**Match Practice:**

Find the actor named "Tom Hanks"...

MATCH (tom {name: "Tom Hanks"}) RETURN tom

Find the movie with title "Cloud Atlas"...

MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas

Find 10 people...

MATCH (people:Person) RETURN people.name LIMIT 10

Find movies released in the 1990s...

MATCH (nineties:Movie) WHERE nineties.released >= 1990 AND nineties.released < 2000 RETURN nineties.title

List all Tom Hanks movies...

MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED\_IN]->(tomHanksMovies) RETURN tom,tomHanksMovies

Who directed "Cloud Atlas"?

MATCH (cloudAtlas {title: "Cloud Atlas"}) <-[:DIRECTED]-(directors) RETURN directors.name

Tom Hanks' co-actors...

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors) RETURN coActors.name

Movies and actors up to 4 "hops" away from Kevin Bacon

MATCH (bacon:Person {name:"Kevin Bacon"})-[\*1..4]-(hollywood)

RETURN DISTINCT Hollywood

Bacon path, the shortest path of any relationships to Meg Ryan

MATCH p=shortestPath(

(bacon:Person {name:"Kevin Bacon"})-[\*]-(meg:Person {name:"Meg Ryan"})

)

RETURN p

Note you only need to compare property values like this when first creating relationships

Extend Tom Hanks co-actors, to find co-co-actors who haven't worked with Tom Hanks...

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors),

(coActors)-[:ACTED\_IN]->(m2)<-[:ACTED\_IN]-(cocoActors)

WHERE NOT (tom)-[:ACTED\_IN]->()<-[:ACTED\_IN]-(cocoActors) AND tom <> cocoActors

RETURN cocoActors.name AS Recommended, count(\*) AS Strength ORDER BY Strength DESC

Find someone to introduce Tom Hanks to Tom Cruise

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED\_IN]->(m)<-[:ACTED\_IN]-(coActors),

(coActors)-[:ACTED\_IN]->(m2)<-[:ACTED\_IN]-(cruise:Person {name:"Tom Cruise"})

RETURN tom, m, coActors, m2, cruise