

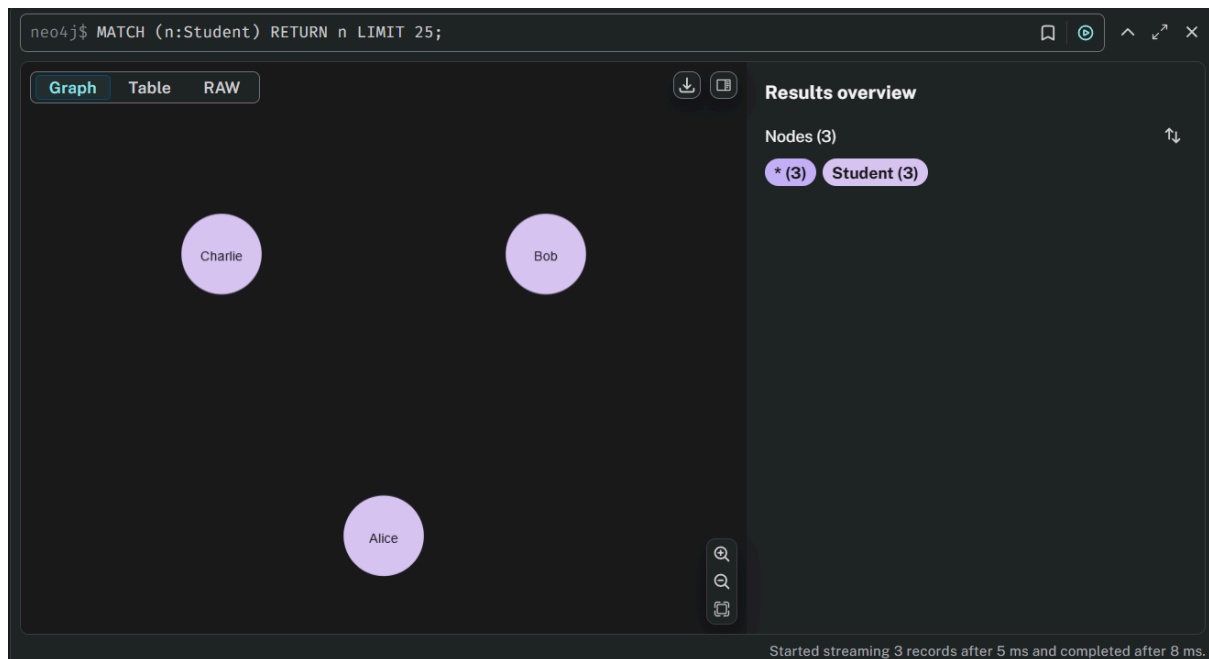
# Lab - 10

## NEO4J

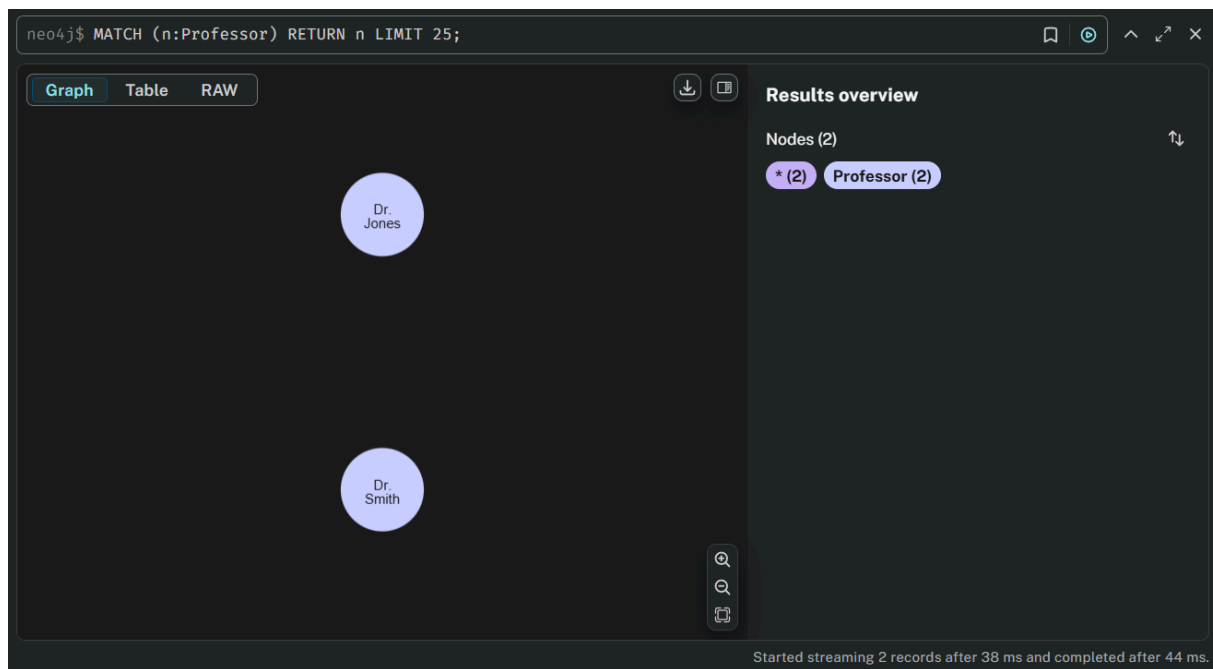
Name	Anirudh
SRN	PES1UG23CS077
Sec	B
Date	31-10-2025

### Part A: Create Nodes:

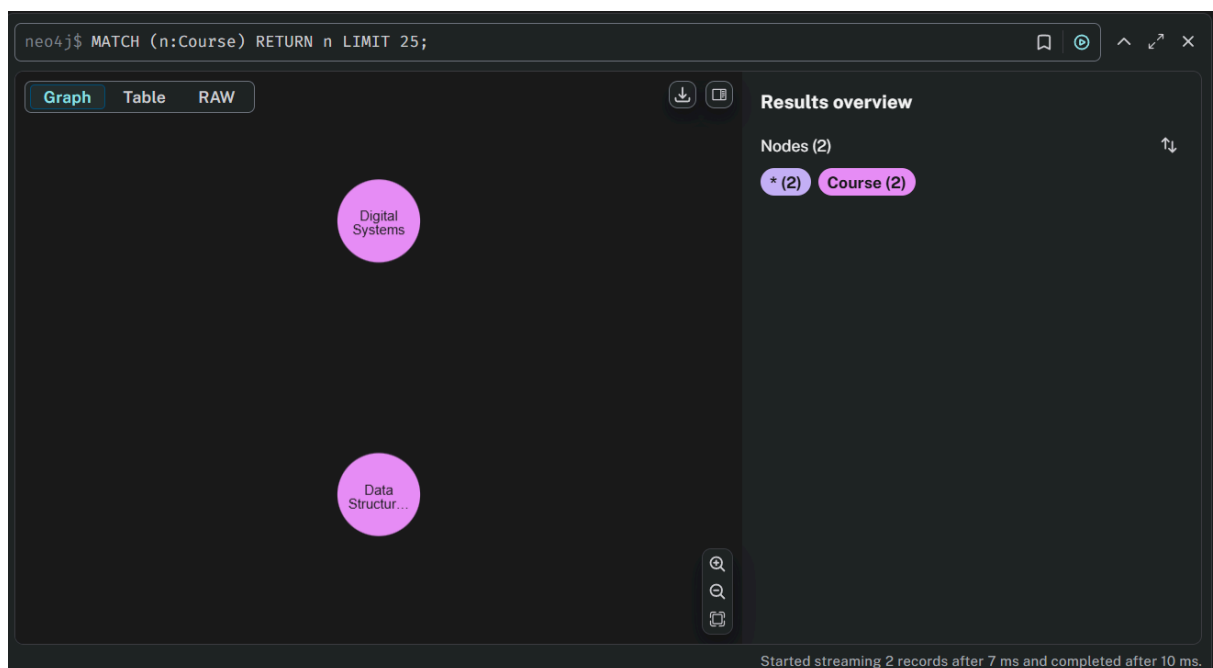
1. Create Student Nodes with name, age and major. We have Alice, 21, CSE & Bob, 22, ECE and Charlie, 20, CSE.



2.Create Professor Nodes, Dr. Smith for the CSE department and Dr. Jones for the ECE department

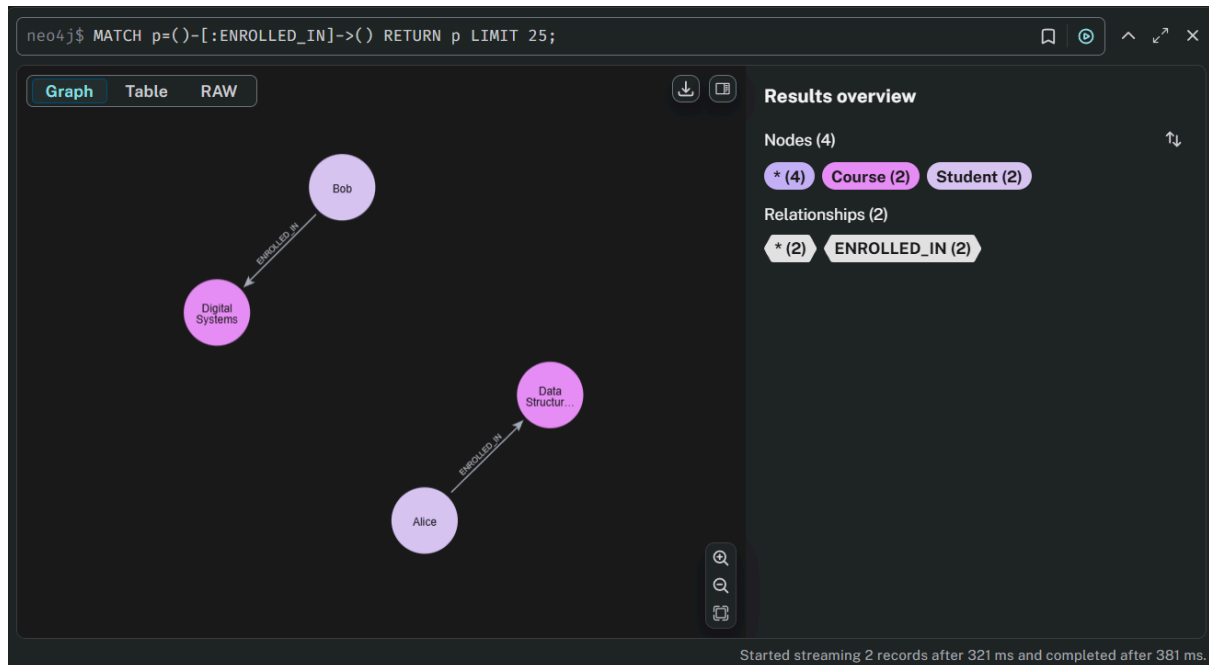


3.Create Course Nodes (CODE: CS101, DATA STRUCTURES & CODE: EC202, DIGITAL SYSTEMS)

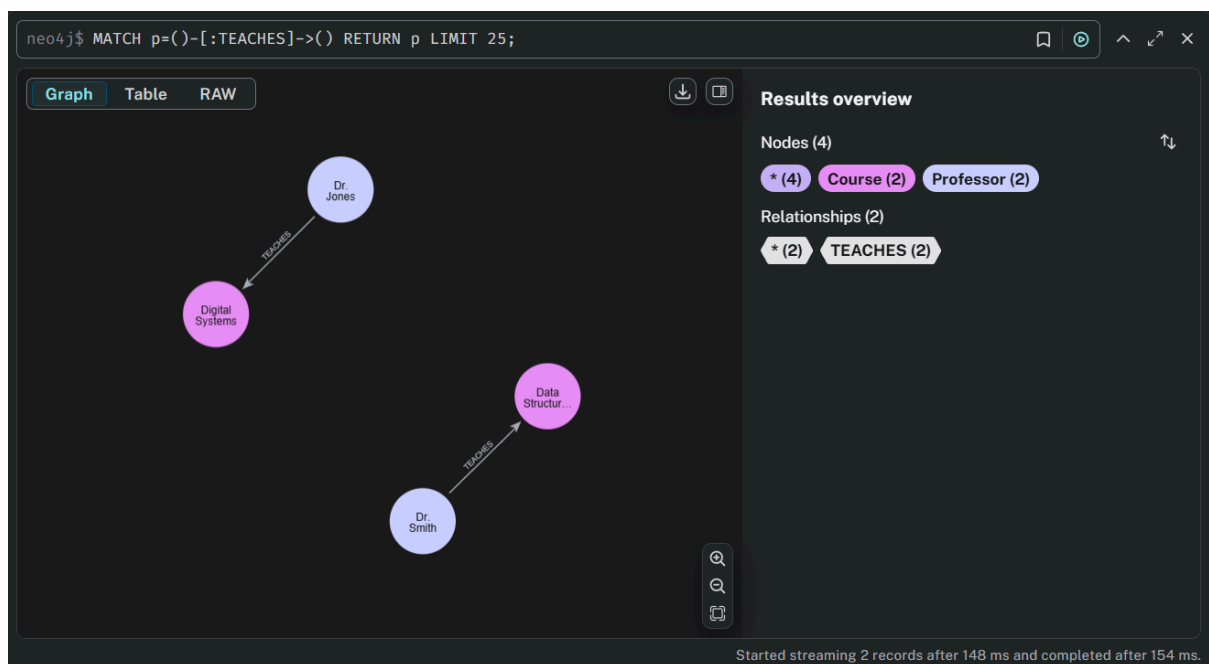


## Part B: Create Relationships

4. Show that Alice has enrolled into “CS101” and Bob has enrolled into “EC202”



5. Show that professor Dr. Smith teaches “CS101” and Dr. Jones teaches “EC202”



## 6. Create friendship between Alice and Charlie

The screenshot shows the Neo4j Desktop interface. The main graph view displays two nodes, Alice and Charlie, connected by a relationship labeled FRIEND\_WITH. The right sidebar, titled 'Results overview', shows the query results: 2 nodes (Student) and 1 relationship (FRIEND\_WITH). The query in the top bar is: `neo4j$ MATCH p=()-[:FRIEND_WITH]->() RETURN p LIMIT 25;`

## Created Nodes and relationships:

The screenshot shows the Neo4j Desktop interface with a list of database information on the left and a series of Cypher queries executed in the main panel. The queries are as follows:

- `neo4j$ MATCH (a:Student {name: 'Alice'}), (c:Student {name: 'Charlie'}) CREATE (a)-[:FRIEND_WITH]->(c);` Completed after 137 ms
- `neo4j$ MATCH (j:Professor {name: 'Dr. Jones'}), (ec:Course {code: 'EC202'}) CREATE (j)-[:TEACHES]->(ec);` Completed after 141 ms
- `neo4j$ MATCH (s:Professor {name: 'Dr. Smith'}), (cs:Course {code: 'CS101'}) CREATE (s)-[:TEACHES]->(cs);` Completed after 147 ms
- `neo4j$ MATCH (b:Student {name: 'Bob'}), (ec:Course {code: 'EC202'}) CREATE (b)-[:ENROLLED_IN]->(ec);` Completed after 149 ms
- `neo4j$ MATCH (a:Student {name: 'Alice'}), (cs:Course {code: 'CS101'}) CREATE (a)-[:ENROLLED_IN]->(cs);` Completed after 311 ms
- `neo4j$ CREATE (cs:Course {code: 'CS101', title: 'Data Structures'}), (ec:Course {code: 'EC202', title: 'Digital Systems'});` Completed after 98 ms
- `neo4j$ CREATE (s:Professor {name: 'Dr. Smith', department: 'CSE'}), (j:Professor {name: 'Dr. Jones', department: 'ECE'});` Completed after 90 ms
- `neo4j$ 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'});` Completed after 184 ms

The interface also shows a 'Database information' sidebar on the left with a tree view of nodes and relationships.

## Part C: Query the Graph:

### 7. List All Students:

```
neo4j$ MATCH (s:Student) RETURN s.name AS Student, s.age AS Age, s.major AS Major;
```

	Student	Age	Major
1	"Alice"	21	"CSE"
2	"Bob"	22	"ECE"
3	"Charlie"	20	"CSE"

Started streaming 3 records after 18 ms and completed after 36 ms.

### 8. Find Courses Taught by Dr. Smith

```
neo4j$ MATCH (p:Professor {name: "Dr. Smith"})-[:TEACHES]->(c:Course) RETURN c.title AS CourseTaughtByDr;
```

	CourseTaughtByDr
1	"Data Structures"

Started streaming 1 record after 1,850 ms and completed after 1,967 ms.

### 9. Find Friends of Charlie

```
neo4j$ MATCH (s:Student {name: "Charlie"})<-[:FRIEND_WITH]-(friend) RETURN friend.name AS FriendsOfCharlie;
```

	FriendsOfCharlie
1	"Alice"

Started streaming 1 record after 268 ms and completed after 273 ms.

### 10. List All Students in the Same Course

```
neo4j$ MATCH (s1:Student)-[:ENROLLED_IN]->(c:Course)<-[:ENROLLED_IN]-(s2:Student) WHERE s1.name < s2.name
```

No changes, no records

Completed after 786 ms

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

```
neo4j$ MATCH (a:Student {name: "Alice"})-[:ENROLLED_IN]->(c:Course)-[:TEACHES]-(p:Professor) RETURN p.name
```

	Professor	CourseTaughtToAl
1	"Dr. Smith"	"Data Structures"

Started streaming 1 record after 303 ms and completed after 306 ms

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

```
neo4j$ MATCH (s1:Student)-[:FRIEND_WITH]-(s2:Student), (s1)-[:ENROLLED_IN]->(c:Course), (s2)-[:ENROLLED_IN]->(c:Course)
```

No changes, no records

Completed after 537 ms

## 13. Find Courses with More Than One Student Enrolled

```
neo4j$ MATCH (s:Student)-[:ENROLLED_IN]->(c:Course) WITH c, COUNT(s) AS num_students WHERE num_students > 1
```

No changes, no records

Completed after 530 ms

## 14. Count how many students each professor teaches.

```
neo4j$ MATCH (p:Professor)-[:TEACHES]->(c:Course)-[:ENROLLED_IN]-(s:Student) RETURN p.name AS Professor, COUNT(s) AS TotalStudents
```

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

Started streaming 2 records after 527 ms and completed after 529 ms.