clause:

CREATE (node1)-[:RelationshipType]->(node2)

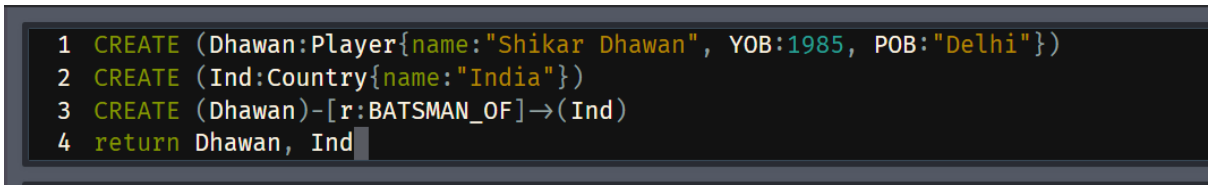
First of all, create two nodes Ind and Dhawan in the database, as shown below



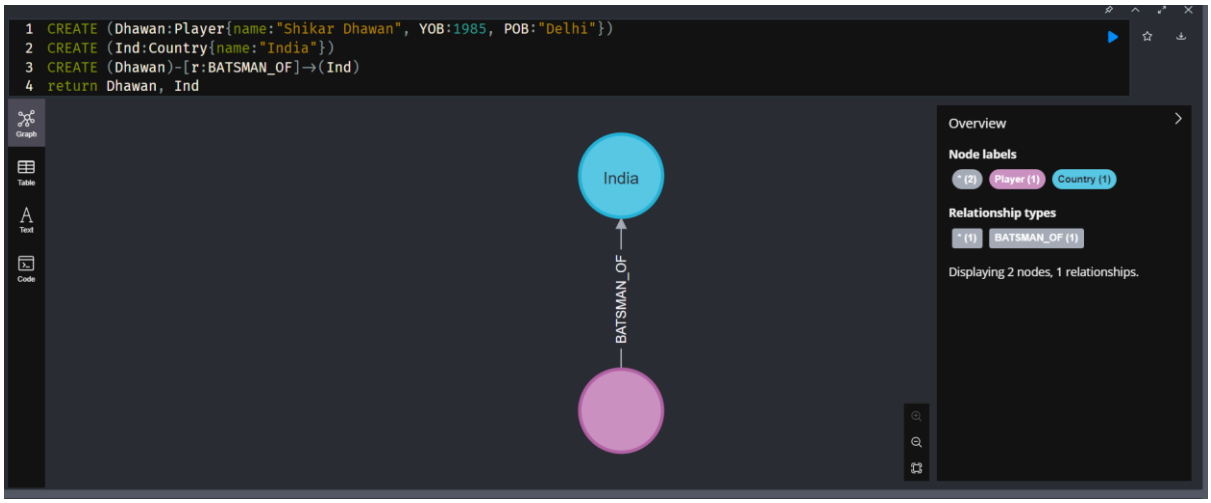
Now, create a relationship named **BATSMAN_OF** between these two nodes as



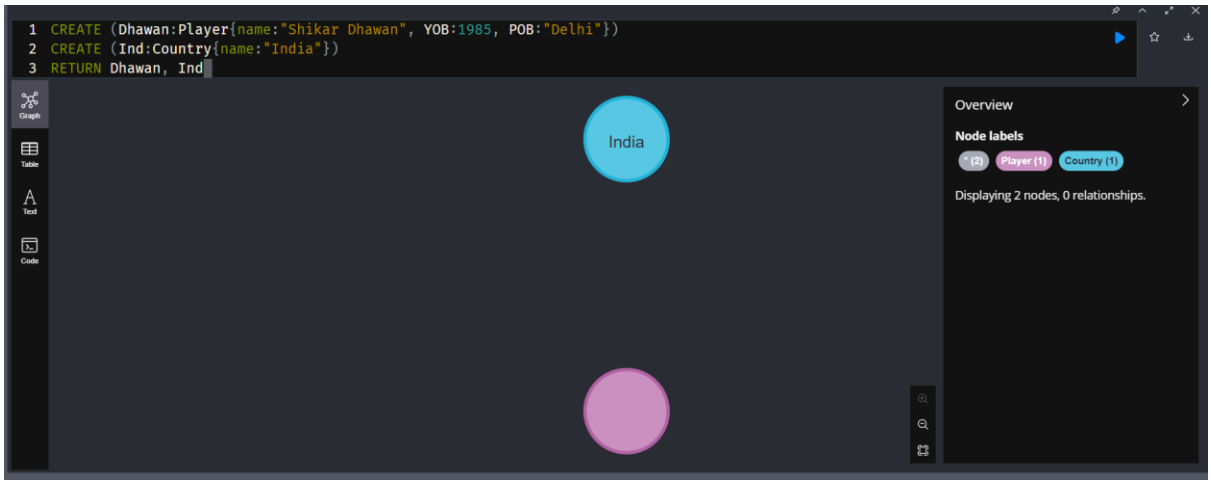
Finally, return both the nodes to see the created relationship.

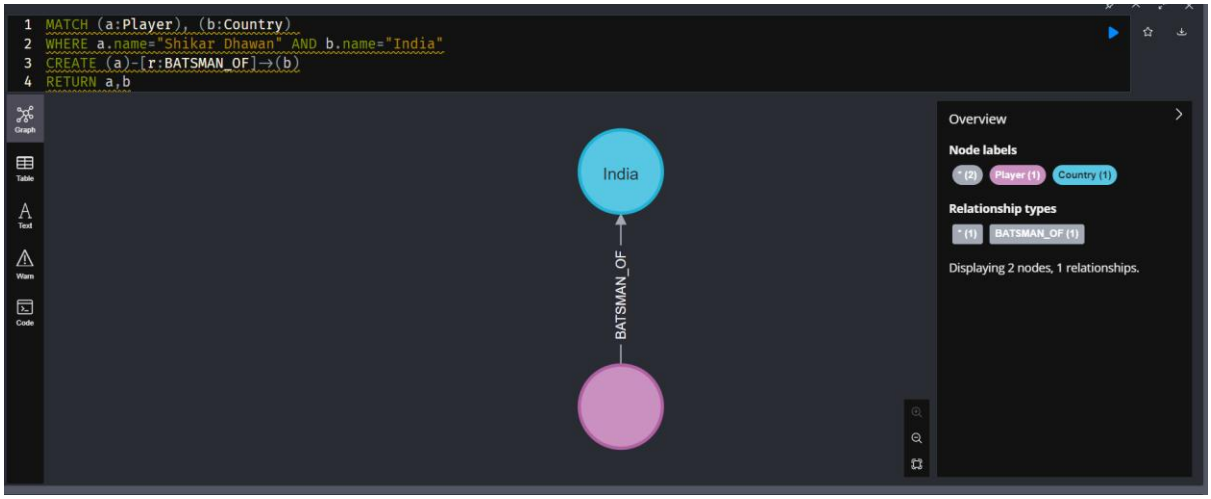


On executing, you will get the following result:

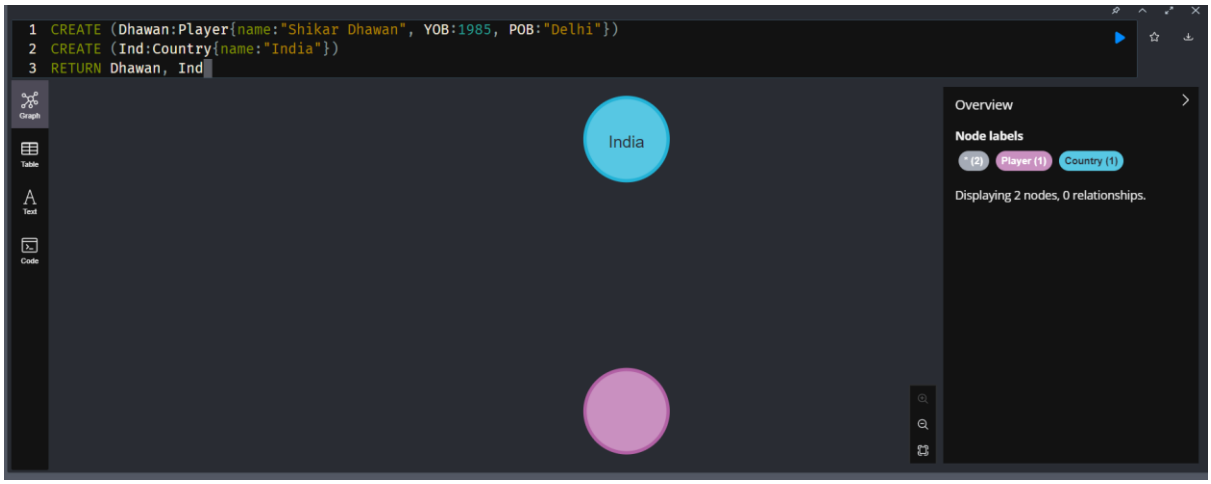


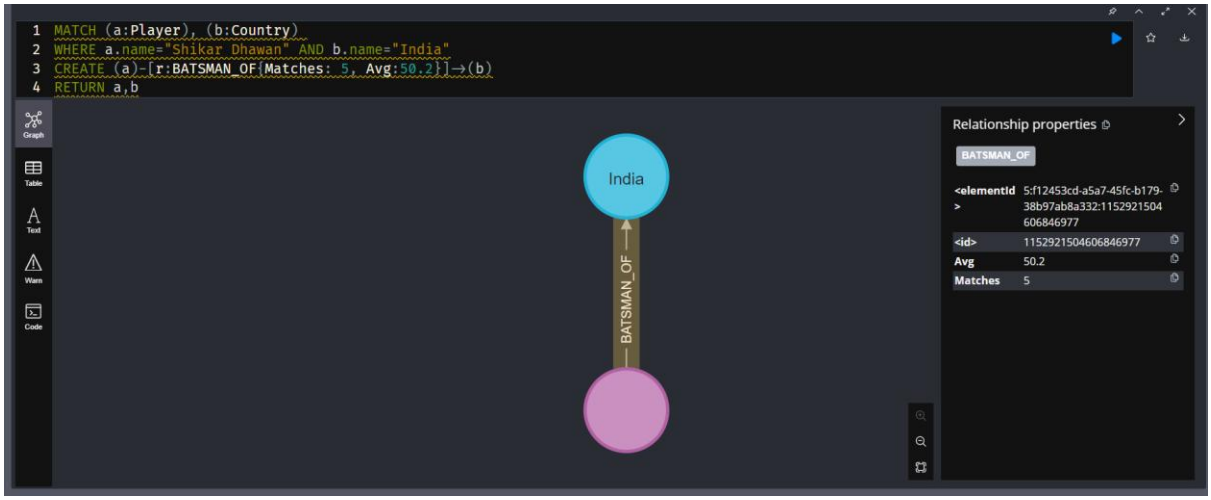
10] Creating a Relationship Between the Existing Nodes.



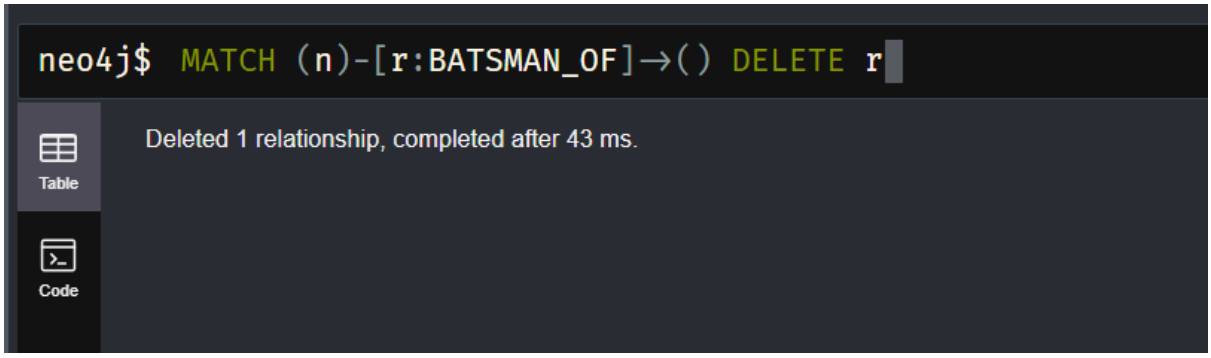


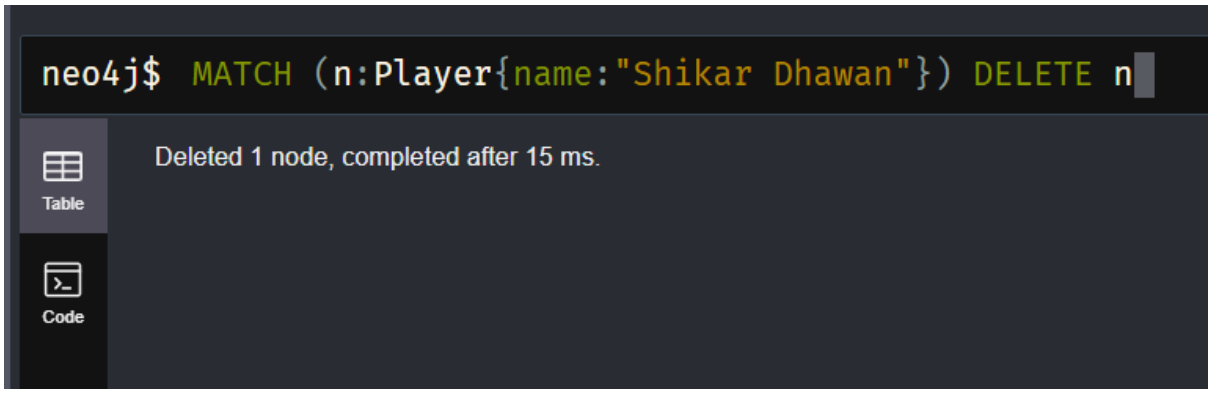
11] Creating a Relationship with Label and Properties



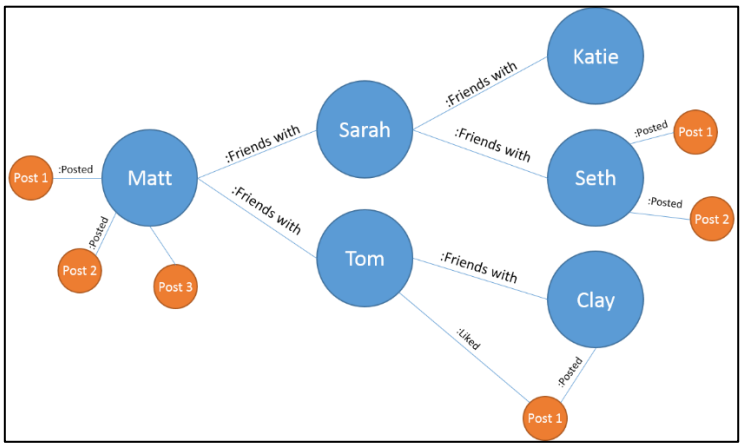


12] Deleting a node that has a relationship



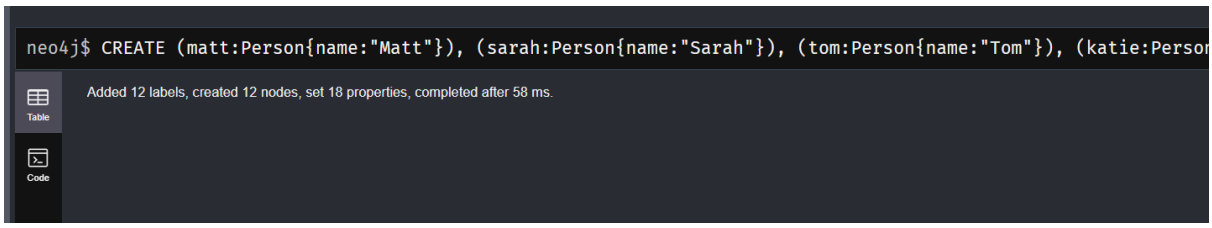


EVEN NUMBERS - Q21 FROM UNIT 3 ASSIGN



a. create the above database.

```
CREATE
(matt:Person{name:"Matt"}),
(sarah:Person{name:"Sarah"}),
(tom:Person{name:"Tom"}),
(katie:Person{name:"Katie"}),
(seth:Person{name:"Seth"}),
(clay:Person{name:"Clay"}),
(post1_matt:Post{name:"Post 1",author:"Matt"}),
(post2_matt:Post{name:"Post 2",author:"Matt"}),
(post3_matt:Post{name:"Post 3",author:"Matt"}),
(post1_seth:Post{name:"Post 1",author:"Seth"}),
(post2_seth:Post{name:"Post 2",author:"Seth"}),
(post1_clay:Post{name:"Post 1",author:"Clay"});
```



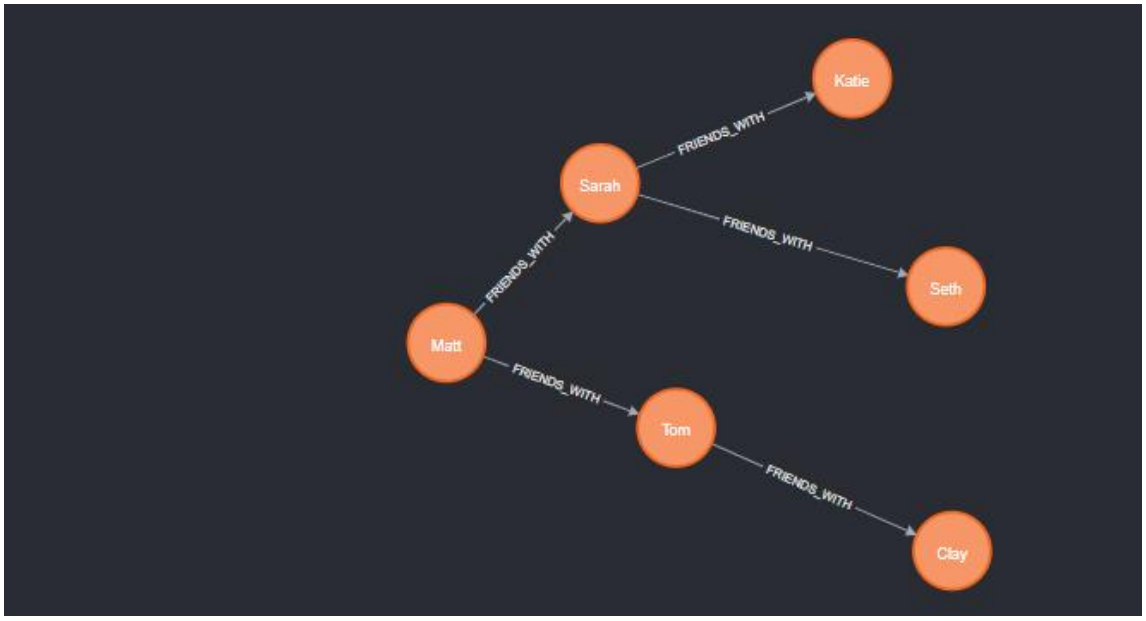

```
MATCH (a:Person), (b:Person)
WHERE a.name="Matt" AND b.name="Sarah"
CREATE (a)-[r:FRIENDS_WITH]->(b)
return a,b

MATCH (a:Person), (b:Person)
WHERE a.name="Matt" AND b.name="Tom"
CREATE (a)-[r:FRIENDS_WITH]->(b)
return a,b

MATCH (a:Person), (b:Person)
WHERE a.name="Sarah" AND b.name="Katie"
CREATE (a)-[r:FRIENDS_WITH]->(b)
return a,b
```

```
MATCH (a:Person), (b:Person)
WHERE a.name="Sarah" AND b.name="Seth"
CREATE (a)-[r:FRIENDS_WITH]->(b)
return a,b
```

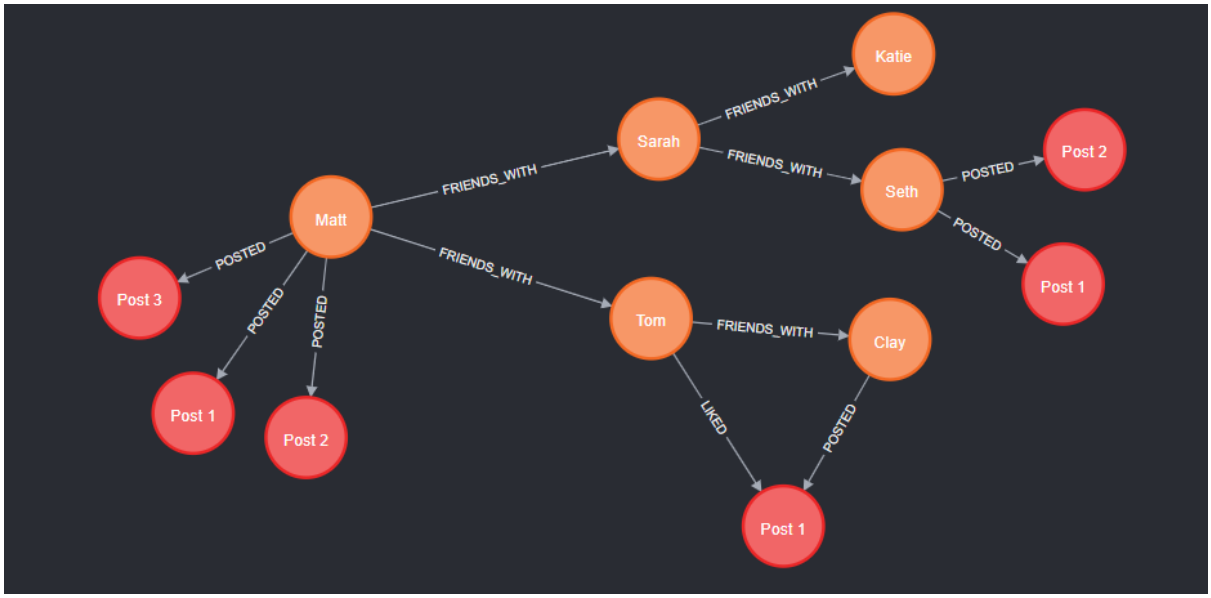
```
MATCH (a:Person), (b:Person)
WHERE a.name="Tom" AND b.name="Clay"
CREATE (a)-[r:FRIENDS_WITH]->(b)
return a,b
```




```
MATCH (a:Person), (b:Post)
WHERE a.name="Matt" AND b.author="Matt"
CREATE (a)-[r:POSTED]->(b)
return a,b
```

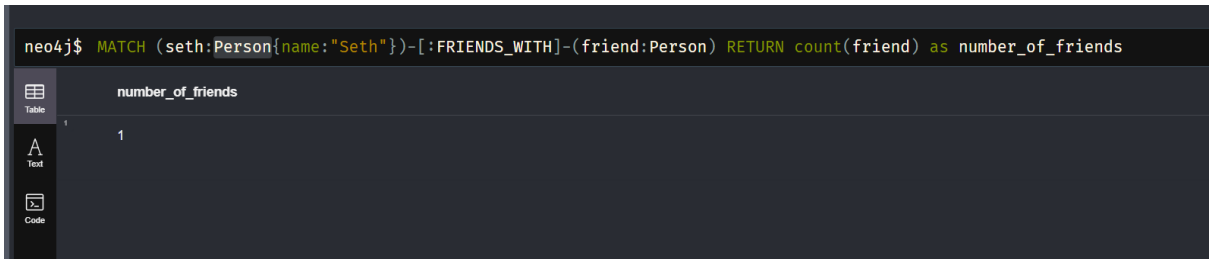
```
MATCH (a:Person), (b:Post)
WHERE a.name="Seth" AND b.author="Seth"
CREATE (a)-[r:POSTED]->(b)
return a,b
```

```
MATCH (a:Person), (b:Post), (c:Person)
WHERE a.name="Clay" AND b.author="Clay" AND c.name="Tom"
CREATE (a)-[:POSTED]->(b)<-[:LIKED]-(c)
return a,b,c
```



b. know that if Katie create any posts?

c. Know how many friends does Seth have?



d. Know that is there a path of friendship between Matt and Clay.