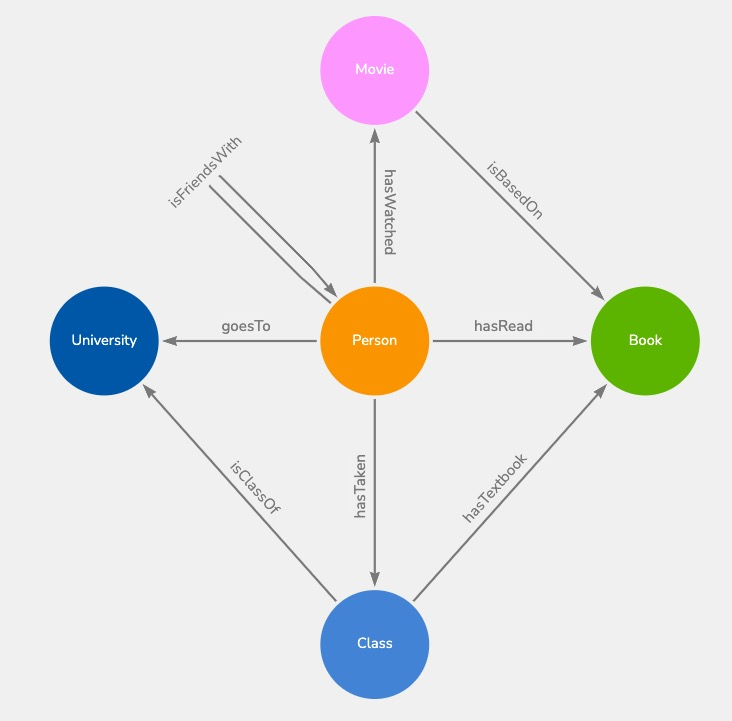
**Exploring Our Connections**

*The* ***purpose*** *of the dataset we have created and our analysis is to see how we all connect in a world where we lead separate lives*

**

Why use a graphical database?

* There are a lot of good reasons to use a graphical database, for example…
  + It represents the complex relationships in our dataset very well
    - Very easy to understand and visualize as well!
  + It allows for complex data exploration
    - We were able to look at complex data relationships throughout the process and fairly easily
  + It is great for pattern matching
    - We were able to infer patterns by its graphical nature and easy to understand displays
  + Scalability
    - If we wanted to expand out the project even more we would easily have that capability
  + Flexibility
    - Trends emerge as the data changed.
    - This gave us a better understanding of the data

Below we have explained our queries in detail

# CYPHER QUERIES

# checking entire network

MATCH (all) RETURN all;

# finding all people who have read books

MATCH (Person)-[r:hasRead]->(Book)

RETURN Person,r,Book

# finding all classes all Person entities have taken

MATCH (p:Person)-[:hasTaken]->(c:Class)

RETURN p.name AS Person, COLLECT(c.name) AS ClassesTaken

# finding the total number of books and movies read & watched by all Person entities

MATCH (p:Person)-[r:hasRead|hasWatched]->(m)

RETURN p.name AS Person, COUNT(m) AS Count

# finding books released after 2010 or movies released in or after 2020

MATCH (b:Book)

MATCH (m:Movie)

WHERE b.year > 2010 OR m.year >= 2020

RETURN b, m

# finding classes that Aayushi has taken but not Kelsey

MATCH (a:Person {name: 'Aayushi'})-[:hasTaken]->(c:Class)

WHERE NOT EXISTS {

MATCH (k:Person {name: 'Kelsey'})-[:hasTaken]->(c)

}

RETURN c.name AS Class

# now the other way around

MATCH (k:Person {name: 'Kelsey'})-[:hasTaken]->(c:Class)

WHERE NOT EXISTS {

MATCH (a:Person {name: 'Aayushi'})-[:hasTaken]->(c)

}

RETURN c.name AS Class

# finding classes that students have taken together

MATCH (a:Person)-[:isFriendsWith]->(b:Person)

MATCH (a)-[:hasTaken]->(c:Class)<-[:hasTaken]-(b)

RETURN a.name AS Person1, b.name AS Person2, COLLECT(DISTINCT c.name) AS CommonClasses

# finding total number of students for each university

MATCH (p:Person)-[:goesTo]->(university:University)

RETURN university.name AS University, university.state AS State, COUNT(p) AS NumberOfStudents

ORDER BY NumberOfStudents DESC

# finding all movies with a wildcard regex search

MATCH (m:Movie)

WHERE m.title =~ 'Harry.\*'

RETURN m.title

# finding all Books which have been used as class textbooks

MATCH (c:Class)-[:hasTextbook]->(b:Book)

WITH b, COUNT(c) AS ClassCount

RETURN b.title AS Book, ClassCount

ORDER BY ClassCount DESC

# finding Person entities who have watched a movie based on a book

MATCH (p:Person)-[:hasWatched]->(m:Movie)-[:isBasedOn]->(b:Book)<-[:hasRead]-(p)

RETURN p,b,m

