

# DBMS LAB 10

NAME: SUBHAM DEY

SRN: PES1UG23CS604

The screenshot shows the Neo4j browser interface with a query history. The queries are as follows:

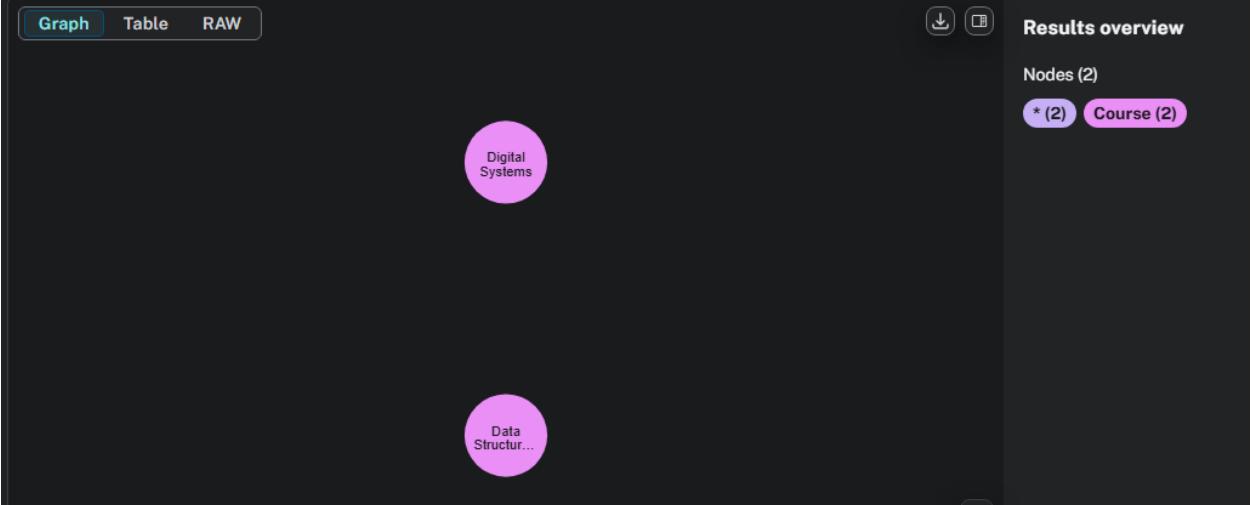
- // 4. Show student enrollments  
MATCH (a:Student {name: 'Alice'}), (cs:Course {code: 'CS101'})  
CREATE (a)-[:ENROLLED\_IN]→(cs);  
Created 1 relationship  
Fetched 0 records
- // 4. Show student enrollments  
MATCH (b:Student {name: 'Bob'}), (ec:Course {code: 'EC202'})  
CREATE (b)-[:ENROLLED\_IN]→(ec);  
Created 1 relationship  
Fetched 0 records
- // 5. Show professor teaching assignments  
MATCH (s:Professor {name: 'Dr. Smith'}), (cs:Course {code: 'CS101'})  
CREATE (s)-[:TEACHES]→(cs);  
Created 1 relationship  
Fetched 0 records
- // 5. Show professor teaching assignments  
MATCH (j:Professor {name: 'Dr. Jones'}), (ec:Course {code: 'EC202'})  
CREATE (j)-[:TEACHES]→(ec);  
Created 1 relationship  
Fetched 0 records
- // 6. Create friendship  
MATCH (a:Student {name: 'Alice'}), (c:Student {name: 'Charlie'})  
CREATE (a)-[:FRIENDS\_WITH]→(c);  
Created 1 relationship  
Fetched 0 records

The screenshot shows the Neo4j browser interface with a query history. The queries are as follows:

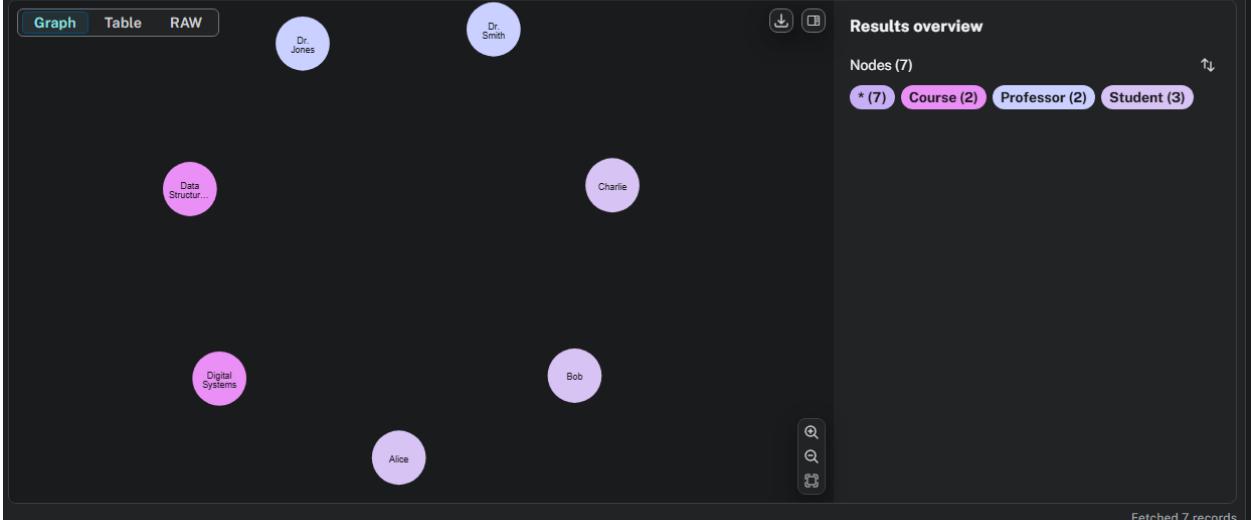
- // 1. Create Student Nodes  
CREATE (a:Student {name: 'Alice', age: 21, major: 'CSE'}),  
(b:Student {name: 'Bob', age: 22, major: 'ECE'}),  
(c:Student {name: 'Charlie', age: 20, major: 'CSE'});  
Created 3 nodes, set 9 properties, added 3 labels  
Fetched 0 records
- // 2. Create Professor Nodes  
CREATE (s:Professor {name: 'Dr. Smith', department: 'CSE'}),  
(j:Professor {name: 'Dr. Jones', department: 'ECE'});  
Created 2 nodes, set 4 properties, added 2 labels  
Fetched 0 records
- // 3. Create Course Nodes  
CREATE (cs:Course {code: 'CS101', name: 'Data Structures'}),  
(ec:Course {code: 'EC202', name: 'Digital Systems'});  
Created 2 nodes, set 4 properties, added 2 labels  
Fetched 0 records

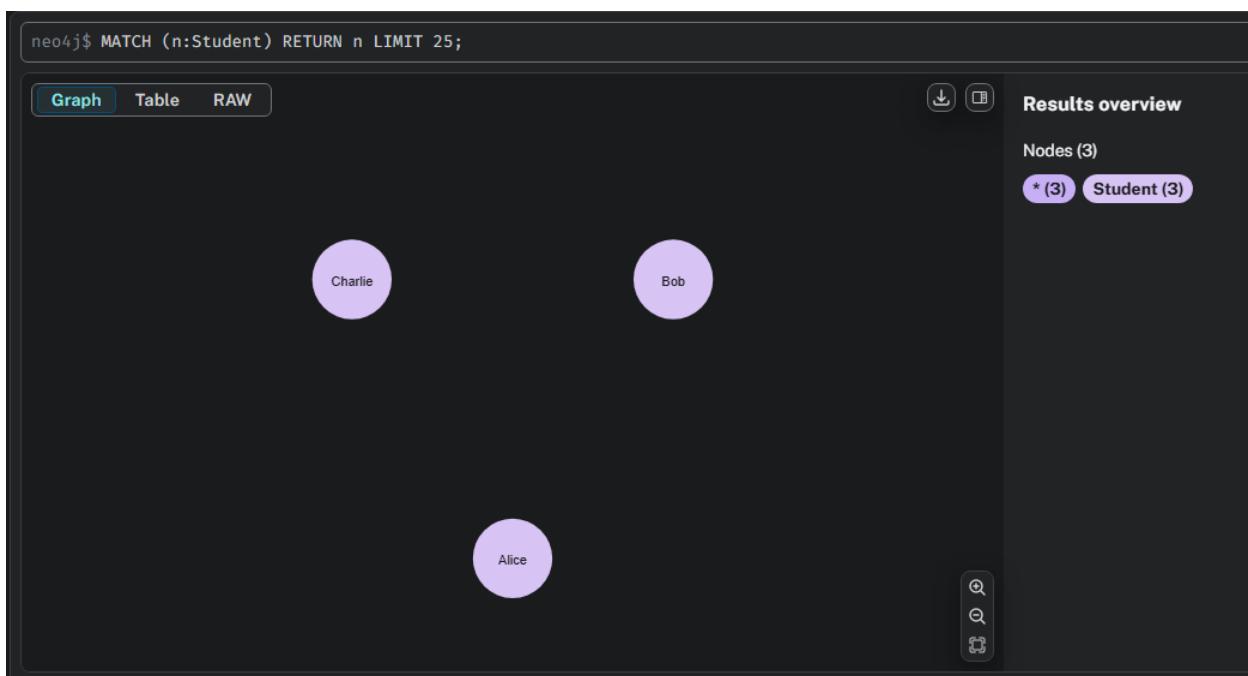
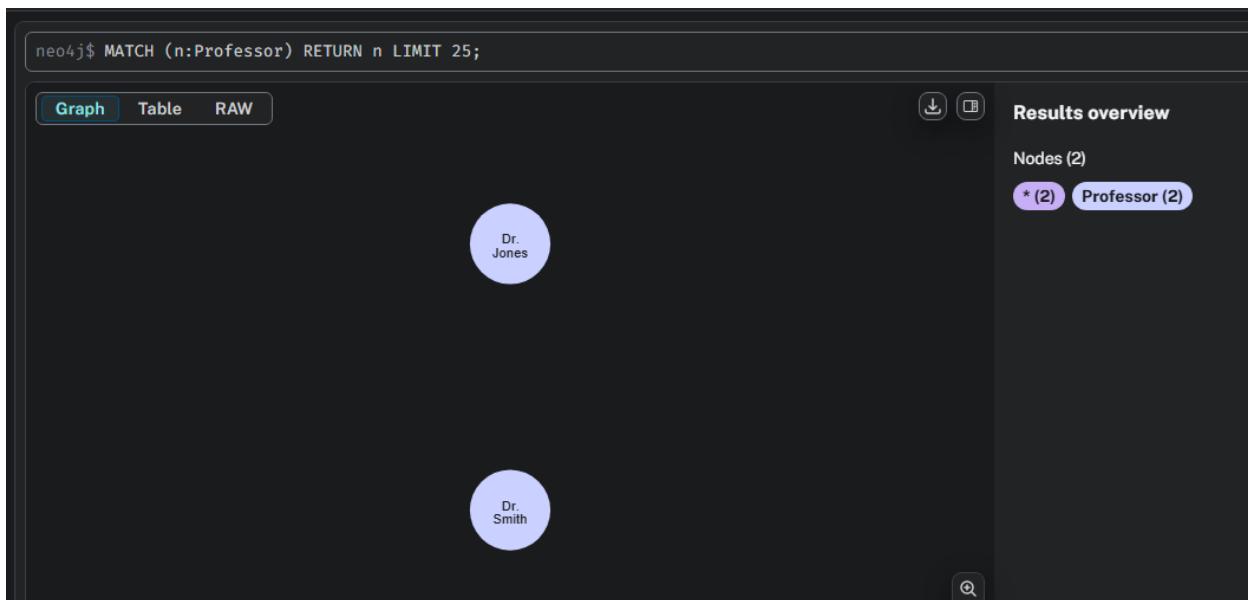
\$ :welcome

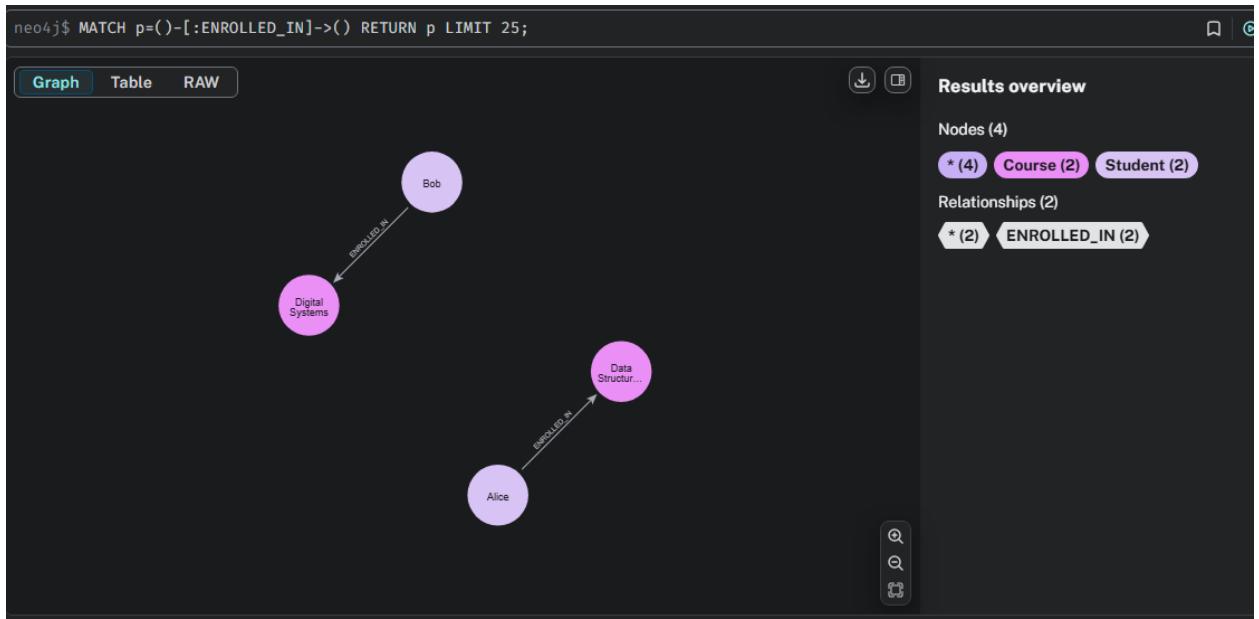
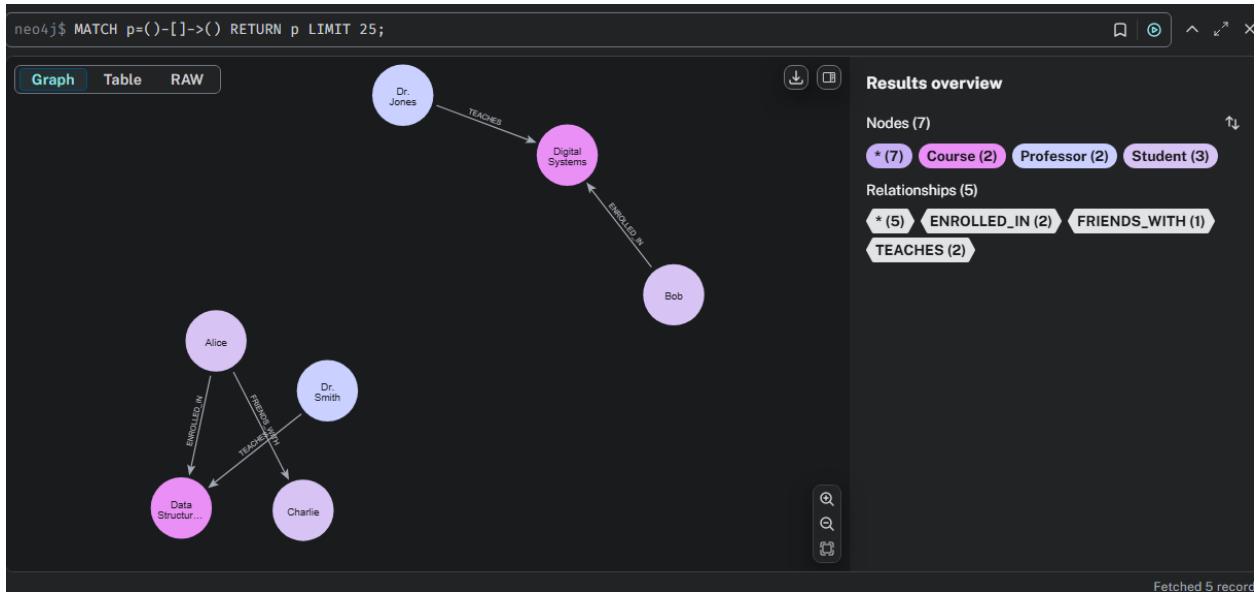
```
neo4j$ MATCH (n:Course) RETURN n LIMIT 25;
```

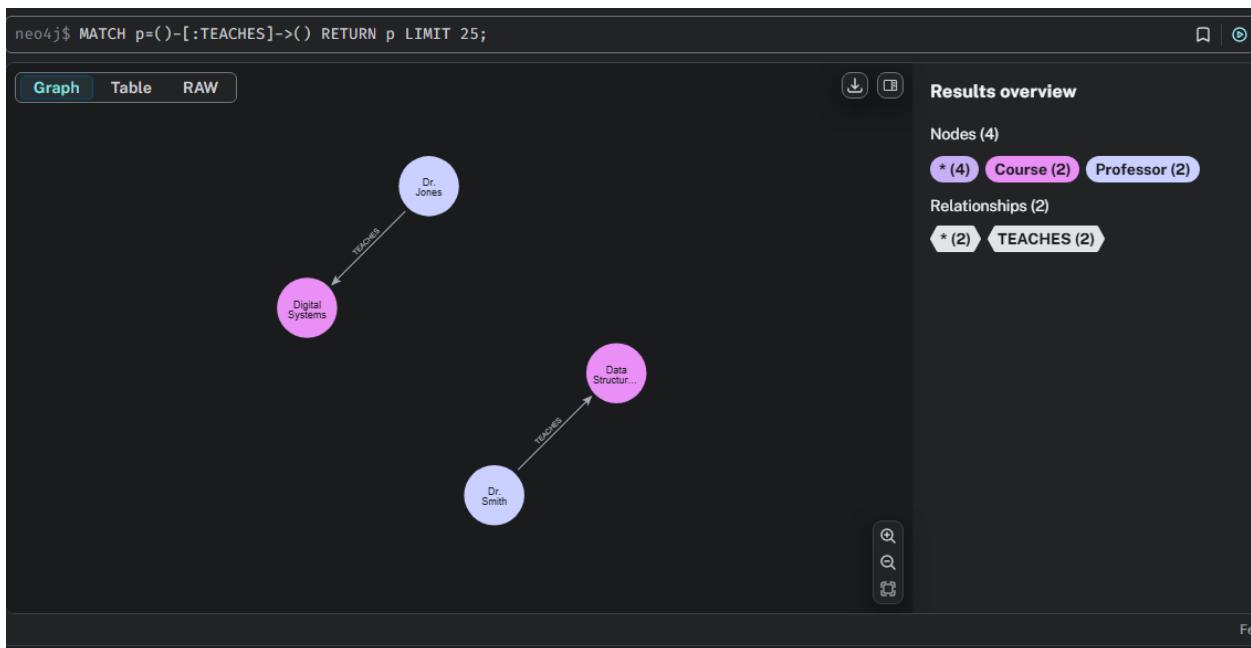
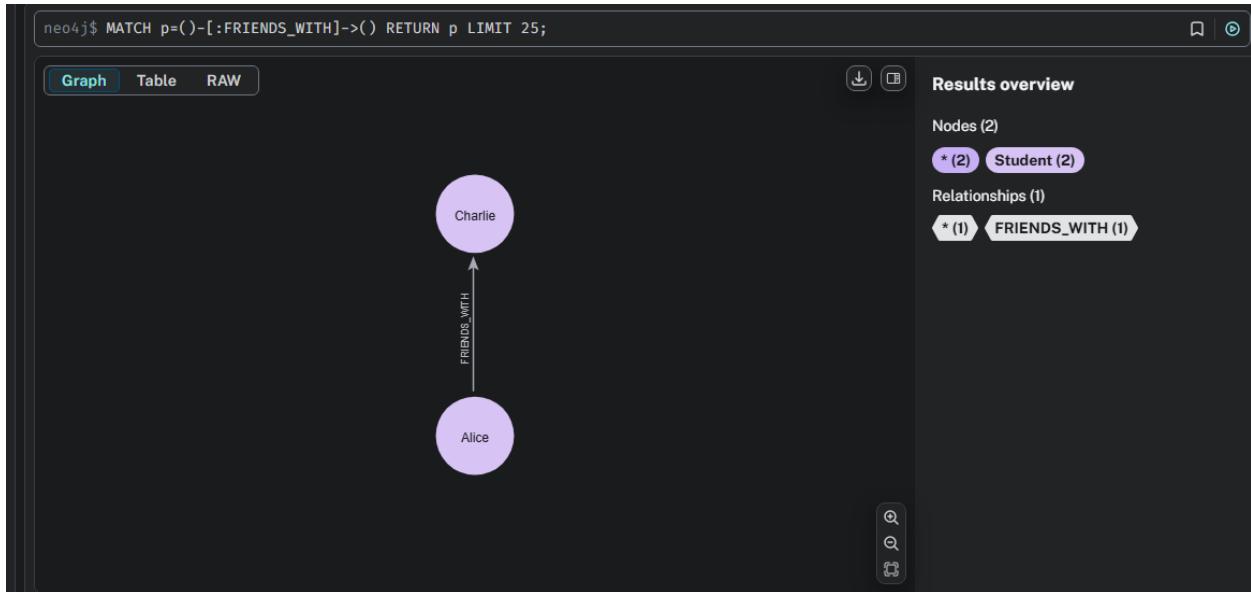


```
neo4j$ MATCH (n) RETURN n LIMIT 25;
```

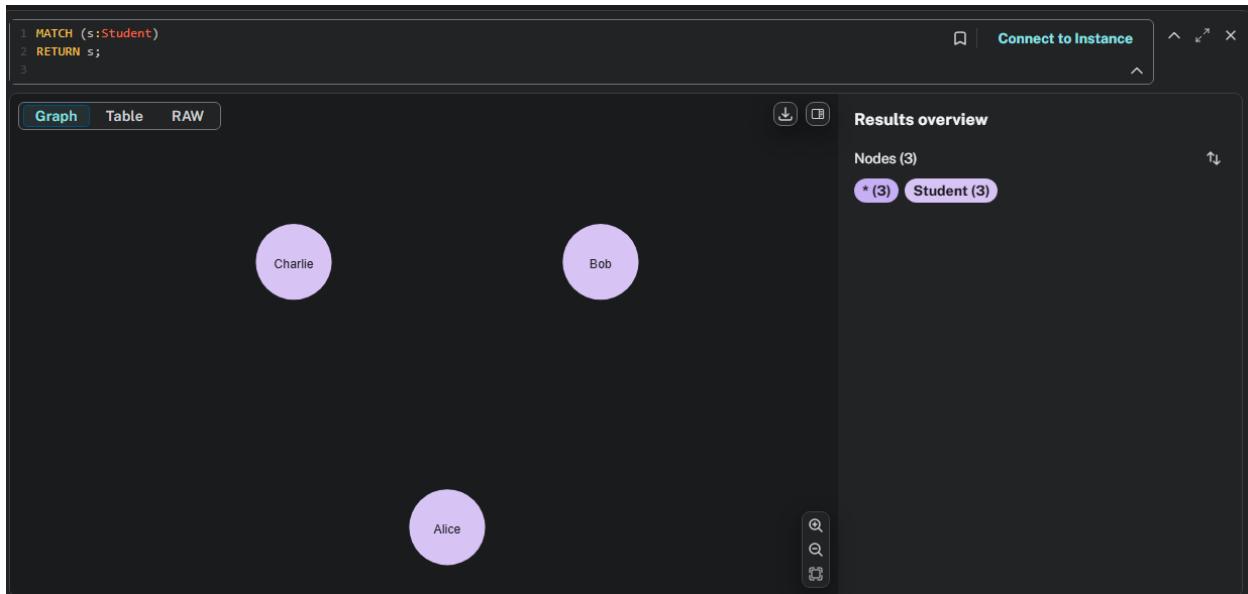




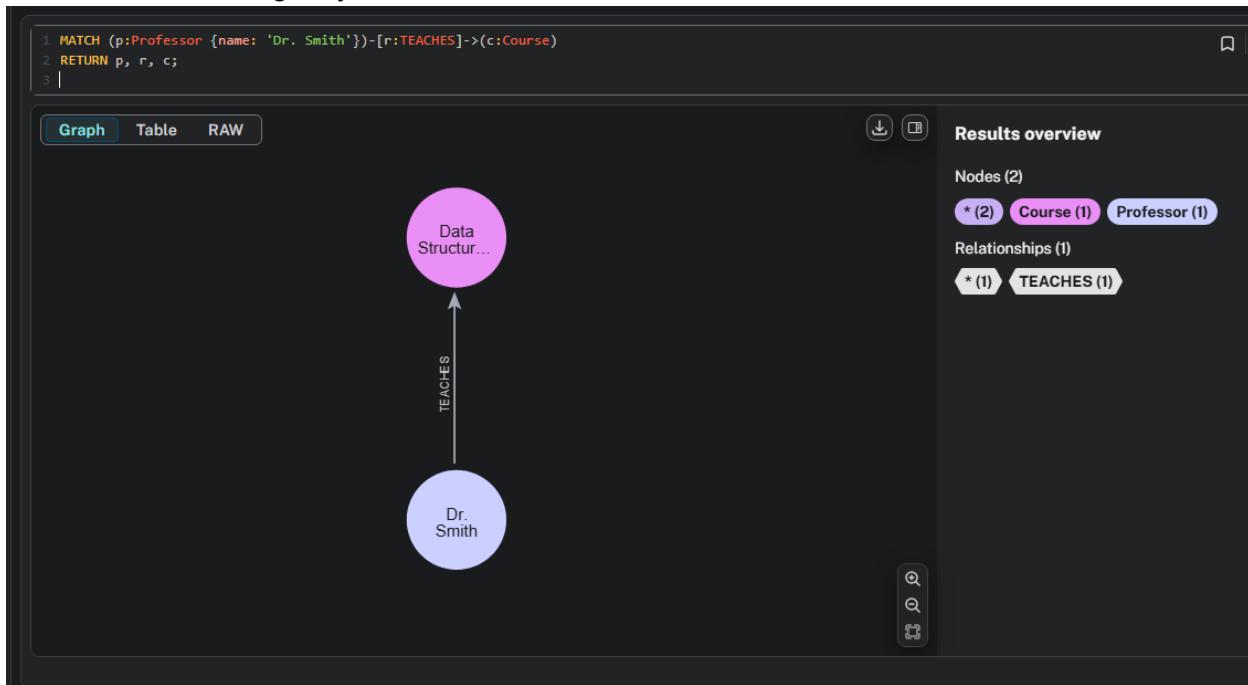




## 7. List All Students



## 8. Find Courses Taught by Dr. Smith



## 9. Find Friends of Charlie

```

1 MATCH (c:Student {name: 'Charlie'})-[r:FRIENDS_WITH]-{friend:Student}
2 RETURN c, r, friend;
3

```

Graph Table RAW

Results overview

Nodes (2)

- \* (2) Student (2)

Relationships (1)

- \* (1) FRIENDS\_WITH (1)

## 10. List All Students in the Same Course

```

1 MATCH (s1:Student)-[r1:ENROLLED_IN]->(c:Course)<-[r2:ENROLLED_IN]-(s2:Student)
2 WHERE elementId(s1) < elementId(s2)
3 RETURN s1, r1, c, r2, s2;

```

No changes, no records

## 11. Find Professors Who Teach Alice's Courses

```

MATCH (a:Student {name: 'Alice'})-[r1:ENROLLED_IN]->(c:Course)<-[r2:TEACHES]-(p:Professor)
RETURN a, r1, c, r2, p;

```

Graph Table RAW

Results overview

Nodes (3)

- \* (3) Course (1) Professor (1) Student (1)

Relationships (2)

- \* (2) ENROLLED\_IN (1) TEACHES (1)

12. Find Students Who Are Friends and Enrolled in the Same Course

```
1 MATCH (s1:Student)-[r1:FRIENDS_WITH]-(s2:Student),
2     (s1)-[r2:ENROLLED_IN]->(c:Course)<-[r3:ENROLLED_IN]-(s2:Student)
3 WHERE elementId(s1) < elementId(s2)
4 RETURN s1, r1, s2, r2, c, r3;
```

No changes, no records

Find

13. Courses with More Than One Student Enrolled

```
1 MATCH (s:Student)-[:ENROLLED_IN]->(c:Course)
2 WITH c, count(s) AS studentCount
3 WHERE studentCount > 1
4 MATCH (s_final:Student)-[r_final:ENROLLED_IN]->(c)
5 RETURN s_final, r_final, c;
```

No changes, no records

14. Count how many students each professor teaches.

```
1 MATCH (p:Professor)-[:TEACHES]->(c:Course)<-[ :ENROLLED_IN]-(s:Student)
2 RETURN p.name AS Professor, count(DISTINCT s) AS NumberOfStudents;
```

Table

RAW

Professor	NumberOfStudents
1 "Dr. Smith"	1
2 "Dr. Jones"	1