Comprehensive Neo4j Cypher Query Guide: Social Network (Users & Friendships)

Using your dataset: `Users` *and* `friendship_pairs` *to demonstrate real-world queries.*

📌 Dataset Recap

Nodes: Users

cypher Copy

```cypher
CREATE (:User {id: 0, name: "Hero"}),
       (:User {id: 1, name: "Dunn"}),
       (:User {id: 2, name: "Sue"}),
       (:User {id: 3, name: "Chi"}),
       (:User {id: 4, name: "Thor"}),
       (:User {id: 5, name: "Clive"}),
       (:User {id: 6, name: "Hicks"}),
       (:User {id: 7, name: "Devin"}),
       (:User {id: 8, name: "Kate"}),
       (:User {id: 9, name: "Klein"})
```

Relationships: Friendships

cypher Copy

```cypher
MATCH (u1:User), (u2:User)
WHERE (u1.id = 0 AND u2.id = 1) OR
      (u1.id = 0 AND u2.id = 2) OR
      (u1.id = 1 AND u2.id = 2) OR
      (u1.id = 1 AND u2.id = 3) OR
      (u1.id = 2 AND u2.id = 3) OR
      (u1.id = 3 AND u2.id = 4) OR
      (u1.id = 4 AND u2.id = 5) OR
      (u1.id = 5 AND u2.id = 6) OR
      (u1.id = 5 AND u2.id = 7) OR
      (u1.id = 6 AND u2.id = 8) OR
      (u1.id = 7 AND u2.id = 8) OR
      (u1.id = 8 AND u2.id = 9)
CREATE (u1)-[:FRIENDS_WITH]->(u2)
```

🔍 1. Basic Queries

1.1. Find All Users

```cypher
MATCH (u:User) RETURN u.name, u.id
```

1.2. Find a User by ID

```cypher
MATCH (u:User {id: 0}) RETURN u.name
```

1.3. Count All Users

```cypher
MATCH (u:User) RETURN COUNT(u) AS total_users
```

🤝 2. Friendship Queries

2.1. Find All Friends of a User

```cypher
MATCH (u:User {name: "Hero"})-[:FRIENDS_WITH]->(friend)
RETURN friend.name
```

Output: Dunn, Sue

2.2. Find Mutual Friends

```cypher
MATCH (a:User {name: "Dunn"})-[:FRIENDS_WITH]->(mutual)<-[:FRIENDS_WITH]-(b:User {name: "Su
e"})
RETURN mutual.name
```

Output: Hero, Chi

2.3. Find Friends-of-Friends (2nd Degree Connections)

```cypher
MATCH (u:User {name: "Hero"})-[:FRIENDS_WITH*2]->(fof)
```

```
RETURN DISTINCT fof.name
```

Output: Chi, Thor

🛠️ 3. Advanced Traversal

3.1. Shortest Path Between Two Users

cypher Copy

```cypher
MATCH (a:User {name: "Hero"}), (b:User {name: "Klein"})
MATCH path = shortestPath((a)-[:FRIENDS_WITH*]-(b))
RETURN [node IN nodes(path) | node.name] AS path
```

Output: ["Hero", "Dunn", "Chi", "Thor", "Clive", "Hicks", "Kate", "Klein"]

3.2. Detect Isolated Users (No Friends)

cypher Copy

```cypher
MATCH (u:User)
WHERE NOT (u)-[:FRIENDS_WITH]-()
RETURN u.name
```

📊 4. Graph Analytics

4.1. Most Popular Users (Degree Centrality)

cypher Copy

```cypher
MATCH (u:User)-[:FRIENDS_WITH]->(friend)
RETURN u.name, COUNT(friend) AS friend_count
ORDER BY friend_count DESC
```

Output:

User	Friends
Dunn	3
Chi	2

4.2. Friend Recommendations (Common Friends)

```cypher
MATCH (u:User {name: "Hero"})-[:FRIENDS_WITH]->(friend)-[:FRIENDS_WITH]->(recommendation)
WHERE NOT (u)-[:FRIENDS_WITH]->(recommendation) AND u <> recommendation
RETURN recommendation.name, COUNT(friend) AS common_friends
ORDER BY common_friends DESC
```

Output:

Recommendation	Common Friends
Chi	2

⚙️ 5. Data Modifications

5.1. Add a New User

```cypher
CREATE (:User {id: 10, name: "Alex"})
```

5.2. Add a New Friendship

```cypher
MATCH (a:User {name: "Hero"}), (b:User {name: "Alex"})
CREATE (a)-[:FRIENDS_WITH]->(b)
```

5.3. Remove a Friendship

```cypher
MATCH (a:User {name: "Hero"})-[r:FRIENDS_WITH]->(b:User {name: "Dunn"})
DELETE r
```

🚀 6. Pro Tips

6.1. Indexing for Faster Queries

```cypher
CREATE INDEX FOR (u:User) ON (u.id, u.name)
```

6.2. Batch Import from CSV

```cypher
LOAD CSV WITH HEADERS FROM "file:///users.csv" AS row
CREATE (:User {id: toInteger(row.id), name: row.name})
```

6.3. Visualize in Neo4j Browser

```cypher
MATCH path = (u:User)-[:FRIENDS_WITH]->()
RETURN path
```

(Run this in Neo4j Browser for a force-directed graph visualization.)

📚 Resources

- Neo4j Cypher Refcard
- Graph Academy Free Courses