QUESTIONS FROM THE COURSES

Day 03: questions from the course on SPARQL.

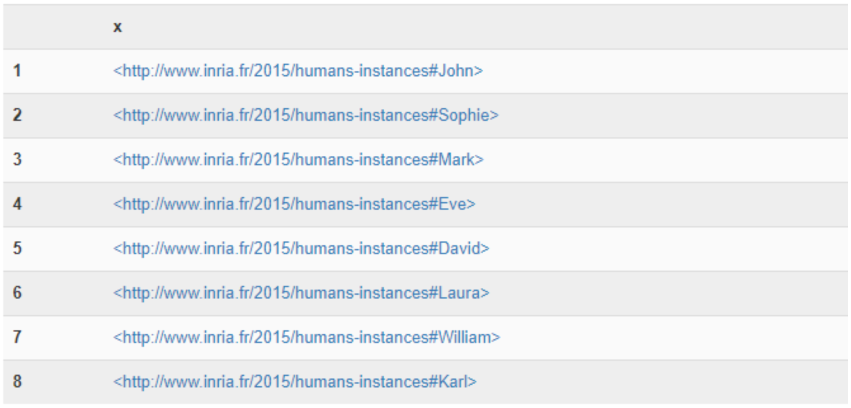
Q3.1 Test SPARQL online

Connect to: https://corese.inria.fr/srv/tutorial/sparql

Answers to the query:

```
prefix v: <http://www.inria.fr/2015/humans#>
select * where {  ?x a v:Person . }
```

Select all which rdf type is person in http://www.inria.fr/2015/humans#m.



	x
1	<http://www.inria.fr/2015/humans-instances#John>
2	<http://www.inria.fr/2015/humans-instances#Sophie>
3	<http://www.inria.fr/2015/humans-instances#Mark>
4	<http://www.inria.fr/2015/humans-instances#Eve>
5	<http://www.inria.fr/2015/humans-instances#David>
6	<http://www.inria.fr/2015/humans-instances#Laura>
7	<http://www.inria.fr/2015/humans-instances#William>
8	<http://www.inria.fr/2015/humans-instances#Karl>

Q3.2 Test SPARQL online

Connect to

http://dbpedia.org/snorql/ or

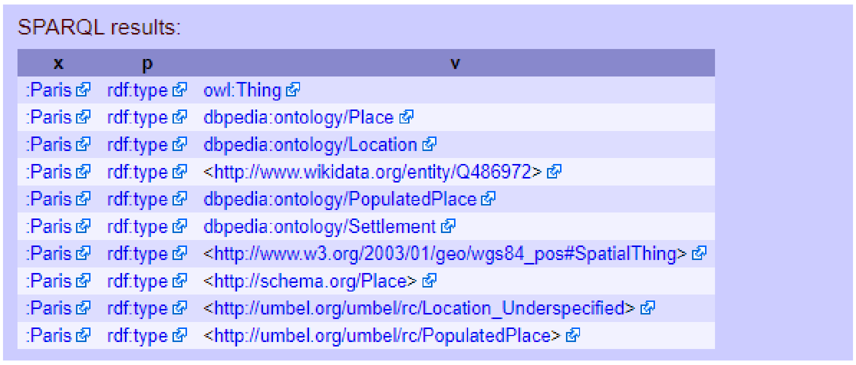
http://fr.dbpedia.org/sparql or …

http://wiki.dbpedia.org/Internationalization/Chapters

Answers to the query:

```
SELECT * WHERE {
  ?x rdfs:label "Paris"@fr .
  ?x ?p ?v .
}
LIMIT 10
```

Select all which rdfs:label is Paris in French, and show triple. Result limited in 10.


SPARQL results:

x	p	v
:Paris	rdf:type	owl:Thing
:Paris	rdf:type	dbpedia:ontology/Place
:Paris	rdf:type	dbpedia:ontology/Location
:Paris	rdf:type	<http://www.wikidata.org/entity/Q486972>
:Paris	rdf:type	dbpedia:ontology/PopulatedPlace
:Paris	rdf:type	dbpedia:ontology/Settlement
:Paris	rdf:type	<http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing>
:Paris	rdf:type	<http://schema.org/Place>
:Paris	rdf:type	<http://umbel.org/umbel/rc/Location_Underspecified>
:Paris	rdf:type	<http://umbel.org/umbel/rc/PopulatedPlace>

Q3.3 Test SPARQL online

Connect to:

https://query.wikidata.org/

What does this query retrieve?

```
SELECT distinct ?p ?n WHERE
{    wd:Q30 p:P6 [ ps:P6 ?p ].
     ?p rdfs:label ?n .
     FILTER (lang(?n)="en") }
```

Discover wd:Q30 using the namespace attached to wd:

PREFIX wd: <http://www.wikidata.org/entity/>

Discover p:P6 using the namespace attached to p:

PREFIX p: <http://www.wikidata.org/prop/>

Find q-name of the property "given name"

https://www.wikidata.org/wiki/Wikidata:List_of_properties

United States of America (Q30)

head of government (P6)

Q3.4 SPARQL query to return 20 persons at most (use type foaf:Person)

SELECT *

WHERE{?x a foaf:Person }

LIMIT 20

Q3.5 SPARQL query to return 20 persons (at most), after the 10th result i.e. from 11th to 30th

```
SELECT *

WHERE{?x a foaf:Person }

LIMIT 20

OFFSET 10
```

Q3.6 You have two properties: c:name and c:age

1.Find the age of resources whose name is ‘Fabien’

2.Find the name of resources whose age is less than 50

3.Find property values of resources whose name is ‘Fabien’ and whose age is less than 50

4.Find other names of resources whose name is ‘Fabien’

5.Find resources which have two different properties with the same value

6.Find resources which have the same property with two different values

select ?c:age where{?x c:name’Fabien’; c:age ?age}
select ?c:name where {?x c:age ?name; c:age ?age. (filter?age>50)}
select ?p ?y where{?x c:name ‘Fabien’; c:age ?age; ?p ?y. filter(?age>50)}
select ?name where{?x c:name ‘Fabien’, ?name. filter(?name != ‘Fabien’)}
select * where{?x ?p ?y; ?q ?y. filter(?p != ?q)}
select * where{?x ?p ?y, ?z. filter(?p != ?z)}

Q3.7 Could this query return `ex:a c:memberOf ex:b` and why ?

```
select * where {
  ?x c:memberOf ?org .
  minus { ex:a c:memberOf ex:b }
}
```

?x is not in the minus filter

Q3.8 get the members of organizations (c:memberOf) but remove the resources author of a document (c:author) by using ‘not exists’

```
select * where {
   ?x c:memberOf ?org .
   filter(! exits { ?x c:memberOf c:author) } )
}
```

Q3.9 what is retrieving this query ?

```
prefix ex: <http://example.org/>
select ?x (count(?doc) as ?c)
where {  ?x ex:author ?doc  }
group by ?x
order by desc(count(?doc))
```

Numbers of author group by their doc.

Q3.10 What expression should we use to find the ?x related to ?y by paths composed of properties foaf:knows and/or rdfs: seeAlso?

- ?x (foaf:knows | rdfs:seeAlso)+ ?y
- ?x foaf:knows+ | rdfs:seeAlso+ ?y
- ?x (foaf:knows / rdfs:seeAlso)+ ?y

?x (foaf:knows | rdfs:seeAlso)+ ?y

Q3.11 what is this query retrieving?

```
prefix foaf: <http://xmlns.com/foaf/0.1/>
select ?x (if (bound(?n), ?n, "John Doe") as ?m)
where {
   ?x foaf:knows ?y
   optional { ?y foaf:name ?n }
}
```

When x knows y, if y have a name return the name, else return “John Doe”

Q3.12 what is this query retrieving?

```
prefix ex:  <http://example.org/>
select ?x (avg(?a) as ?b)
where {
  ?x ex:knows ?y .
  ?y ex:age ?a
}
group by ?x
```

x knows y and y get an age. Group the average age of y, group them.

Q3.13 You have two properties: c:name and c:study and the resources c:Informatics and c:Mathematics

1. Find resources that study informatics or mathematics
2. In addition return the name of the resource if it has a name
3. In addition return the graph where the name is given

select *   where {{?x c:study c:Informatics} union {?x c:study c:Mathematics}}

select *
 where {{?x c:study c:Informatics} union {?x c:study c:Mathematics}
 optional {?x c:name ?name}}

select *
 where {{?x c:study c:Informatics} union {?x c:study c:Mathematics}
 optional {graph {?x c:name ?name}}

Q3.14 On which graph(s) is calculated ?x ?p ?y

On which graph(s) is calculated graph ?g { ?y ?q ?z }

```
prefix ex:  <http://example.org/>
select *
from ex:g1
from named ex:g2
where {
  ?x ?p ?y .
  graph ?g { ?y ?q ?z}  }
```

g1/ g2

Q3.15 Write a query to change foaf:name into rdfs:label

delete{?x foaf:name ?y}

insert{?x rdf:label ?y}

select ?x, ?y where{?x rdf:label ?y}

Q3.16 what is this query performing?

```
prefix ex: <http://example.org/>
delete { ?x ex:age ?a }
insert { ?x ex:age ?i }
where  {
  select ?x (xsd:integer(?a) as ?i)
  where {
    ?x ex:age ?a
    filter(datatype(?a) = xsd:string)
  }
}
```

Change type of age from string to integer.

Q3.17 Which clauses could you use to obtained results as RDF triples following a specific pattern?

- SELECT … WHERE {…} …
- CONSTRUCT { } WHERE {…} …
- DESCRIBE <…>   DESCRIBE … {…}
- ASK {…}
- DELETE { … } INSERT { … } WHERE {…} …

CONSTRUCT { } WHERE {…} …

PRACTICAL SESSIONS

Day 03: Answers to the practical session on SPARQL.

Software requirements

- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine: https://project.inria.fr/corese/

Basic query on RDF human.rdf

If you haven't done it yet download the SPARQL Corese engine.

On Window double-click the file ".jar". If it does not work or on other platforms, run the command " java -jar -Dfile.encoding=UTF8 " followed by the name of the ".jar" archive. Notice that you need java on your machine and proper path configuration

This interface provides two tabs: (1) one to load input files and see traces of execution, and (2) the default tab to start loading or writing queries and see their result.

If you don't have the human dataset file yet download the following file of annotations and save it as "human.rdf":

http://wimmics.inria.fr/doc/tutorial/human_2013.rdf

Load the file human.rdf as RDF data in corese.

Question 1:

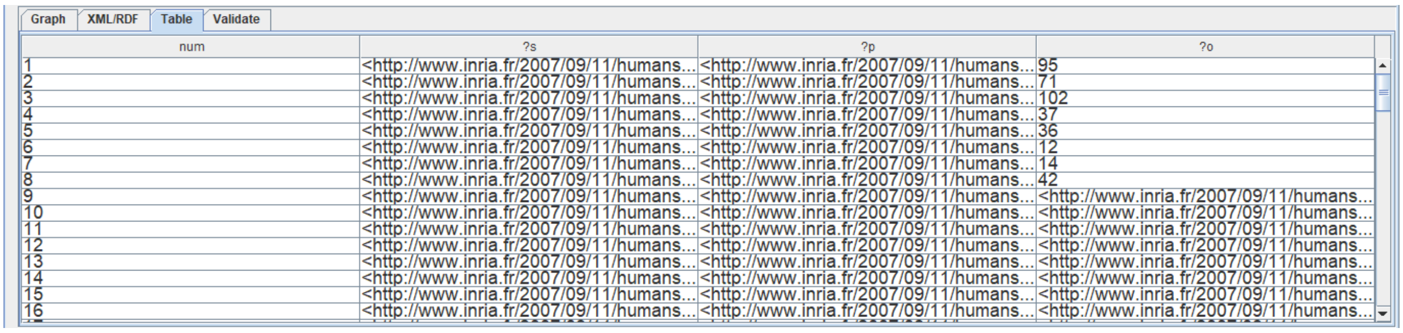
Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

```
CONSTRUCT { ?s ?p ?o } WHERE { ?s ?p ?o }
```

Explanation:

Showing subject , predicate and object in triple table.

Screenshot:



num	?s	?p	?o
1	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	95
2	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	71
3	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	102
4	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	37
5	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	36
6	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	12
7	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	14
8	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	42
9	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
10	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
11	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
12	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
13	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
14	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
15	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...
16	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...	<http://www.inria.fr/2007/09/11/humans...

Question 2:

Create a new tab to enter the following query:

```
prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where { ?x a ?t . filter(strstarts(?t, h:)) }
```

Translate this query in plain English.

Select all instance with rdf:type in <http://www.inria.fr/2007/09/11/humans.rdfs#>

Run this query. How many answers do you get?

21

Find John and his types in the answers.

John's types:

Person

Question 3:

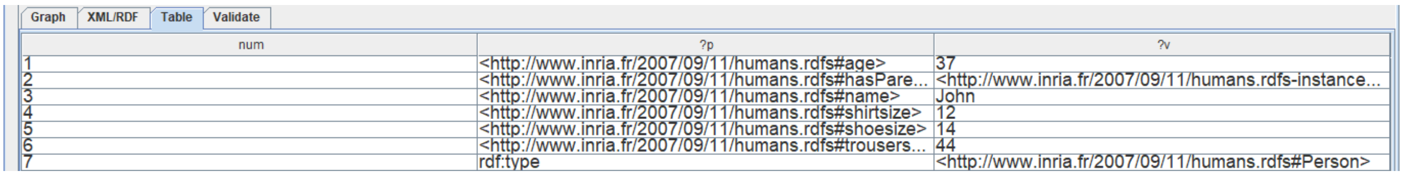
In the previous answer, locate the URI of John.

1. formulate a SELECT query to find all the properties of John, using his URI

   Query

```
select * where {

    <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John> ?p ?v.

}
```

   Results:



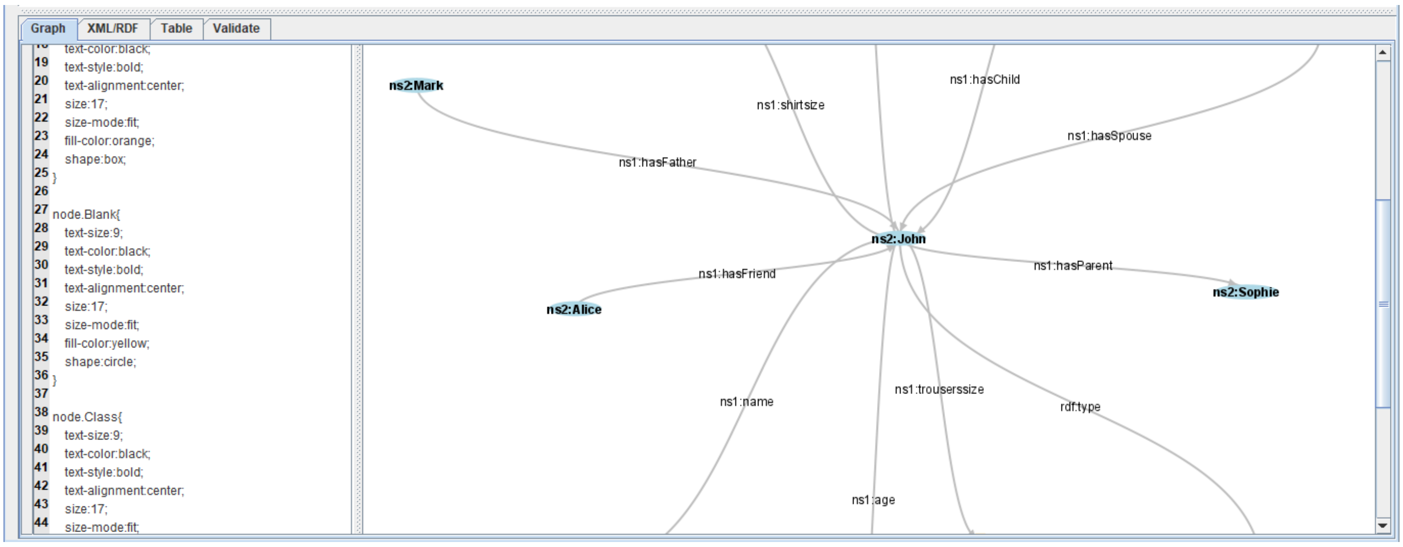
num	?p	?v
1	<http://www.inria.fr/2007/09/11/humans.rdfs#age>	37
2	<http://www.inria.fr/2007/09/11/humans.rdfs#hasPare...	<http://www.inria.fr/2007/09/11/humans.rdfs-instance...
3	<http://www.inria.fr/2007/09/11/humans.rdfs#name>	John
4	<http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize>	12
5	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	14
6	<http://www.inria.fr/2007/09/11/humans.rdfs#trousers...	44
7	rdf:type	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>

2. request a description of John using the SPARQL clause for this.

Query

describe <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>

Results:



```
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:orange;
   shape:box;
}

node.Blank{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:yellow;
   shape:circle;
}

node.Class{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
```

Question 4

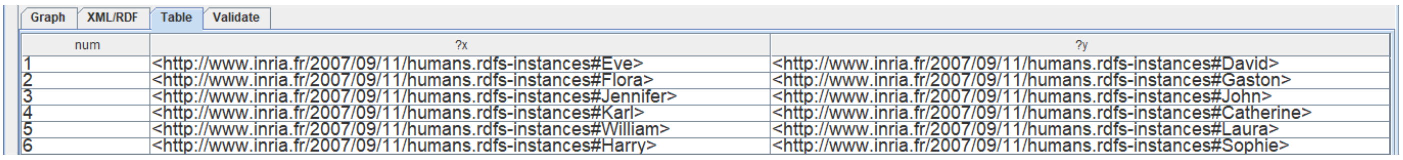
Create a new tab to enter the following query:

```
prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where { ?x h:hasSpouse ?y }
```

Translate this query in plain English.

Select everyone who has a spouse and the spouse.

Run this query. How many answers do you get?



num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>

Question 5:

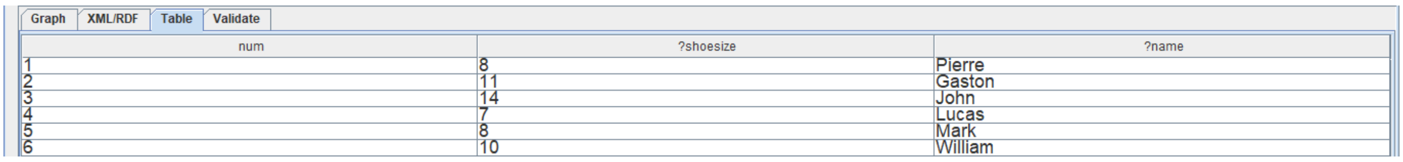
In the RDF file, find the name of the property that is used to give the shoe size of a person.

1. Deduce a query to extract all the persons (h:Person) with their shoe size.

Query:

```
select ?shoesize ?name where {

   ?x <http://www.inria.fr/2007/09/11/humans.rdfs#shoesize> ?shoesize;

   <http://www.inria.fr/2007/09/11/humans.rdfs#name> ?name;

}
```

Result:



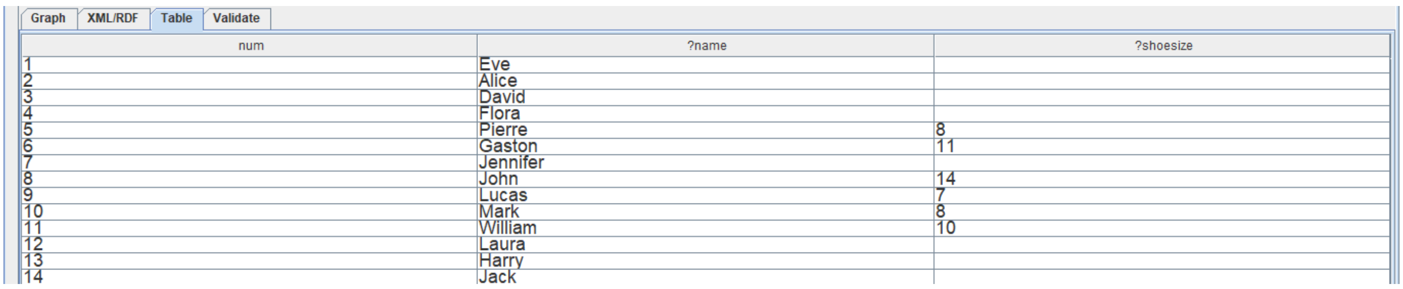
num	?shoesize	?name
1	8	Pierre
2	11	Gaston
3	14	John
4	7	Lucas
5	8	Mark
6	10	William

2. Change this query to retrieve all the persons and, if available, their shoe size.

Query:

```
select ?name ?shoesize where {

   ?x <http://www.inria.fr/2007/09/11/humans.rdfs#name> ?name.

   optional{?x <http://www.inria.fr/2007/09/11/humans.rdfs#shoesize> ?shoesize.}

}
```

Result:



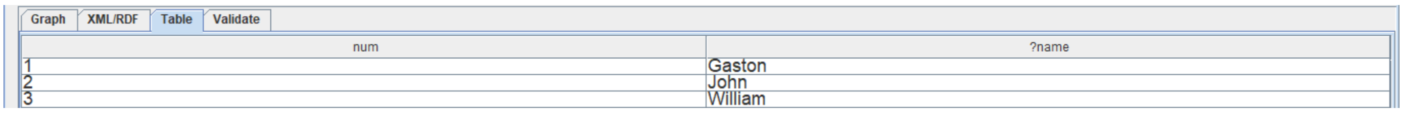
num	?name	?shoesize
1	Eve	
2	Alice	
3	David	
4	Flora	
5	Pierre	8
6	Gaston	11
7	Jennifer	
8	John	14
9	Lucas	7
10	Mark	8
11	William	10
12	Laura	
13	Harry	
14	Jack	

3. Change this query to retrieve all the persons whose shoe size is greater than 8 <u>or</u> whose shirt size is greater than 12.

Query:

```
select ?name where {

   ?x <http://www.inria.fr/2007/09/11/humans.rdfs#name> ?name;

  <http://www.inria.fr/2007/09/11/humans.rdfs#shoesize> ?shoesize;
  <http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize> ?shirtsize.

    FILTER (?shoesize > 8 || ?shirtsize >12).

}
```

Result:



num	?name
1	Gaston
2	John
3	William

Question 6:

In the RDF file, find the name of the property that is used to indicate the children of a person.

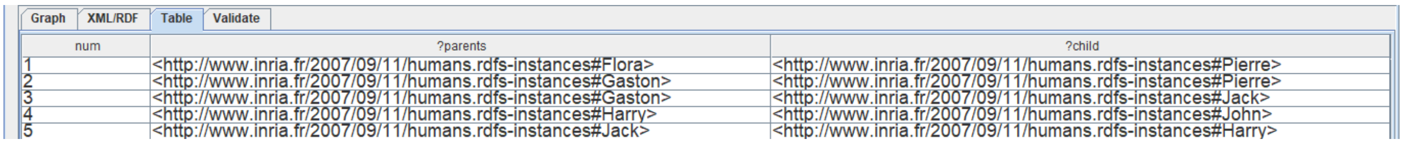
1. Formulate a query to find the parents who have at least one child.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

    ?parents <http://www.inria.fr/2007/09/11/humans.rdfs#hasChild> ?child.

}
```

How many answers do you get? How many duplicates do you identify in these responses?

5 with 1 duplicate



num	?parents	?child
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>

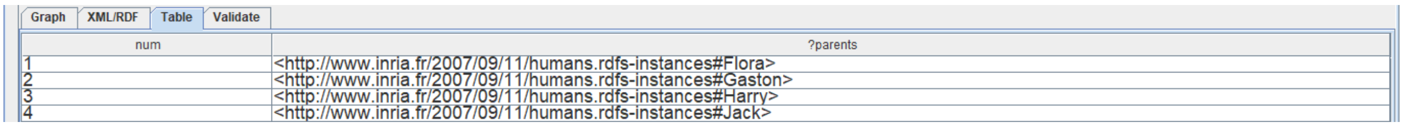
2. Find a way to avoid duplicates.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select distinct ?parents where {

    ?parents h:hasChild ?child;

  }
```

How many answers do you get then?

4

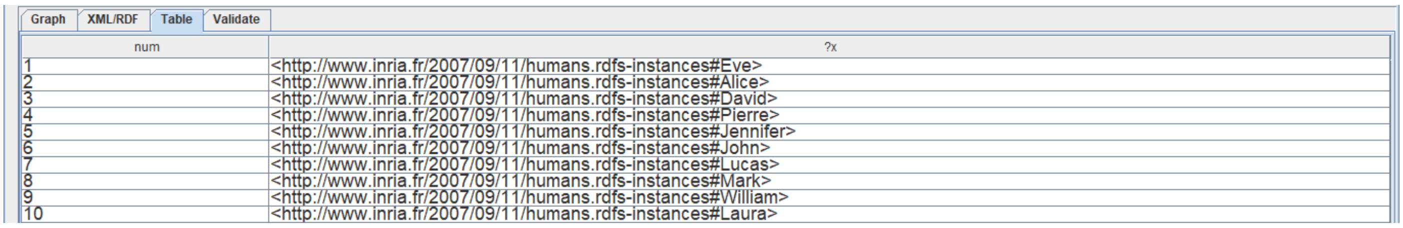


num	?parents
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>

3. Rewrite a query to find the Persons who have no child.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select distinct ?x where {

  ?x h:name ?name.

  FILTER (! exists {?x h:hasChild ?child})

 }
```



num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Question 7

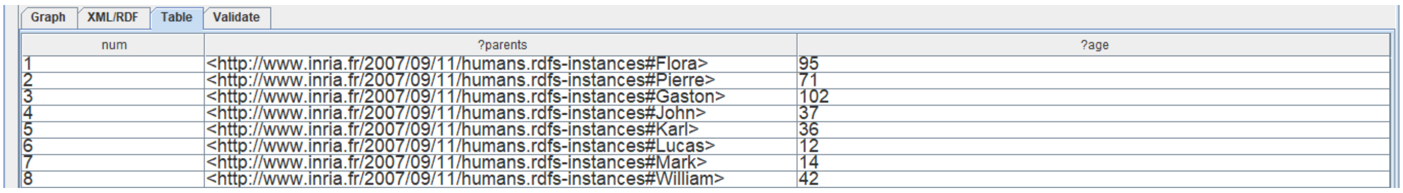
In the RDF file, find the name of the property that is used to give the age of a person.

1. Formulate a query to find people with their age.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

  ?parents h:age ?age

}
```

Result:



num	?parents	?age
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	95
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	71
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	102
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	37
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	36
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	12
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	14
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	42

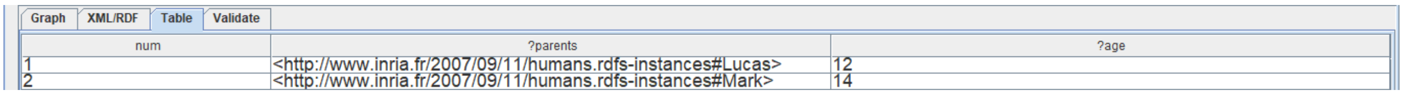
2. Formulate a query to find people who are not adults.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

  ?parents h:age ?age.

  FILTER (?age < 18)

}
```

How many answers do you get?

2



num	?parents	?age
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	12
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	14

3. Use the appropriate query clause to check if Mark is an adult; use the proper clause statement for this type of query to get a true or false answer.

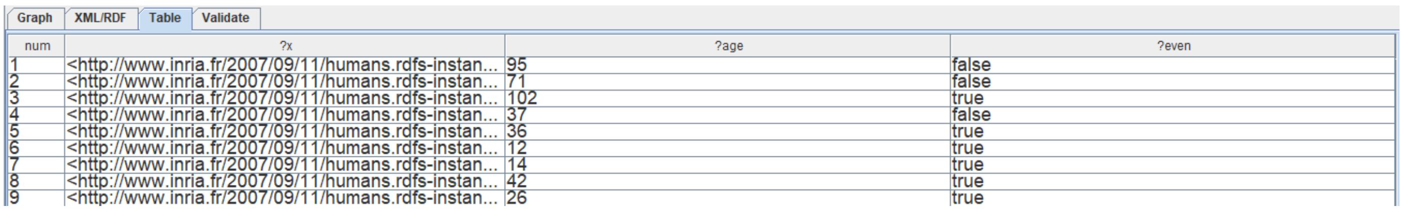
Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

ASK { <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark> h:age ?age. FILTER ( ?age>18) }
```

4. Write a query that indicates for each person if her age is even (true or false).

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

?x h:age ?age.

bind ((?age/2 = floor( ?age/2)) as ?even)

}
```



num	?x	?age	?even
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	95	false
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	71	false
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	102	true
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	37	false
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	36	true
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	12	true
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	14	true
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	42	true
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instan...	26	true

Question 8

1. **Construct** the symmetric of all hasFriend relations using the good SPARQL statement (ex. When finding `Thomas hasFriend Fabien`, your query should construct `Fabien hasFriend Thomas`)

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>
```

CONSTRUCT { ?x h:hasFriend ?y . ?x h:hasFriend ?y. }

WHERE { ?x h:hasFriend ?y. }

2. **<u>Insert</u>** the symmetric of all hasFriend relations using the adequate SPARQL statement but check the results with a select query before and after.

Query:

prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

INSERT { ?y h:hasFriend ?x}

WHERE { ?x h:hasFriend ?y }

Question 9

Choose and edit one of the SELECT WHERE queries previously written to transform them into a CONSTRUCT WHERE query (retaining the same WHERE clause) in order to visualize the results as a graph.

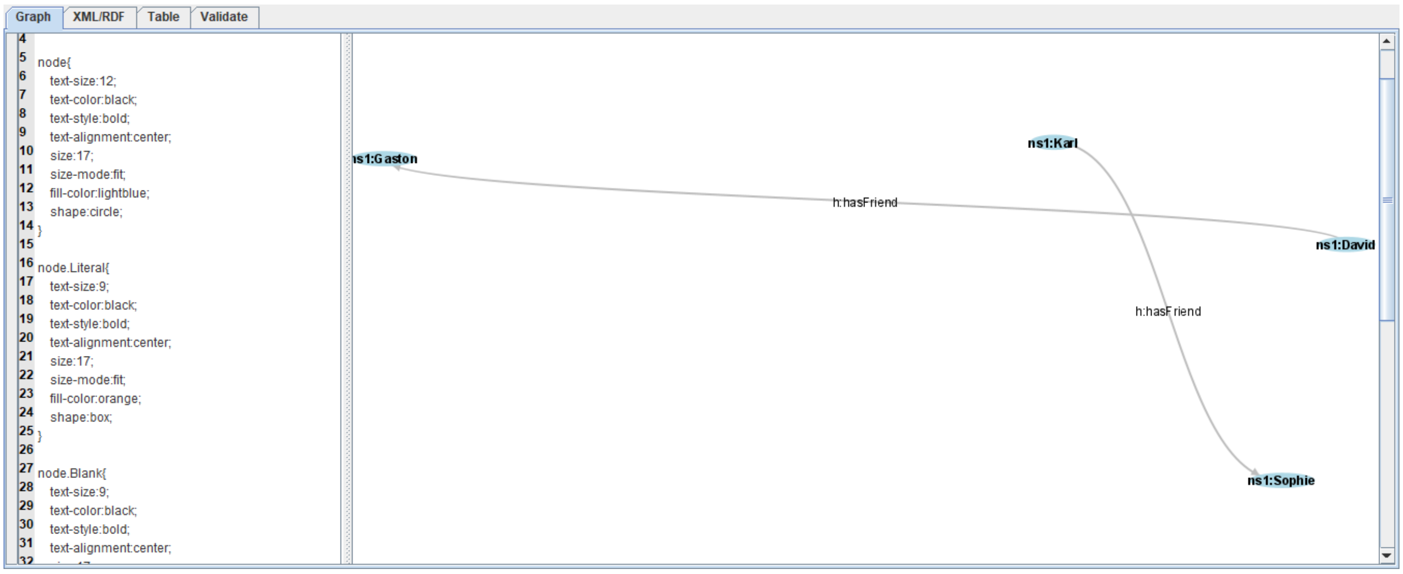
Query:

prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

CONSTRUCT { ?x h:hasFriend ?friend }

WHERE { ?x h:hasFriend ?friend }

Result:



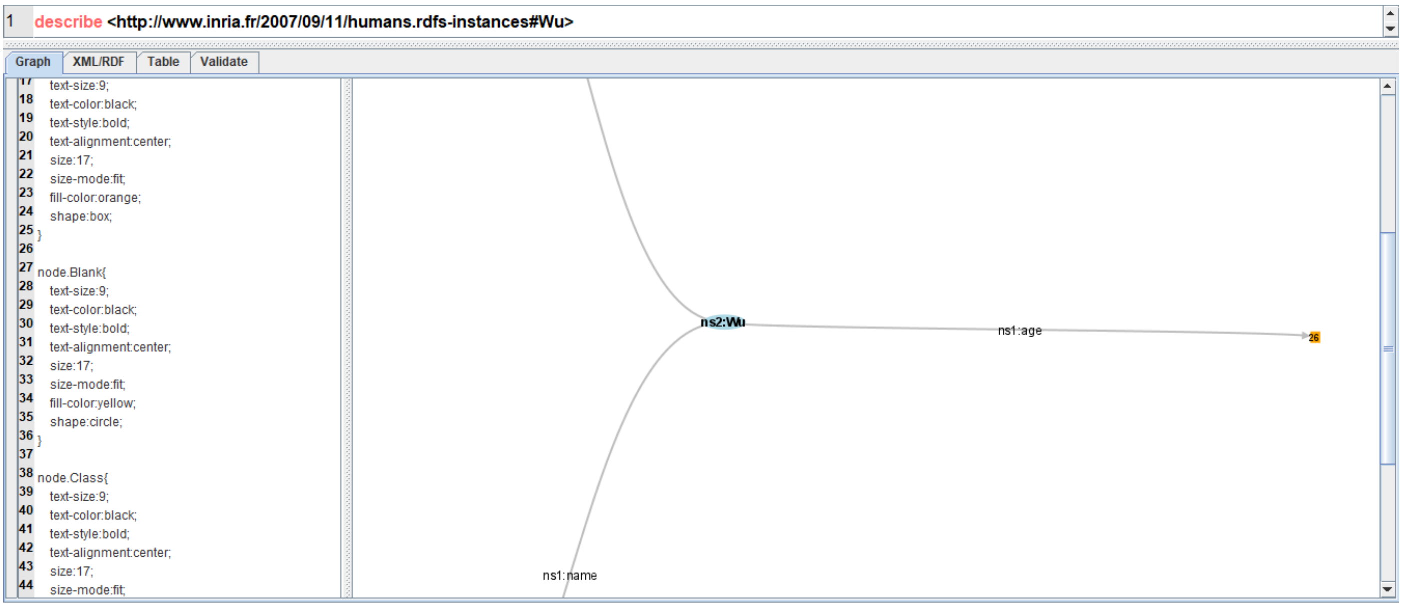
```

node{
   text-size:12;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:lightblue;
   shape:circle;
}

node.Literal{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:orange;
   shape:box;
}

node.Blank{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
```

Question 10

Edit the file to add your own annotation (about you) to the RDF file reusing the properties of the file. Build queries to verify and visualize the annotations you added.

screenshots:

```
@prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix xml: <http://www.w3.org/XML/1998/namespace> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Wu> a :Woman ;

        :age 26 ;

        :name "Wu" ;

        :shirtsize 4 ;

        :shoesize 37 ;

        :trouserssize 36 .
```



```
describe <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Wu>
```

```
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:orange;
    shape:box;
}

node.Blank{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:yellow;
    shape:circle;
}

node.Class{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
```

Question 11

1. Formulate a query to find the persons who share the same shirt size.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

  ?p1 h:shirtsize ?size.
```

```
    ?p2 h:shirtsize ?size.

    filter (?p1 != ?p2)

}
```

2. Find the persons who have the same size shirt and construct a seeAlso relationship between them.

Query:

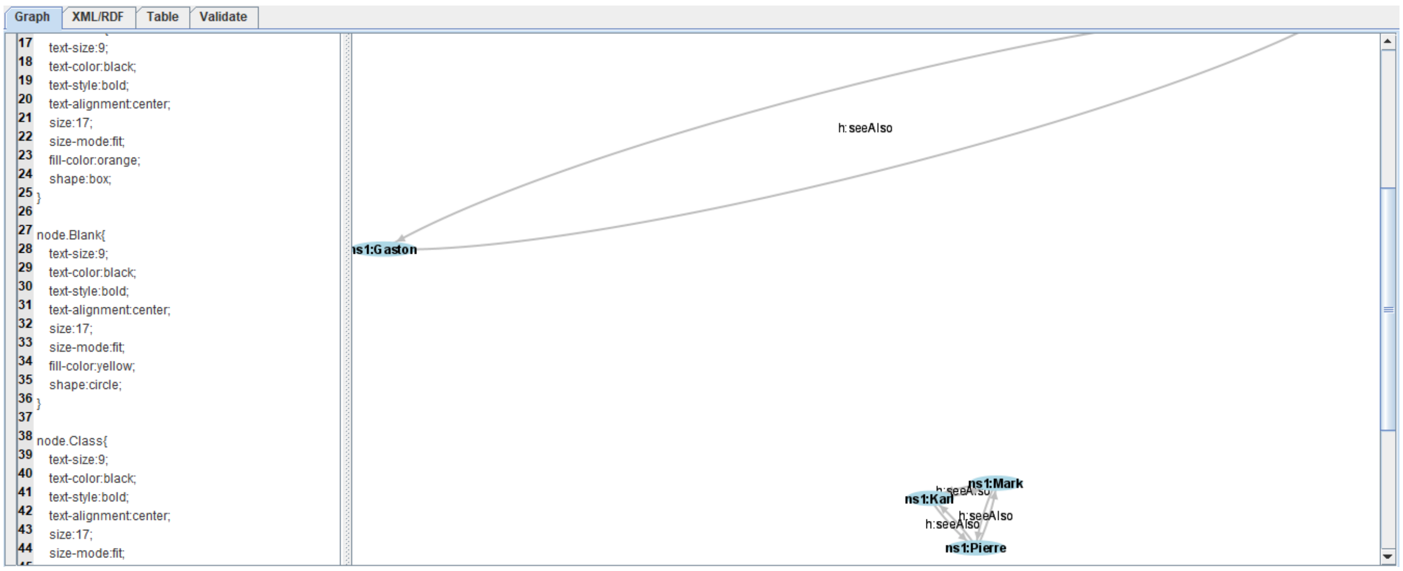
```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

construct {?p1 h:seeAlso ?p2}

where {

    ?p1 h:shirtsize ?size.

    ?p2 h:shirtsize ?size.

    filter (?p1 != ?p2)

}
```

3. Change the query into an insert.

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

insert {?p1 h:seeAlso ?p2}

where {

    ?p1 h:shirtsize ?size.

    ?p2 h:shirtsize ?size.

    filter (?p1 != ?p2)

}
```

4. Visualize the resources connected by seeAlso (use the CONSTRUCT clause).

screenshot:



```
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:orange;
   shape:box;
}

node.Blank{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:yellow;
   shape:circle;
}

node.Class{
   text-size:9;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
```

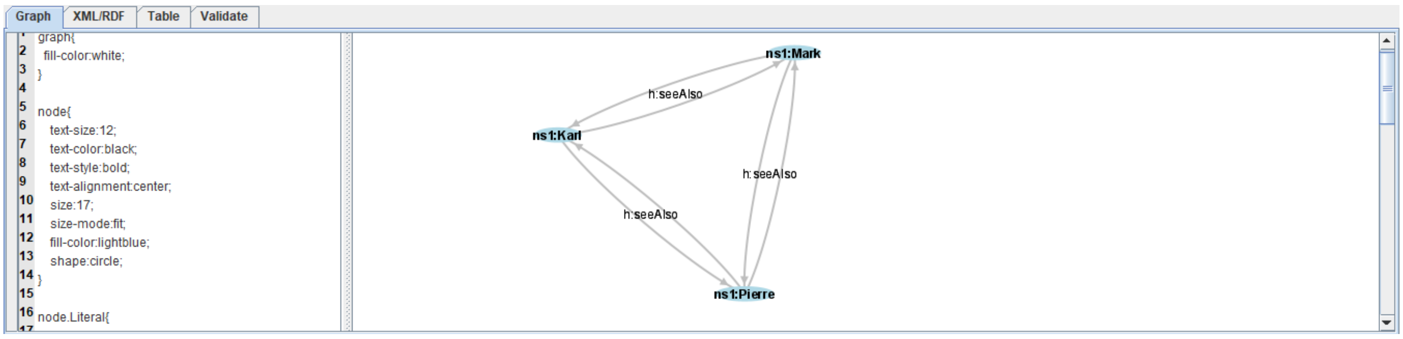
5. Adapt the first query to find persons who have the same shoe size and insert a seeAlso relationship between them.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

insert {?p1 h:seeAlso ?p2}

where {

   ?p1 h:shirtsize ?size.

   ?p2 h:shirtsize ?size.

   filter (?p1 != ?p2)

}
```

6. Visualize the resources connected by seeAlso (use the CONSTRUCT clause)

screenshot:



```
graph{
  fill-color:white;
}

node{
   text-size:12;
   text-color:black;
   text-style:bold;
   text-alignment:center;
   size:17;
   size-mode:fit;
   fill-color:lightblue;
   shape:circle;
}

node.Literal{
```

7. Change the query to find the resources connected by a path consisting of one or several seeAlso relationships.

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>
```

```
select distinct ?p1

where {?p1 h:seeAlso ?p2}
```

8. Reload the engine (option reload in the menu) and rerun the last visualization query.

Question 12

1. Find the largest shoe size

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select ?x (max(?y) as ?max) where {

  ?x h:shoesize ?y

}
```

2. Find people who have the biggest size of shoe (subquery + aggregate)

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select ?name ?max {

  {select (max(?y) as ?max) where {

    ?x h:shoesize ?y}

  }

  ?name h:shoesize ?max

}
```

3. Calculate the average shoe size using the appropriate aggregation operator

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select (avg(?y) as ?avg) where {

  ?x h:shoesize ?y

}
```

4. Check the average with your own calculation using `sum()` and `count()`

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select (sum(?y) / count(?x) as ?avg) where {
    ?x h:shoesize ?y

}
```

Question 13

Find couples without children

Query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where {

    ?x h:hasSpouse ?y

    minus {?x h:hasChild ?z}

}
```

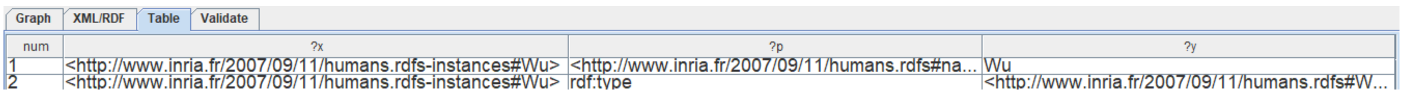
Question 14

Using INSERT DATA, create a new person with its properties. Then, check that it has been created.

Insert:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

INSERT DATA {<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Wu> a h:Woman; h:name "Wu"}
```

Screenshot result:



num	?x	?p	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Wu>	<http://www.inria.fr/2007/09/11/humans.rdfs#na...	Wu
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Wu>	rdf:type	<http://www.inria.fr/2007/09/11/humans.rdfs#W...

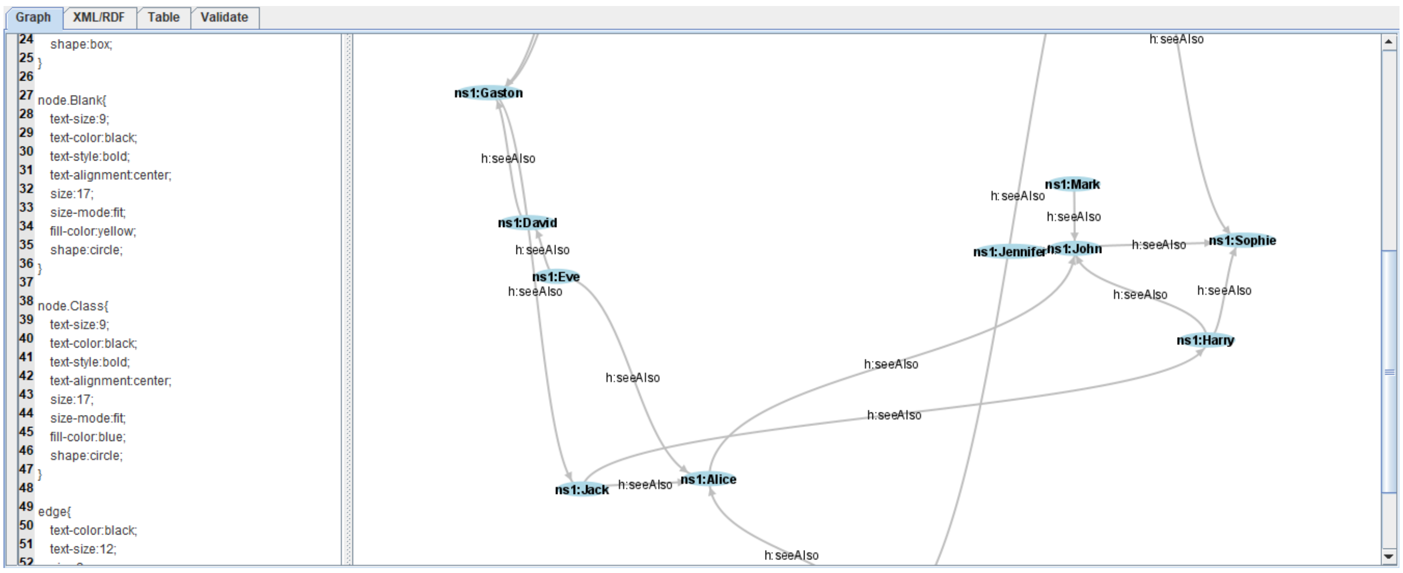
Question 15

Find the people connected by paths of any family links. Construct an arc seeAlso between them to visualize the result.

query:

```
prefix h:<http://www.inria.fr/2007/09/11/humans.rdfs#>

construct {?x h:seeAlso ?y}

where {?x ?p ?y. filter(strstarts(?p, h:has))}
```

screenshot:



```
    shape:box;
}

node.Blank{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:yellow;
    shape:circle;
}

node.Class{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:blue;
    shape:circle;
}

edge{
    text-color:black;
    text-size:12;
```

Question 16

Run the following query:

```
prefix db: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
construct  { ?x h:name ?nx . ?y h:name ?ny . ?x h:hasSpouse ?y }
where {
  service <http://fr.dbpedia.org/sparql/> {
     select * where {
       ?x db:spouse ?y .
       ?x foaf:name ?nx .
       ?y foaf:name ?ny .
     }
     limit 20
  }
}
```

Explain what it does

Connect to <http://fr.dbpedia.org/sparql/>. Select the name of those who has spouse in <http://dbpedia.org/ontology/> , and the name of the spouse name.

modify it to <u>insert</u> new persons in the base and check the results.

query:

prefix db: <http://dbpedia.org/ontology/>

prefix foaf: <http://xmlns.com/foaf/0.1/>

prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

```
INSERT DATA { ?x h:name foaf:name. ?y h:name foaf:name. ?x h:hasSpouse db:spouse. }

limit 20
```