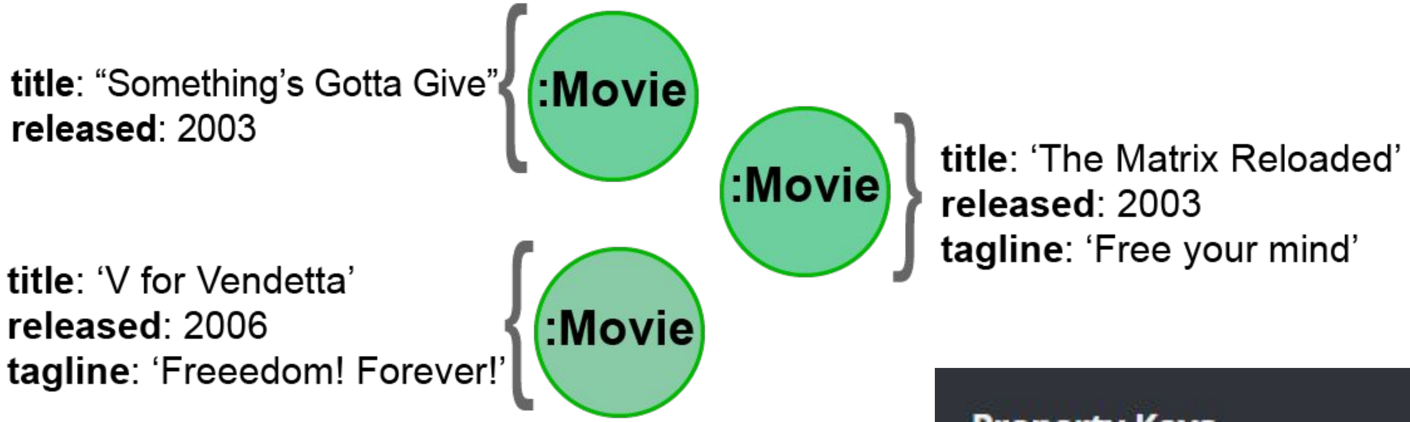
Introduction to Cypher

Dr. Sirasit Lochanachit

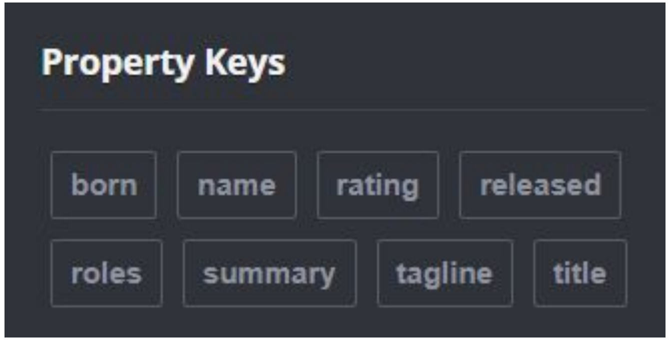
Today's Outline

- Filtering Queries
  - Using Nodes, Property Values, and Relationships
  - Using the WHERE clause to filter queries

Properties


title: "Something's Gotta Give"
released: 2003
:Movie
:Movie
title: 'The Matrix Reloaded'
released: 2003
tagline: 'Free your mind'
title: 'V for Vendetta'
released: 2006
tagline: 'Freeedom! Forever!'
:Movie

CALL db.propertyKeys()


Property Keys

born name rating released
roles summary tagline title

Ref: https://neo4j.com/graphacademy/training-querying-40/01-querying40-introduction-to-cypher/

MATCH and RETURN

```
MATCH (variable:Label {propertyKey: propertyValue, propertyKey2: propertyValue2})
RETURN variable
```

Example

- Retrieve all *Person* nodes that have a *born* property value of *1970*.

```
MATCH (p:Person {born: 1970})
RETURN p
```

p is a variable

:Person is a node label

Returning Property Values

```
MATCH (variable:Label {prop1: value, prop2: value})
RETURN variable.prop3, variable.prop4
```

Example

- Retrieve all *name* and *born* values that have a *born* property value of *1970*.

```
MATCH (p:Person {born: 1970})
RETURN p.name, p.born
```

p is a variable

:Person is a node label

Returning Property Values with alias

```
MATCH (variable:Label {prop1: value, prop2: value})
RETURN variable.prop3 AS alias3
```

Example

- Retrieve all *name* and *born* values that have a *born* property value of *1970.*

```
MATCH (p:Person {born: 1970})
RETURN p.name AS name, p.born AS `birth year`
```

Exercise 2: Filtering Queries using Property Values

1) Write a query to retrieve all *Movie* nodes that *released* in 2019.
2) Write a query to retrieve all Movie nodes that were released in 2020.
3) Write a query to retrieve all *Movie* released in 2003, returning their titles.
4) Write a query to retrieve all *Movie* nodes and display the *title*, *released*, and *tagline* values.
5) Modify 4) query to rename the columns as 'movie title', 'released year', and 'tagLine'

Relationship Syntax

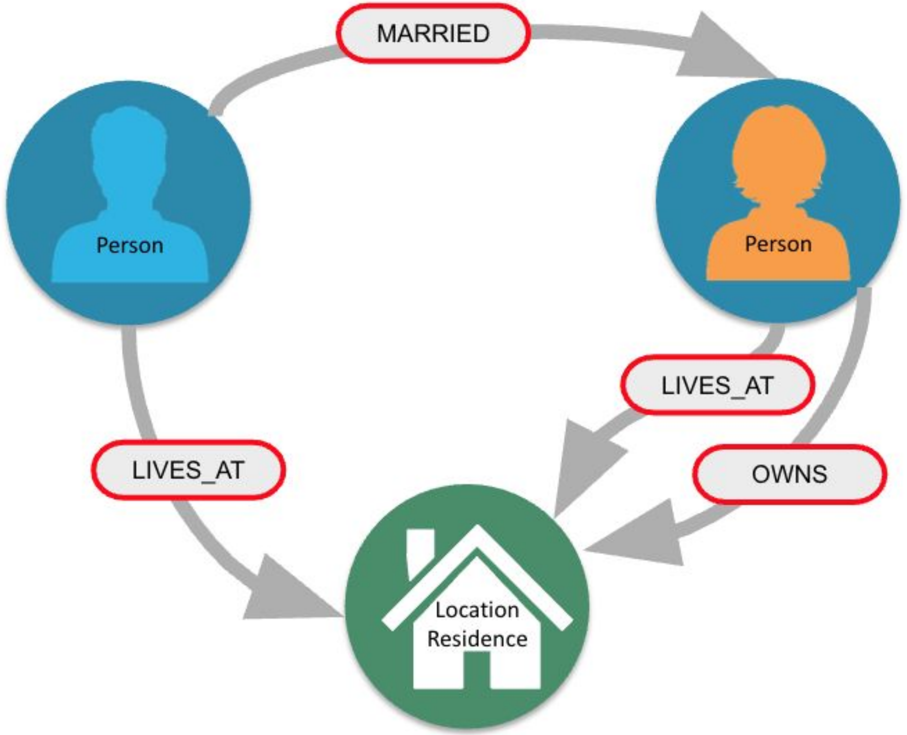
```
()

() -- ()

() - [] - ()

() --> ()

() <-- ()
```



Ref: https://neo4j.com/graphacademy/training-querying-40/01-querying40-introduction-to-cypher/

MATCH and RETURN using relationships

```
MATCH (node1) -[:REL_TYPE_A | REL_TYPE_B]-> (node2)
RETURN node1, node2
```

Example

- Retrieve all *Person* nodes that have acted in *The Matrix*.

```
MATCH (p:Person) -[rel:ACTED_IN]-> (m:Movie {title: 'The Matrix'})
RETURN p, rel, m
```

p, rel, m is a variable

:Person, :Movie is a node label

:ACTED_IN is a relationship label

Using an Anonymous Relationship for a Query

- Retrieve all *Person* nodes that have any relationship in *The Matrix*.

```
MATCH (p:Person) --> (m:Movie {title: 'The Matrix'})
RETURN p, m

MATCH (p:Person) -- (m:Movie {title: 'The Matrix'})
RETURN p, m

MATCH (p:Person) -[]- (m:Movie {title: 'The Matrix'})
RETURN p, m
```

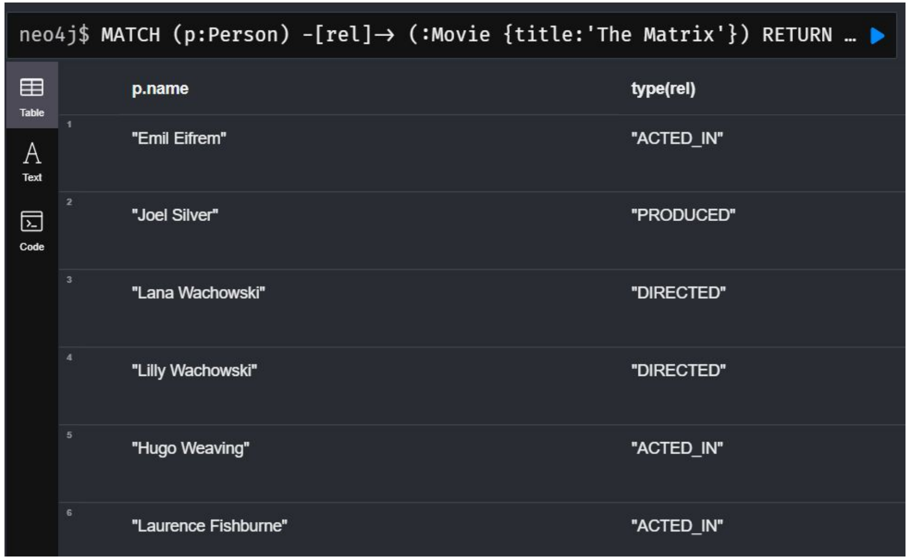
- Retrieve all *Movie* nodes that have any relationship with *Keanu Reeves*.

```
MATCH (m:Movie) <-- (p:Person {name: 'Keanu Reeves'})
RETURN p, m
```

Retrieving the relationship types

- Retrieve all *Person* nodes that have any relationship in *The Matrix*.

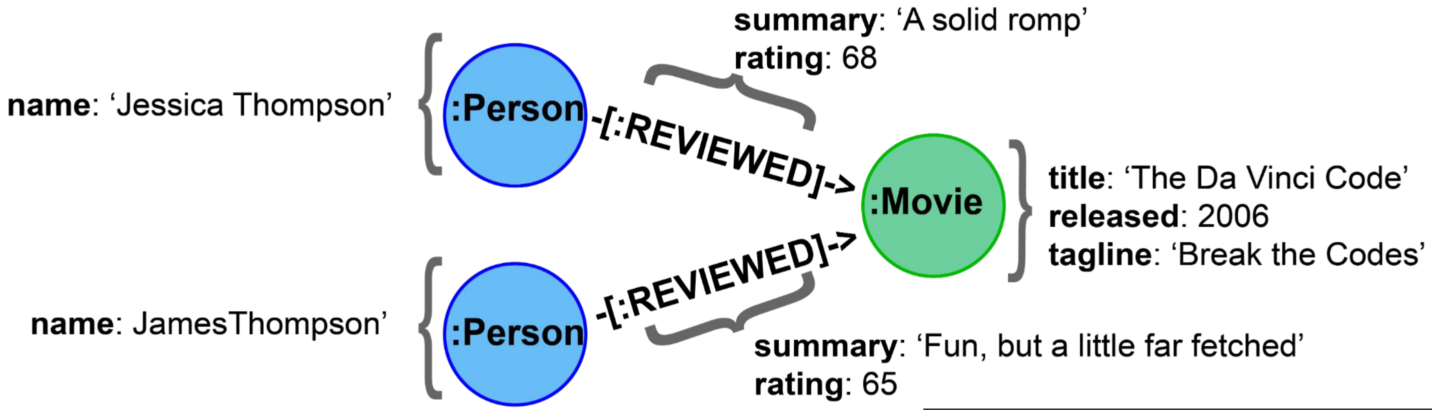
```
MATCH (p:Person) -[rel]-> (:Movie {title:'The Matrix'})
RETURN p.name, type(rel)
```



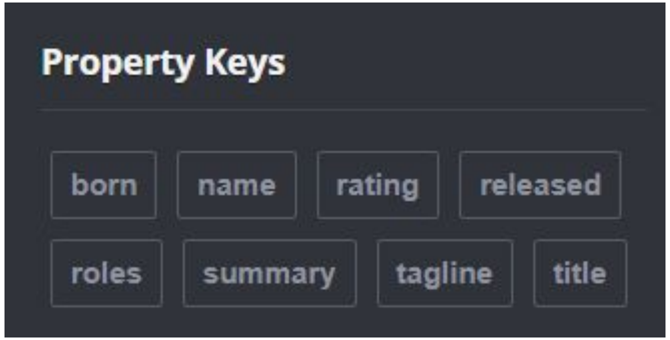
```
neo4j$ MATCH (p:Person) -[rel]→ (:Movie {title:'The Matrix'}) RETURN …
```

	p.name	type(rel)
1	"Emil Eifrem"	"ACTED_IN"
2	"Joel Silver"	"PRODUCED"
3	"Lana Wachowski"	"DIRECTED"
4	"Lilly Wachowski"	"DIRECTED"
5	"Hugo Weaving"	"ACTED_IN"
6	"Laurence Fishburne"	"ACTED_IN"

Properties for relationships



CALL db.propertyKeys()


Property Keys

born name rating released
roles summary tagline title

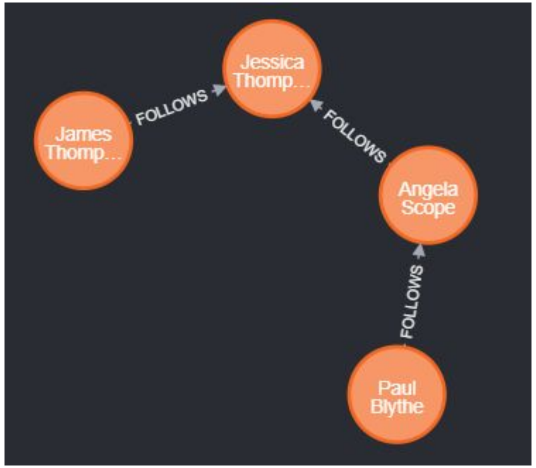
Ref: https://neo4j.com/graphacademy/training-querying-40/01-querying40-introduction-to-cypher/

MATCH and RETURN using relationship properties

- Retrieve the name of the person who gave *The Da Vinci Code* movie a rating of *65*.

```
    MATCH (p:Person) -[:REVIEWED {rating: 65}]-> (:Movie {title: 'The Da Vinci Code'})
    RETURN p.name
```

Patterns in the graph



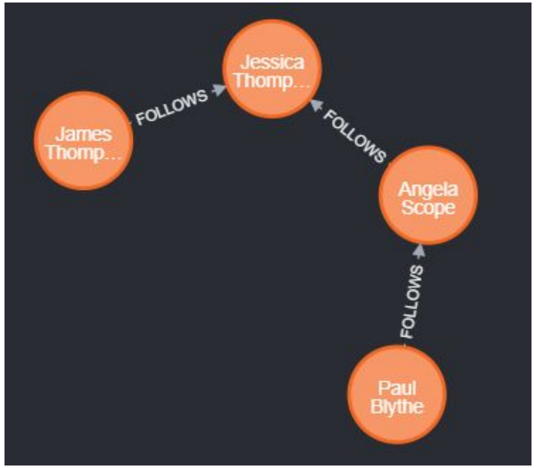
- Retrieve all *Person* nodes who follow *Angela Scope*.

```
MATCH  (p:Person) -[:FOLLOWS]-> (:Person {name:'Angela Scope'})
RETURN p
```

- Retrieve all *Person* nodes who is followed by *Angela Scope*.

```
MATCH  (p:Person) <-[:FOLLOWS]- (:Person {name:'Angela Scope'})
RETURN p
```

Patterns in the graph



- Retrieve all *Person* nodes who follow or is followed by *Angela Scope*.

```
MATCH  (p1:Person) -[:FOLLOWS]- (p2:Person {name:'Angela Scope'})
RETURN p1, p2
```

Traversing Multiple Relationships

- Return all followers of the followers of *Jessica Thompson*.

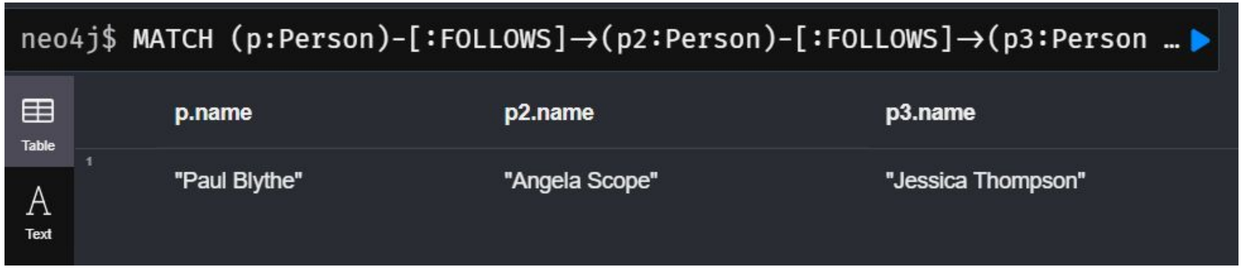
MATCH (p:Person) -[:FOLLOWS]-> (:Person) -[:FOLLOWS]-> (:Person {name:'Jessica Thompson'})

RETURN p

- Return each person name

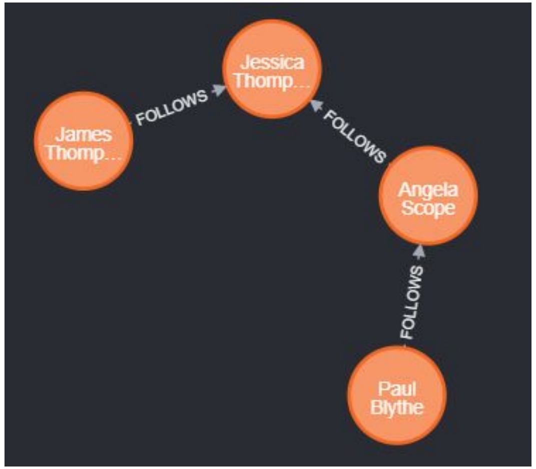
MATCH (p:Person) -[:FOLLOWS]-> (p2:Person) -[:FOLLOWS]-> (p3:Person {name:'Jessica Thompson'})

RETURN p.name, p2.name, p3.name



```
neo4j$ MATCH (p:Person)-[:FOLLOWS]→(p2:Person)-[:FOLLOWS]→(p3:Person …
```

	p.name	p2.name	p3.name
1	"Paul Blythe"	"Angela Scope"	"Jessica Thompson"



Exercise 3: Filtering Queries using Relationships

1) Write a query to retrieve all *Person* names who wrote the movie *Top Gun*.
2) Write a query to retrieve all movie titles connected with *Tom Hanks*.

   Hint: Tom Hanks has multiple relationships with a movie (Actor and Director)

3) Modify 2) query to return the information <u>as a table</u> about the type of relationships between Tom Hanks and the movies.
4) Retrieve information about the <u>movies and roles</u> that Tom Hanks acted in.

Using WHERE to Filter Queries

- Retrieve all *Person* nodes that have acted in 2008 movies.

```
MATCH (p:Person) -[:ACTED_IN]-> (m:Movie {released: 2008})
RETURN p, m
```

```
MATCH (p:Person) -[:ACTED_IN]-> (m:Movie)
WHERE m.released = 2008
RETURN p, m
```

- Boolean operators: AND, OR, XOR, and NOT
- Comparison Operators: =, <, >, <=, >=, <>, IS NULL, IS NOT NULL, STARTS WITH, ENDS WITH, CONTAINS
  - Etc.

Specifying a Range

```
MATCH (p:Person) -[:ACTED_IN]-> (m:Movie)
WHERE m.released >= 2003 AND m.released <= 2004
RETURN p.name, m.title, m.released
```

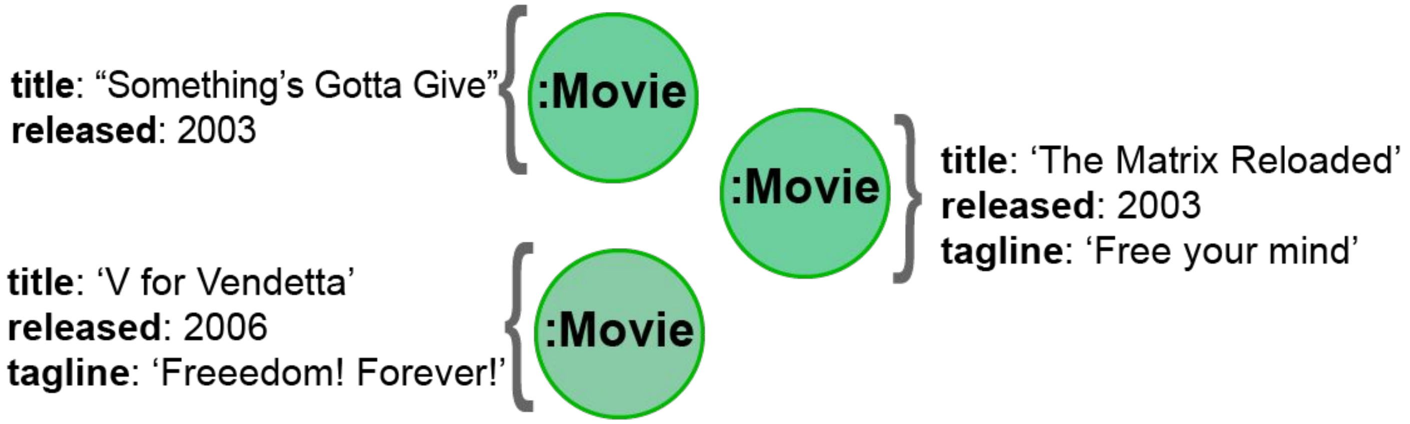
```
MATCH (p:Person) -[:ACTED_IN]-> (m:Movie)
WHERE 2003 <= m.released <= 2004
RETURN p.name, m.title, m.released
```

Specifying Labels

```
MATCH (p:Person) -[:ACTED_IN]-> (:Movie {title: 'The Matrix'})
RETURN p.name
```

```
MATCH (p) -[:ACTED_IN]-> (m)
WHERE p:Person AND m:Movie AND m.title='The Matrix'
RETURN p.name
```

Testing Existence of a property


title: "Something's Gotta Give"
released: 2003
:Movie
:Movie
title: 'The Matrix Reloaded'
released: 2003
tagline: 'Free your mind'
title: 'V for Vendetta'
released: 2006
tagline: 'Freeedom! Forever!'
:Movie

```
MATCH (p:Person) -[:ACTED_IN]-> (m:Movie)

WHERE p.name='Jack Nicholson' AND m.tagline IS NOT NULL

RETURN m.title, m.tagline
```

Ref: https://neo4j.com/graphacademy/training-querying-40/01-querying40-introduction-to-cypher/

Testing Strings

- STARTS WITH, ENDS WITH, and CONTAINS.
  - String comparisons are case-sensitive.

```
MATCH (p:Person) -[:ACTED_IN]-> ()
WHERE p.name STARTS WITH 'Michael'
RETURN p.name
```

Testing with Patterns

- Return all *Person* nodes of people who wrote movies.

```
MATCH (p:Person) -[:WROTE]-> (m:Movie)
RETURN p.name, m.title
```

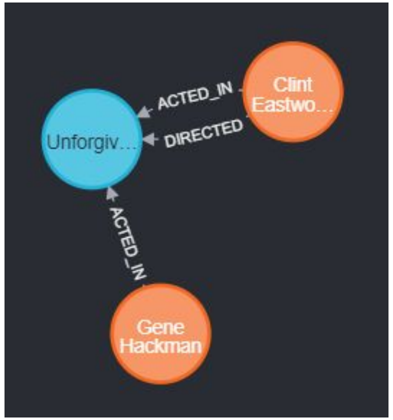
- Exclude people who directed that particular movie.

```
MATCH (p:Person) -[:WROTE]-> (m:Movie)
WHERE NOT( (p) -[:DIRECTED]-> (m) )
RETURN p.name, m.title
```

Testing with Patterns

- Find *Gene Hackman* and the movies that he acted in with another person who also directed the movie.

```
MATCH (gene:Person) -[:ACTED_IN]-> (m:Movie) <-[:ACTED_IN]- (other:Person)
WHERE gene.name= 'Gene Hackman' AND ( (other) -[:DIRECTED]-> (m) )
RETURN  gene, other, m
```



Testing with List Values

- Return all *Person* nodes of people born in 1965 or 1970.

```
MATCH (p:Person)
WHERE p.born IN [1965, 1970]
RETURN p.name as name, p.born as yearBorn
```

- Return the name of the actor who played *Neo* in the movie *The Matrix*.

```
MATCH (p:Person) -[r:ACTED_IN]-> (m:Movie)
WHERE  'Neo' IN r.roles AND m.title= 'The Matrix'
RETURN p.name
```

Exercise 4: Filtering Queries using WHERE clause

* Every results must have alias name (e.g. column names)

1) ✓ Retrieve all movies that Tom Cruise acted in and return their titles (using WHERE clause).
2) ✓ Retrieve all people that were born in the 70's and return their names and year born.
3) ✓ Retrieve the actors who acted in the movie *The Matrix* who were born after 1960, and return their names and year born (using WHERE clause only, specifying a condition directly in a MATCH clause is not allowed).
4) ✓ Retrieve all people in the graph that do not have a born property, returning their names.

Exercise 4: Filtering Queries using WHERE clause

5) Retrieve all people related to movies where the relationship has the *rating* property, then return their name, movie title, and the rating.

6) Retrieve all REVIEWED relationships from the graph where the summary of the review contains the string '*fun*', returning the movie title reviewed and the rating and summary of the relationship.

7) Retrieve the movies and their actors where one of the actors also directed the movie, returning the actors names, the director's name, and the movie title. The results should exclude a record that has the same actor's name and director's name.

8) Retrieve the movies that have an actor's role that is the name of the movie, return the movie title and the actor's name.