Neo4j II - Relationships

HIGHER DIPLOMA IN DATA ANALYTICS
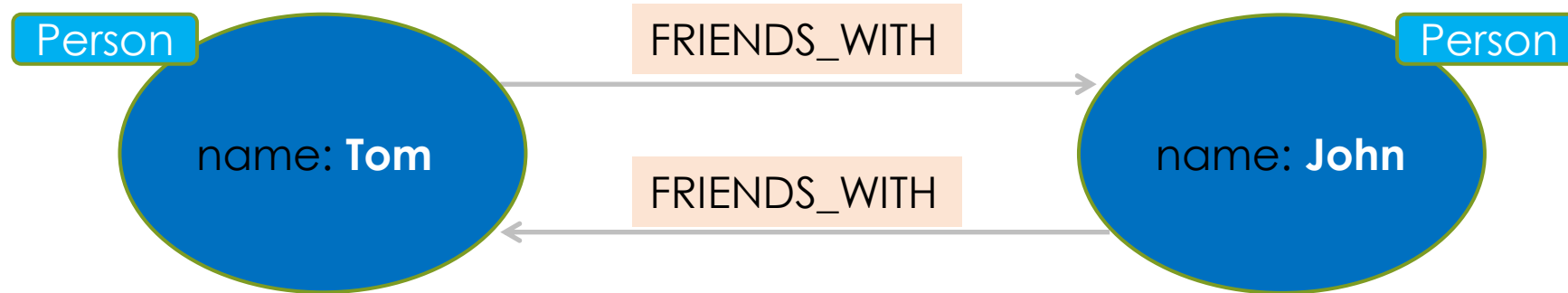
Relationships

▶ A *relationship* is an entity that encodes a directed connection between exactly two nodes, the *source node* and the *target node*.

▶ An *outgoing* relationship is a directed relationship from the point of view of its source node.

▶ An *incoming* relationship is a directed relationship from the point of view of its target node.

▶ A relationship can have properties but is assigned exactly one relationship type.
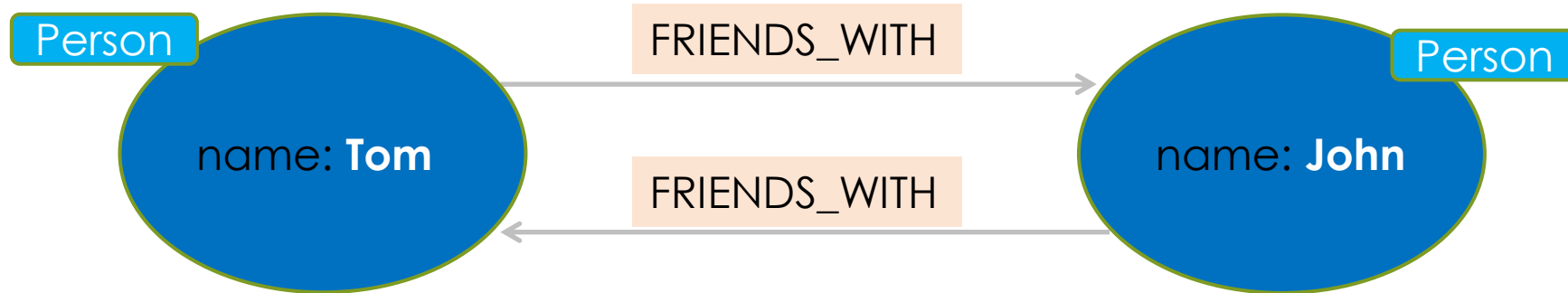
Relationships

Relationships

- ()
- (:Person)
- (:Person{name:"Tom"})

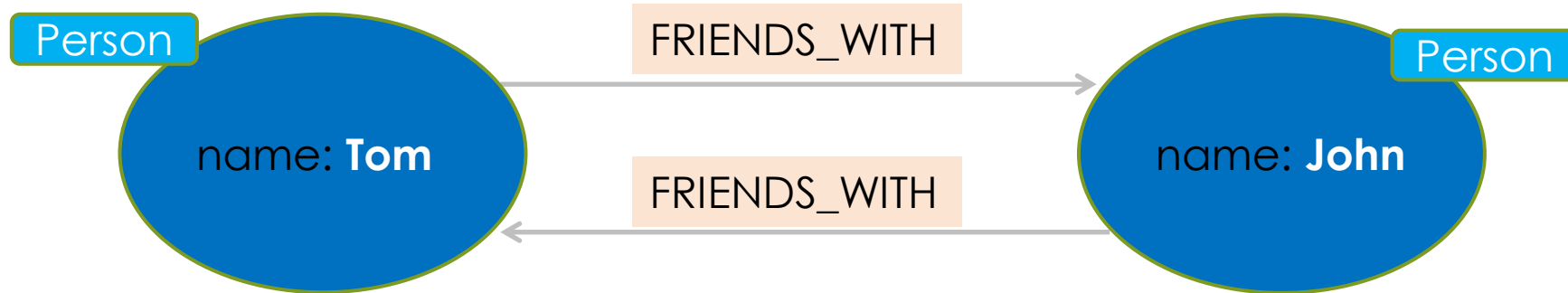- -[:FRIENDS_WITH]->
- <-[:FRIENDS_WITH]-

Creating a Relationship

```
CREATE(p1:Person{name:"Tom"})-[r:FRIENDS_WITH]->(p2:Person{name:"John"})

RETURN p1, r, p2
```

Person

FRIENDS_WITH

Person

name: **Tom**

FRIENDS_WITH

name: **John**

```
MATCH(p1:Person{name:"John"}),(p2:Person{name:"Tom"})
CREATE(p1)-[r:FRIENDS_WITH]->(p2)
RETURN p1, r, p2
```
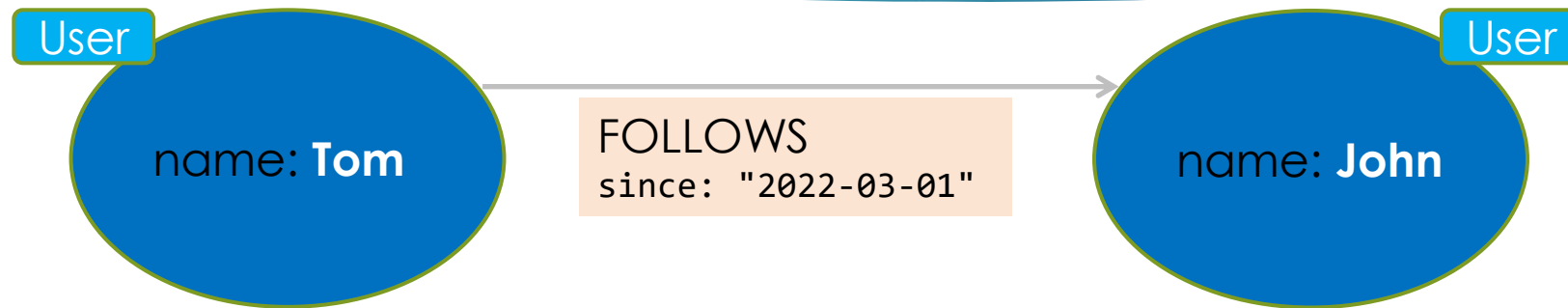
Creating a Relationship


```
MATCH(p1:Person{name:"John"}),(p2:Person{name:"Tom"})
MERGE(p1)-[r:FRIENDS_WITH]->(p2)
RETURN p1,r,p2
```

Creating a Relationship

User

name: **Tom**

FOLLOWS
since: "2022-03-01"

User

name: **John**
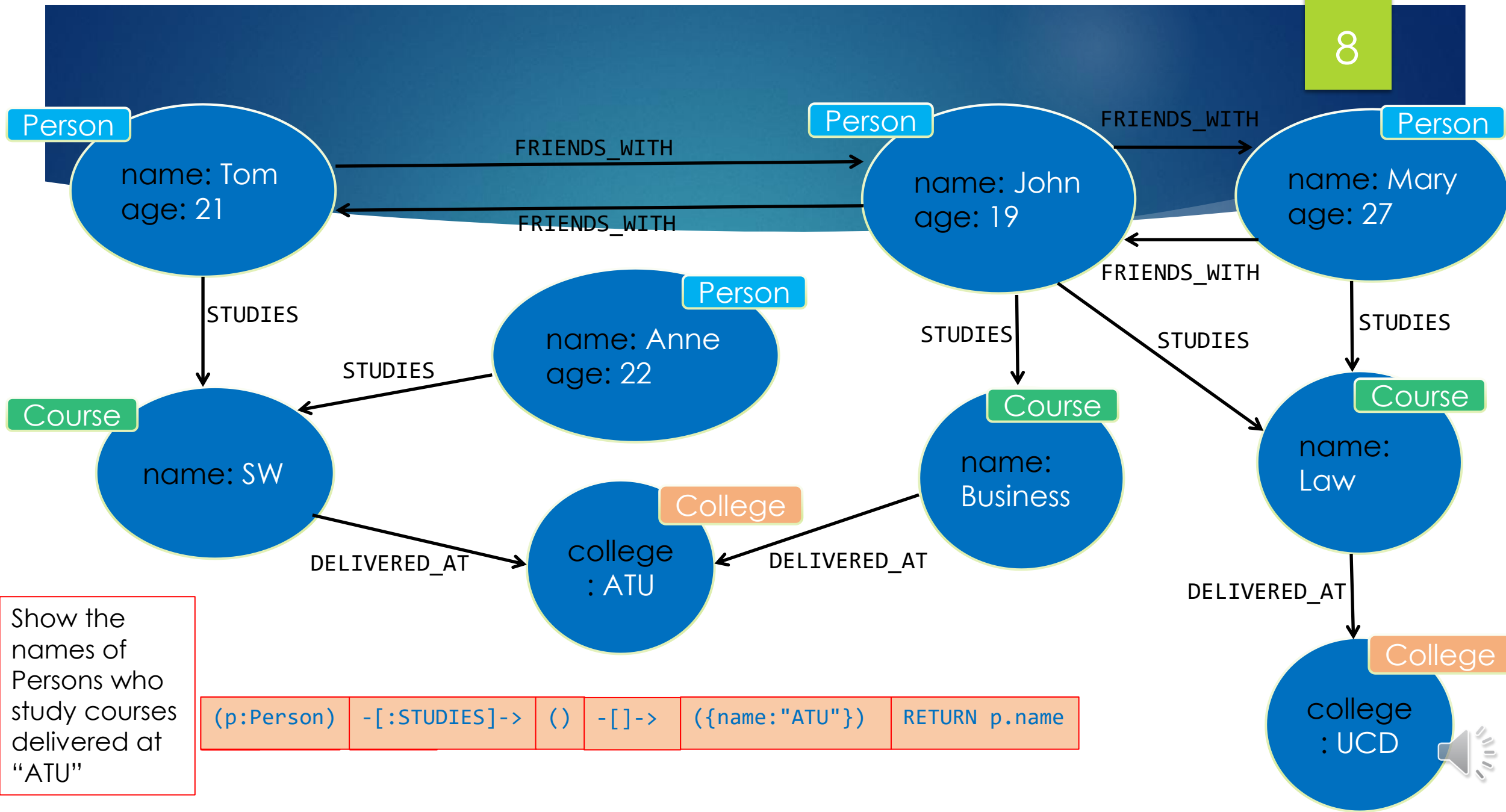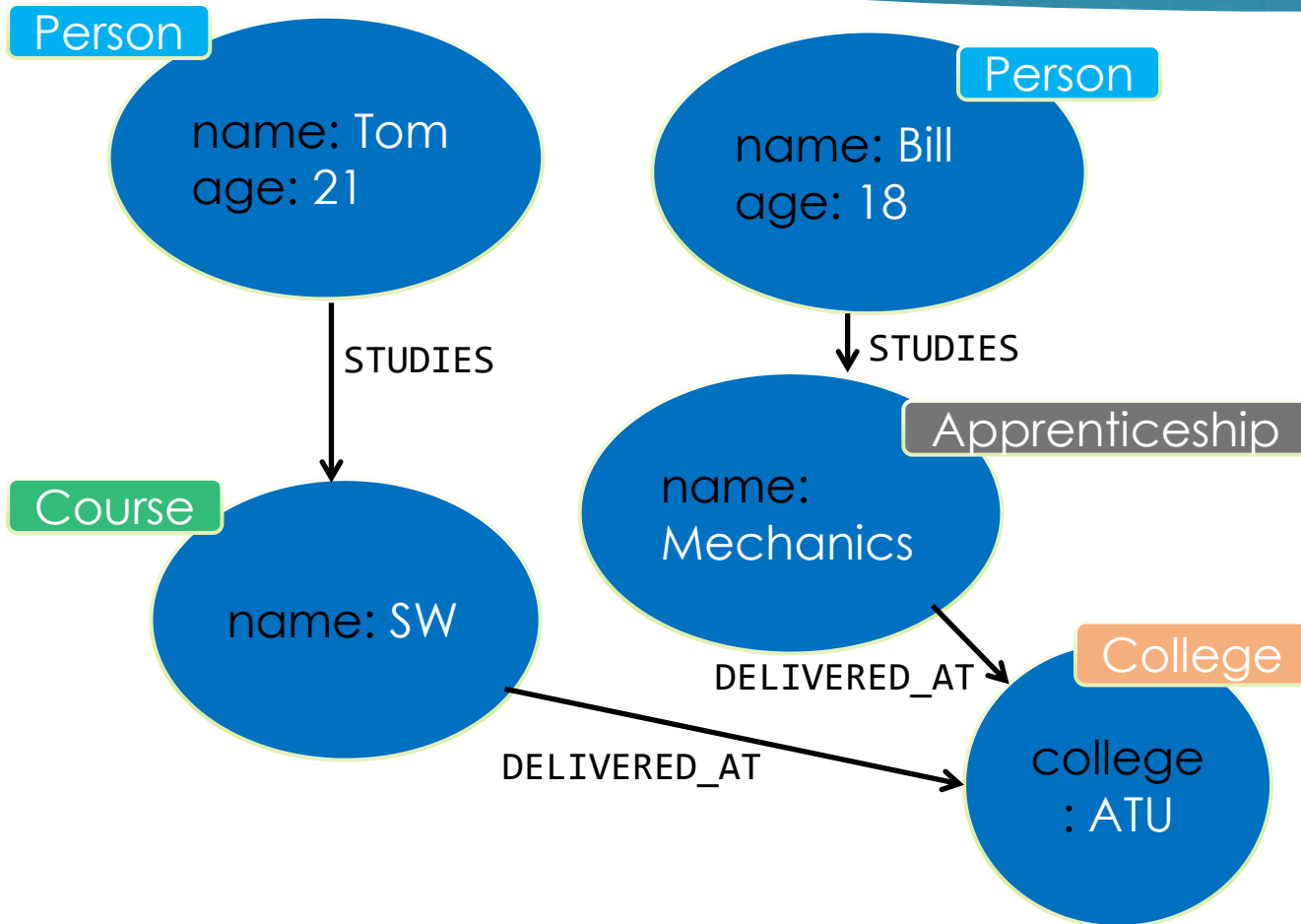
▶ Relationships can also have properties.

▶ Create a directional "FOLLOWS" Relationship between existing nodes "Tom" and "John", with a property key = *since* and a value = "*2022-03-01*".

```
MATCH(p1:Person{name:"Tom"}),(p2:Person{name:"John"})

MERGE(p1)-[r:FOLLOWS{since:"2022-03-01"}]->(p2)

RETURN p1, r, p2
```

Relationships

Person

name: Tom
age: 21

Person

name: Bill
age: 18

Course

name: SW

STUDIES

STUDIES

Apprenticeship

name:
Mechanics

DELIVERED_AT

DELIVERED_AT

College

college
: ATU

Show the names of Persons who study courses delivered at "ATU"
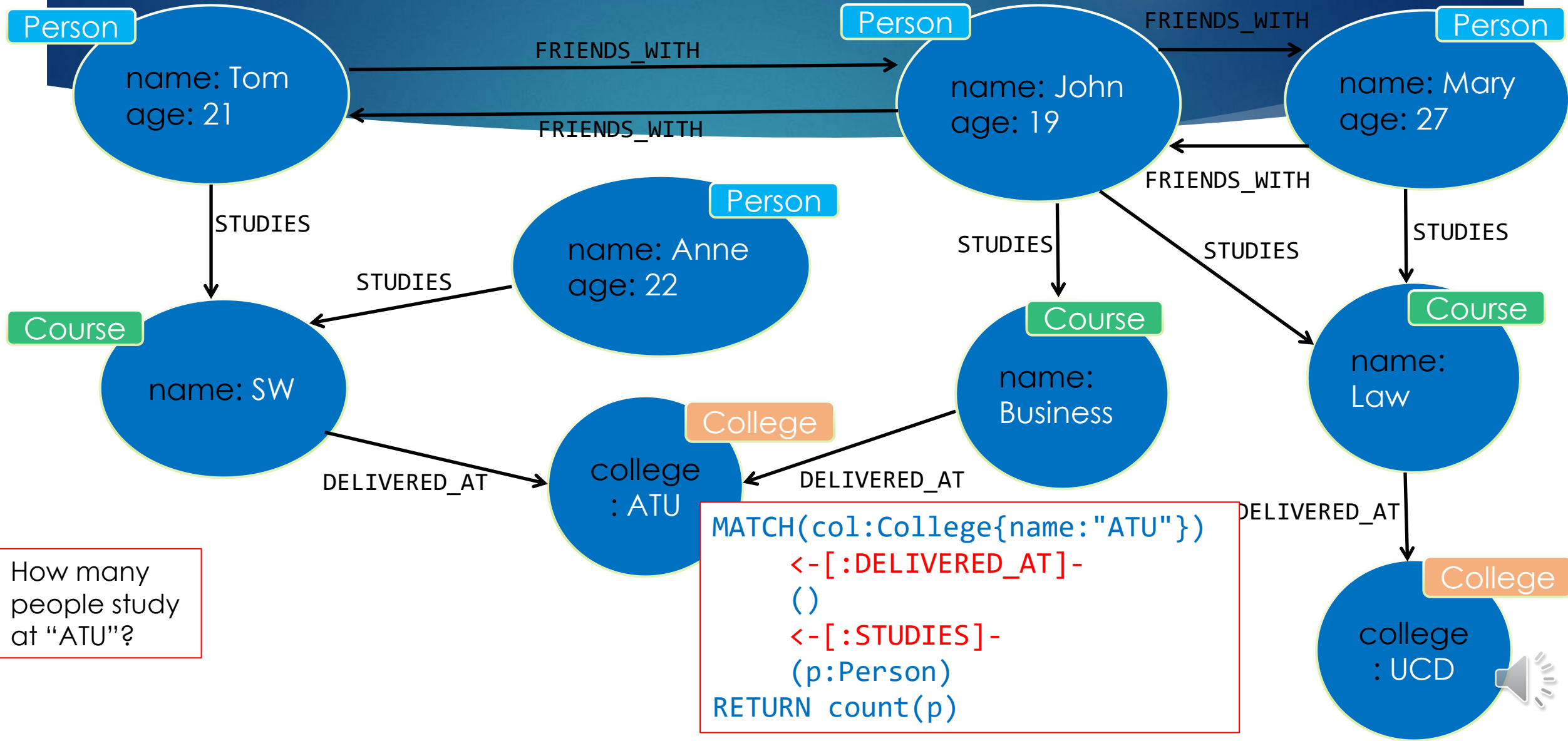
```
MATCH(p:Person)
    -[:STUDIES]->
    ()
    -->
    ({name:"ATU"})
RETURN p.name
```

```
MATCH(p:Person)
    -[:STUDIES]->
    (c:Course)
    -[d:DELIVERED_AT]->
    (col:College{name:"ATU"})
RETURN p.name
```

Person — name: Tom, age: 21

Person — name: John, age: 19

Person — name: Mary, age: 27

FRIENDS_WITH
FRIENDS_WITH
FRIENDS_WITH
FRIENDS_WITH

STUDIES

Person — name: Anne, age: 22

STUDIES

STUDIES
STUDIES
STUDIES

Course — name: SW

Course — name: Business

Course — name: Law

College — college: ATU

DELIVERED_AT
DELIVERED_AT
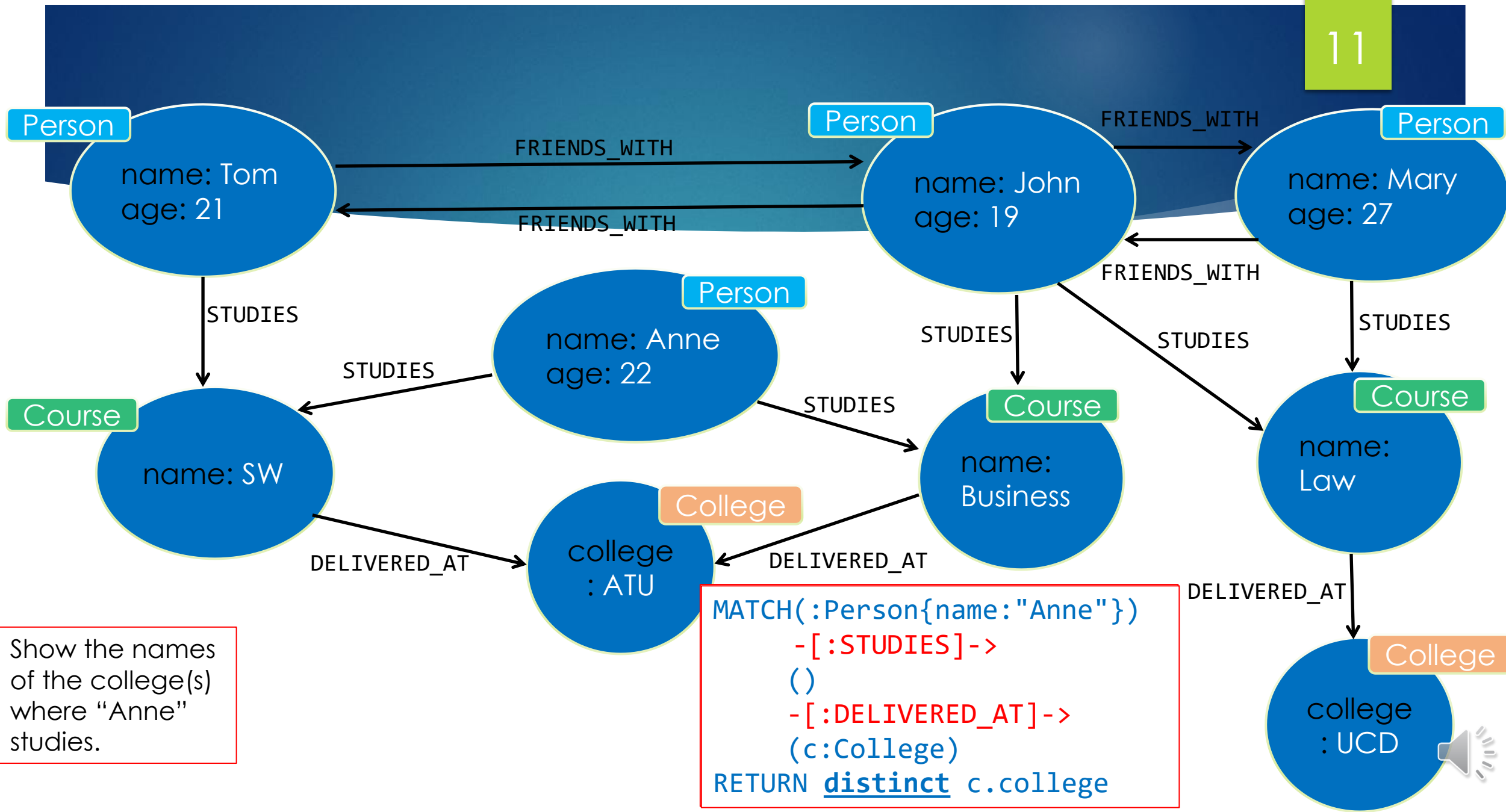DELIVERED_AT

College — college: UCD

How many people study at "ATU"?

```
MATCH(col:College{name:"ATU"})
     <-[:DELIVERED_AT]-
     ()
     <-[:STUDIES]-
     (p:Person)
RETURN count(p)
```

Show the names of the college(s) where "Anne" studies.

```
MATCH(:Person{name:"Anne"})
    -[:STUDIES]->
    ()
    -[:DELIVERED_AT]->
    (c:College)
RETURN distinct c.college
```

Show the Person's name and the name of the course(s) he/she studies.

```
MATCH(n:Person)
    -[:STUDIES]->
    (c:Course)
RETURN n.name as Name,
       c.name as Courses
```

"Name"	"Courses"
"Tom"	"SW"
"John"	"Law"
"John"	"Business"
"Anne"	"SW"
"Mary"	"Law"

```
MATCH(n:Person)
    -[:STUDIES]->
    (c:Course)
RETURN n.name as Name,
collect(c.name) as Courses
```
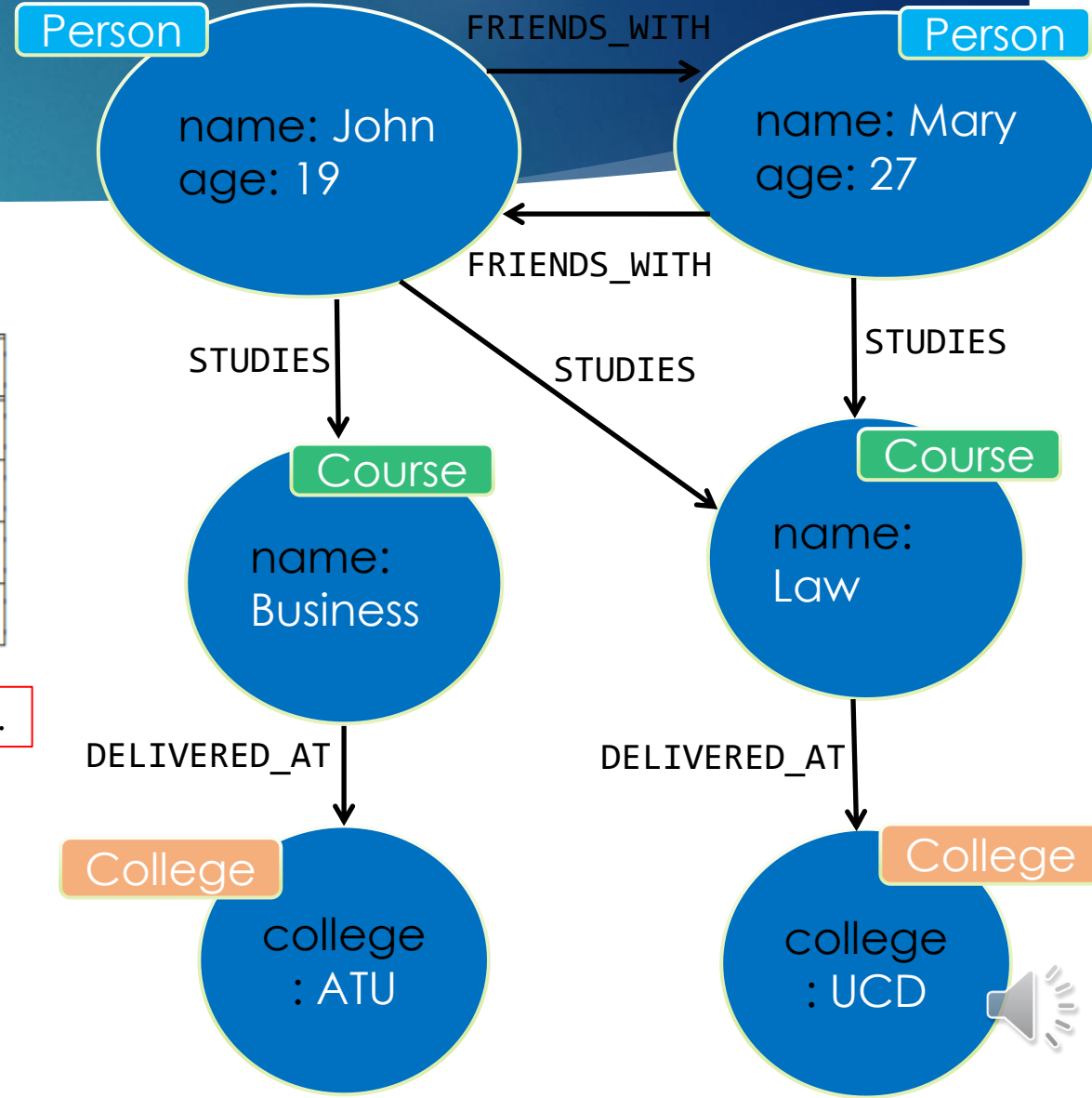
"Name"	"Courses"
"Tom"	["SW"]
"John"	["Law","Business"]
"Anne"	["SW"]
"Mary"	["Law"]

Show the Person's name and the number of the course(s) he/she studies.

```
MATCH(n:Person)
    -[:STUDIES]->
    (c:Course)
RETURN n.name as Name,
size(collect(c.name)) as Courses
```

"Name"	"Courses"
"Tom"	1
"John"	2
"Anne"	1
"Mary"	1

Person
name: John
age: 19

FRIENDS_WITH

Person
name: Mary
age: 27

FRIENDS_WITH

STUDIES

STUDIES

STUDIES

Course
name: Business

Course
name: Law

DELIVERED_AT

DELIVERED_AT

College
college : ATU

College
college : UCD

Variable Length Pattern Matching

- Rather than describing a long path using a sequence of many node and relationship descriptions in a pattern, many relationships (and the intermediate nodes) can be described by specifying a length in the relationship description of a pattern.
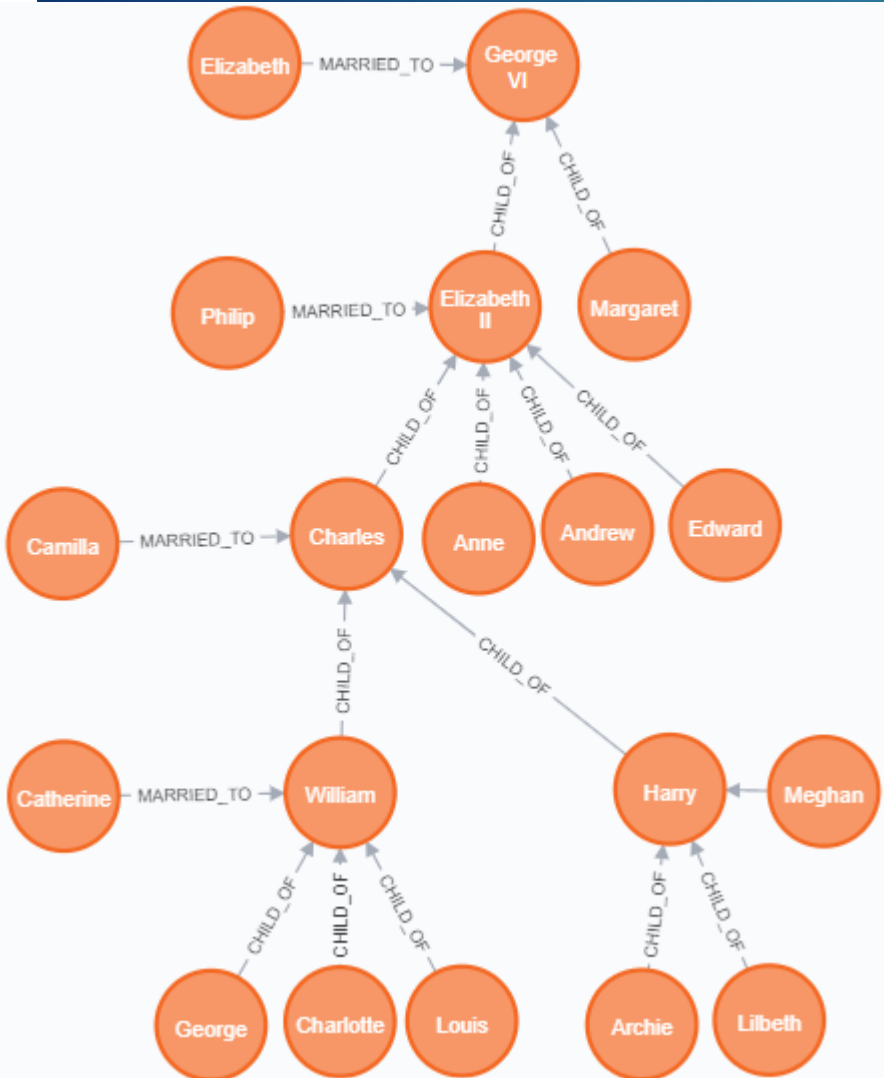
Show "George"s grandparent.

```
MATCH(g{name:"George"})
     -[:CHILD_OF]->
     ()
     -[:CHILD_OF]->
     (gp)
RETURN gp.name
```

```
MATCH(g{name:"George"})
     -[:CHILD_OF*2]->
     (gp)
RETURN gp.name
```

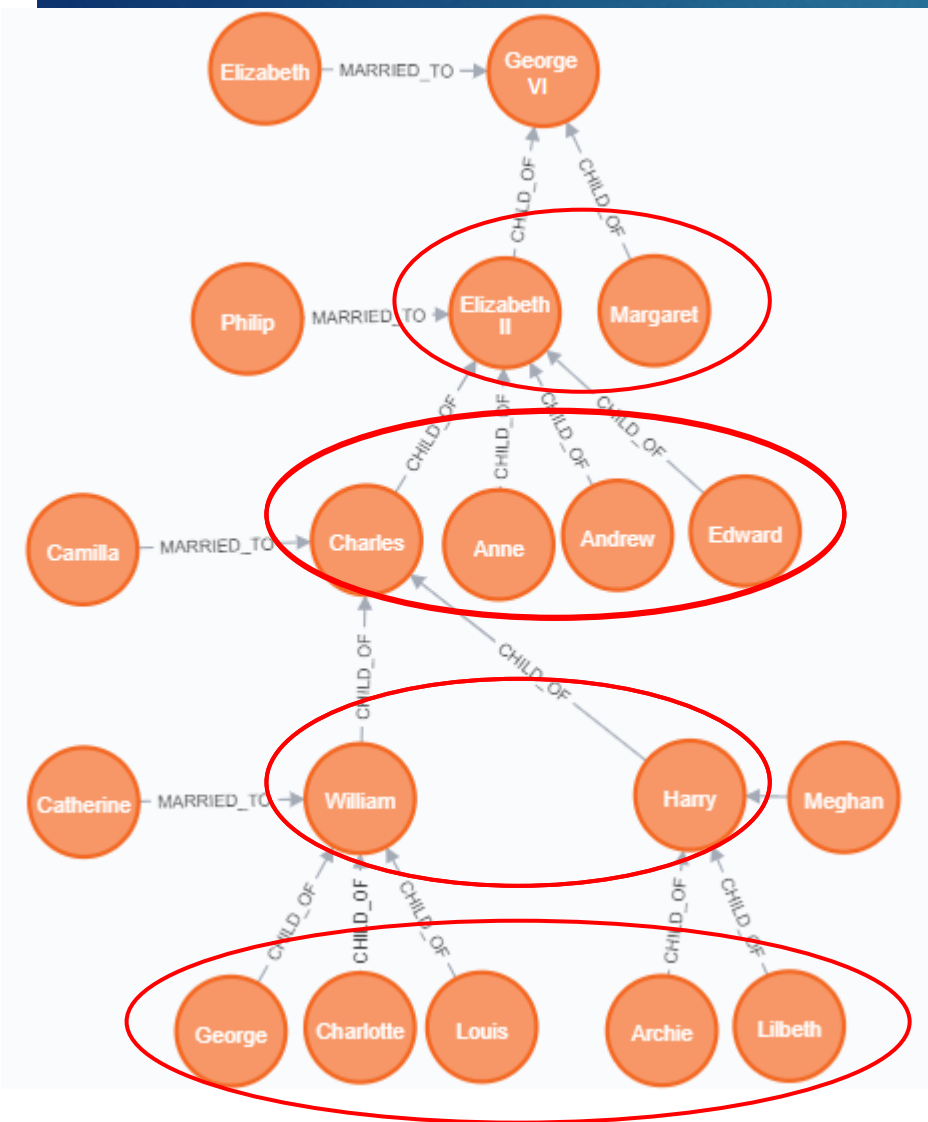Show "George"s great-grandparent.

```
MATCH(g{name:"George"})
     -[:CHILD_OF]->
     ()
     -[:CHILD_OF]->
     ()
     -[:CHILD_OF]->
     (ggp)
RETURN ggp.name
```

```
MATCH(g{name:"George"})
     -[:CHILD_OF*3]->
     (gp)
RETURN gp.name
```

Show "George VI"'s grandchildren and great-grandchildren.

```
MATCH(g{name:"George VI"})
    <-[:CHILD_OF*2..3]-
    (gc)
RETURN gc.name
```

*min.. max

Show all "George VI"'s descendants except his children.

```
MATCH(g{name:"George VI"})
    <-[:CHILD_OF*2..]-
    (gc)
RETURN gc.name
```
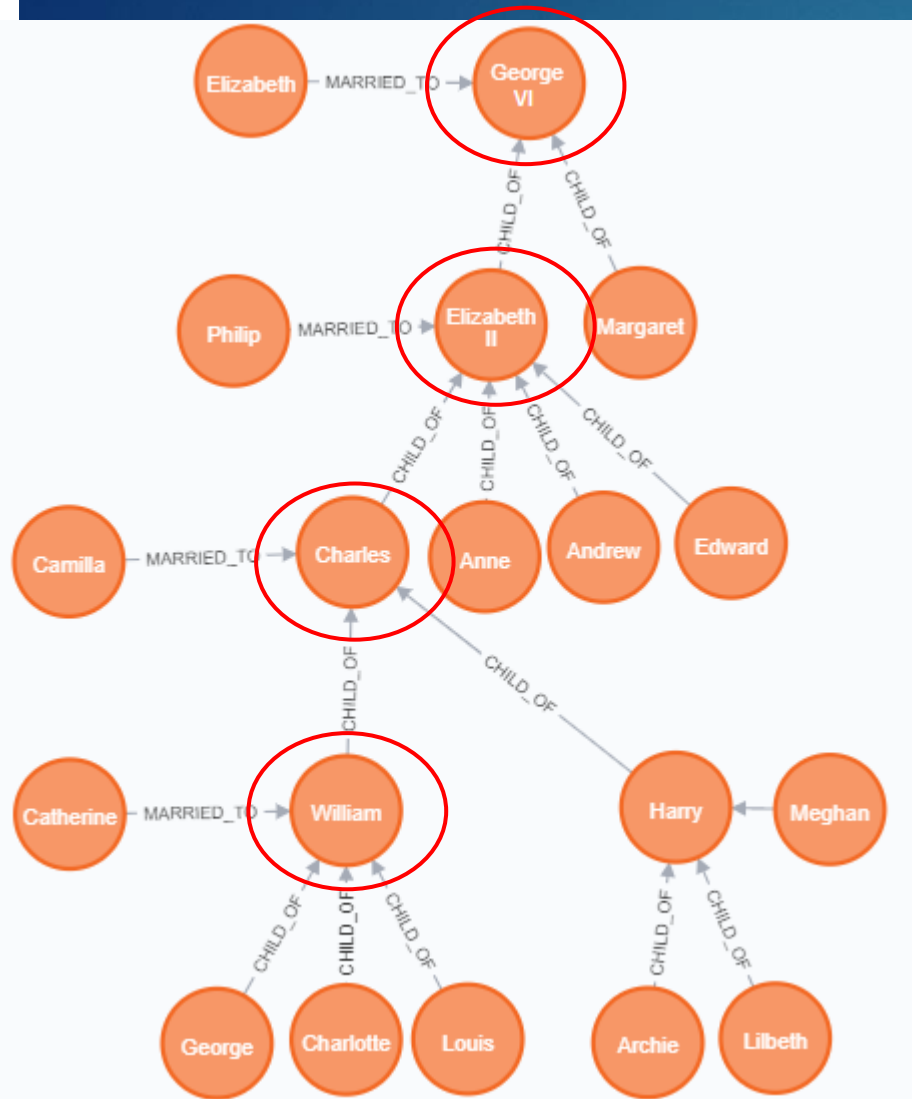
*min.. (no max)

Show all "George VI"'s children and grandchildren.

```
MATCH(g{name:"George VI"})
    <-[:CHILD_OF*..2]-
    (gc)
RETURN gc.name
```

*(no min).. max

Show all "George"'s direct ancestors.

```
MATCH(g{name:"George"})
    -[:CHILD_OF*]->
    (a)
RETURN a.name
```

* *(no min) (no max)*

Variable Length Pattern Matching

[*2]   *exactly 2

[*2..3]   *min 2   max 3

[*2..]   *min 2   no max

[*..2]   *no min   max 2

[*]   *no min   no max