

Profesional Social Media Database

Khushi Patt

October 2025

Introduction

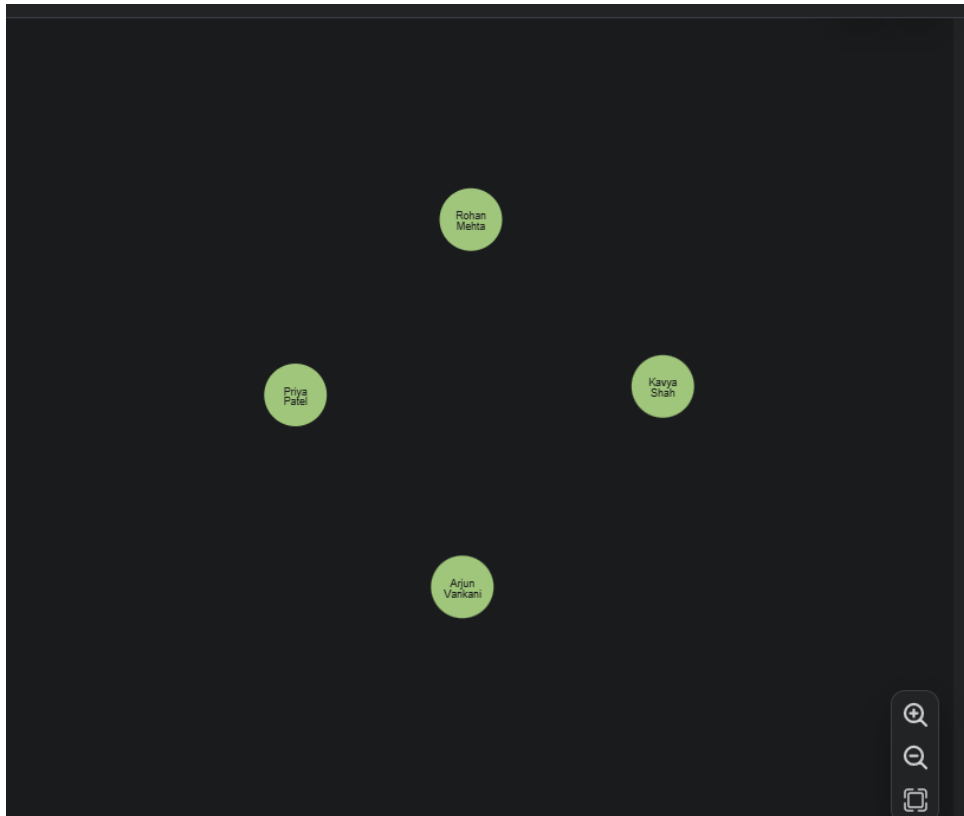
This project is something I built using Neo4j to create a simple graph database that works like LinkedIn. I track users, their skills, jobs, and companies, and it helps me find connections, suggest new contacts, and match users to jobs.

1:Modeling Nodes Creation

Create nodes for Users,Companies and skills with given properties

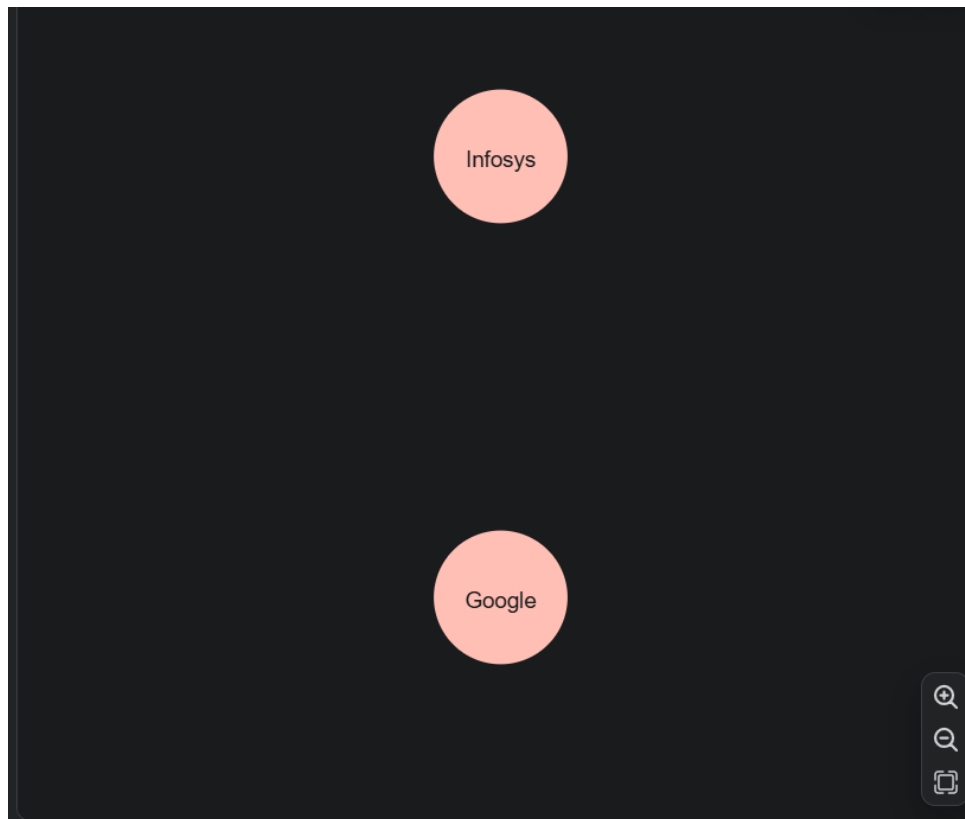
```
CREATE
(:User {name:'Arjun Vankani', title:'Data Scientist', location:'
    Ahmedabad'}),
(:User {name:'Kavya Shah', title:'Software Engineer', location:'
    Mumbai'}),
(:User {name:'Rohan Mehta', title:'ML Engineer', location:'Delhi'
    }),
(:User {name:'Priya Patel', title:'Business Analyst', location:'
    Bangalore'});
```

The above code is used to create node:Users.



```
CREATE
(:Company {name:'Google', industry:'Technology'}),
(:Company {name:'Infosys', industry:'IT Services'});
```

The above code is used to create node:Companies.



```
CREATE
(:Skill {name:'Python'}),
(:Skill {name:'Machine Learning'}),
(:Skill {name:'Big Data'}),
(:Skill {name:'SQL'}),
(:Skill {name:'Business Analysis'});
```

The above code is used to create node:Skills.

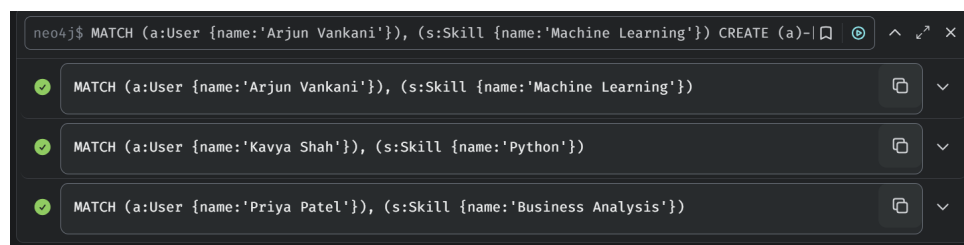


2: Creating Relationships

Link nodes according to the relationships described.

```
MATCH (a:User {name:'Arjun Vankani'}), (s:Skill {name:'Machine Learning'})
CREATE (a)-[:HAS_SKILL]->(s);
MATCH (a:User {name:'Kavya Shah'}), (s:Skill {name:'Python'})
CREATE (a)-[:HAS_SKILL]->(s);
MATCH (a:User {name:'Priya Patel'}), (s:Skill {name:'Business Analysis'})
CREATE (a)-[:HAS_SKILL]->(s);
```

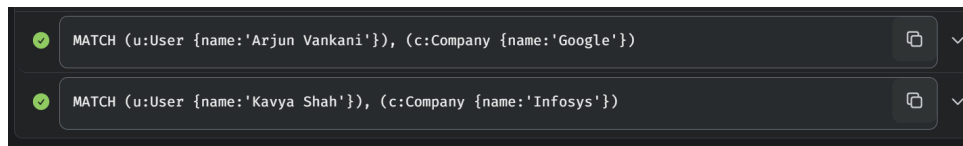
The above code is used to connect users with skills.



```
MATCH (u:User {name:'Arjun Vankani'}), (c:Company {name:'Google'})
CREATE (u)-[:WORKS_AT]->(c);
```

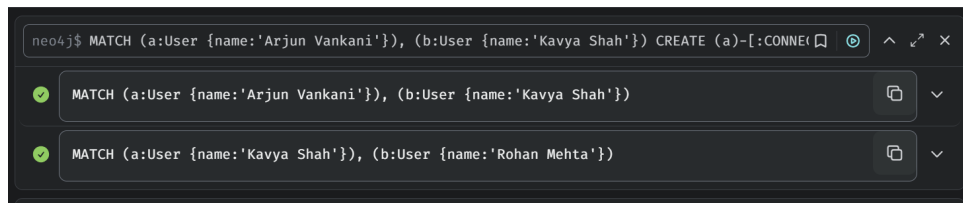
```
MATCH (u:User {name:'Kavya Shah'}), (c:Company {name:'Infosys'})
CREATE (u)-[:WORKS_AT]->(c);
```

The above code is used to connect users with companies.



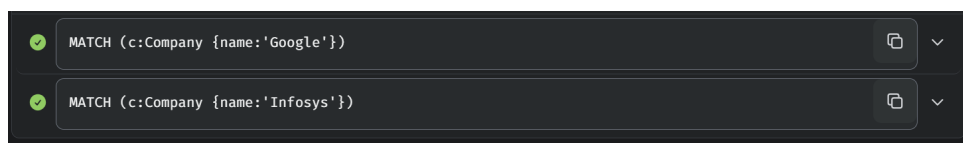
```
MATCH (a:User {name:'Arjun Vankani'}), (b:User {name:'Kavya Shah'})
CREATE (a)-[:CONNECTED_TO]->(b);
MATCH (a:User {name:'Kavya Shah'}), (b:User {name:'Rohan Mehta'})
CREATE (a)-[:CONNECTED_TO]->(b);
```

The above code is used to create connections between users.



```
MATCH (c:Company {name:'Google'})
CREATE (c)-[:POSTED]->(Job {title:'ML Engineer', location:'Bangalore'});
MATCH (c:Company {name:'Infosys'})
CREATE (c)-[:POSTED]->(Job {title:'Data Analyst', location:'Pune'});
```

The above code is used to users with their job postions.



```
MATCH (u:User {name:'Arjun Vankani'}), (j:Job {title:'ML Engineer'})
CREATE (u)-[:APPLIED_TO]->(j);
MATCH (u:User {name:'Priya Patel'}), (j:Job {title:'Data Analyst'})
CREATE (u)-[:APPLIED_TO]->(j);
```

The above code is used to connect users with the job positions they applied for.

```
neo4j$ MATCH (u:User {name:'Arjun Vankani'}), (j:Job {title:'ML Engineer'}) CREATE (u)-[:APPL:
MATCH (u:User {name:'Arjun Vankani'}), (j:Job {title:'ML Engineer'})
MATCH (u:User {name:'Priya Patel'}), (j:Job {title:'Data Analyst'})
```

Task 3: Query Examples

3.1. Find all connections of Arjun

```
MATCH (u:User {name:'Arjun Vankani'})-[:CONNECTED_TO]->(f:User)
RETURN f.name;
```

Table	RAW
f.name	
"Kavya Shah"	

Started streaming 1 record after 5 ms and completed after 10 ms.

3.2. Suggest new connections (friends-of-friends for Arjun)

```
MATCH (u:User {name:'Arjun Vankani'})-[:CONNECTED_TO]->(f)-[:CONNECTED_TO]->(fof)
WHERE fof <> u AND NOT (u)-[:CONNECTED_TO]->(fof)
RETURN DISTINCT fof.name AS SuggestedConnection;
```

Table	RAW
SuggestedConnec	
"Rohan Mehta"	

Started streaming 1 record after 13 ms and completed after 22 ms.

3.3. Find all users with Machine Learning skill

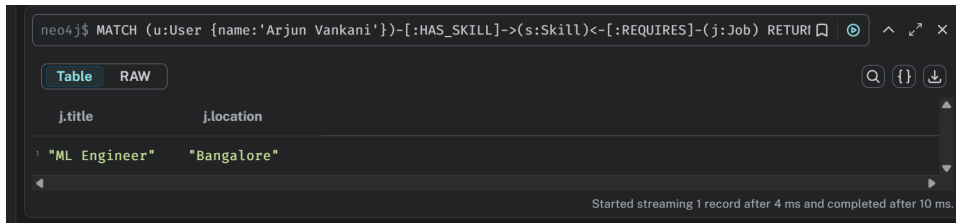
```
MATCH (u:User)-[:HAS_SKILL]->(s:Skill {name:'Machine Learning'})
RETURN u.name, u.title;
```

Table	RAW
u.name	u.title
"Arjun Vankani"	"Data Scientist"

Started streaming 1 record after 23 ms and completed after 25 ms.

3.4. Find jobs that match Arjun's skills

```
MATCH (u:User {name:'Arjun Vankani'})-[:HAS_SKILL]->(s:Skill)-[:REQUIRES]-(j:Job)
RETURN j.title, j.location;
```



The screenshot shows the Neo4j query results interface. The query is: `neo4j$ MATCH (u:User {name:'Arjun Vankani'})-[:HAS_SKILL]->(s:Skill)-[:REQUIRES]-(j:Job) RETURN j.title, j.location;`. The results are displayed in a table with two columns: `j.title` and `j.location`. The first row shows `"ML Engineer"` and `"Bangalore"`. The interface also includes a search bar, a toggle for 'Table' and 'RAW' views, and a status bar at the bottom indicating 'Started streaming 1 record after 4 ms and completed after 10 ms.'

j.title	j.location
"ML Engineer"	"Bangalore"

This output is found after using 'REQUIRES' to get a relationship between job and skills

3.5 List all employees of Google

```
MATCH (u:User)-[:WORKS_AT]->(c:Company {name:'Google'})
RETURN u.name, u.title;
```



The screenshot shows the Neo4j query results interface. The query is: `neo4j$ MATCH (u:User)-[:WORKS_AT]->(c:Company {name:'Google'}) RETURN u.name, u.title;`. The results are displayed in a table with two columns: `u.name` and `u.title`. The first row shows `"Arjun Vankani"` and `"Data Scientist"`. The interface also includes a search bar, a toggle for 'Table' and 'RAW' views, and a status bar at the bottom indicating 'Started streaming 1 record after 9 ms and completed after 11 ms.'

u.name	u.title
"Arjun Vankani"	"Data Scientist"

Conclusion

This project taught me how to build a LinkedIn-like network using Neo4j. I created users, skills, jobs, and connections to show how they relate. Writing it up in LaTeX helped me explain it clearly. It was a great way to learn both graph databases and documentation.