See the slides on Neo4j.

Let’s use the movie database included in Neo4j. Our goal is to to show recommendations for other actors to work with or similar movies to watch. By following the meaningful relationships between actors and movies, we can determine occurrences of actors working together, the frequency of actors working with one another, and the movies they have in common in the graph. This structure forms the basis for many recommendation engines.

Let's start the Movies sandbox (https://sandbox.neo4j.com/) and let's type the following commands into the Neo4j Browser command line.

```
call db.schema.visualization()
```

```
MATCH (tom:Person {name: 'Tom Hanks'})
RETURN tom
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[r:ACTED_IN]->(movie:Movie)
RETURN tom, r, movie
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(:Movie)<-
[:ACTED_IN]-(coActor:Person)
RETURN coActor.name
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(:Movie)<-
[:ACTED_IN]-(coActor:Person)
RETURN distinct coActor.name
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(movie1:Movie)<-
[:ACTED_IN]-(coActor:Person)-[:ACTED_IN]->(movie2:Movie)<-
[:ACTED_IN]-(coCoActor:Person)
RETURN coCoActor.name
```

What is wrong with the previous query? Let's improve it as follows:

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(movie1:Movie)<-
[:ACTED_IN]-(coActor:Person)-[:ACTED_IN]->(movie2:Movie)<-
[:ACTED_IN]-(coCoActor:Person)
WHERE tom <> coCoActor AND NOT (tom)-[:ACTED_IN]->(:Movie)<-
[:ACTED_IN]-(coCoActor)
RETURN coCoActor.name
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(movie1:Movie)<-
[:ACTED_IN]-(coActor:Person)-[:ACTED_IN]->(movie2:Movie)<-
[:ACTED_IN]-(coCoActor:Person)
WHERE tom <> coCoActor AND NOT (tom)-[:ACTED_IN]->(:Movie)<-
[:ACTED_IN]-(coCoActor)
RETURN coCoActor.name, count(coCoActor) as frequency
ORDER BY frequency DESC
LIMIT 5
```

```
MATCH (tom:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(movie1:Movie)<-
[:ACTED_IN]-(coActor:Person)-[:ACTED_IN]->(movie2:Movie)<-
[:ACTED_IN]-(cruise:Person {name: 'Tom Cruise'})
WHERE NOT (tom)-[:ACTED_IN]->(:Movie)<-[:ACTED_IN]-(cruise)
RETURN tom, movie1, coActor, movie2, cruise
```