Semantic Web handout including: lecture questions and practical sessions

In this document, you must provide your answers to the questions asked during the course **and** to the questions of the practical sessions; everything in one document.

The questions of the course have been repeated here; **do not delete the questions** but provide your answer to each question just below the question. You can use screenshots when appropriate as an answer to a question but keep your answers and this file as small and concise as possible.

At the end, you must generate and submit only one final PDF file based on this template.

In questions where you are asked to create, invent or use your own data, make sure they are different from other student's.

First name: Lan

Family name: LE

Email: le.lan@edu.dsti.institute

TABLE OF CONTENTS:

QUESTIONS FROM THE COURSES

Questions from the course on Linked Data.

Q1.1 Practice XML replace missing parts

```
<archi_book>
 <AAAA>Architecture Now</short_title>
 <main_author>Jodidio, Philip<BBBB>
 <ID isbn10="3822840912" CCCC>
<DDDD>
```

```
<archi_book>
 <short_title>Architecture Now</short_title>
 <main_author>Jodidio, Philip</main_author>
 <ID isbn10="3822840912" />
</archi_book>
```

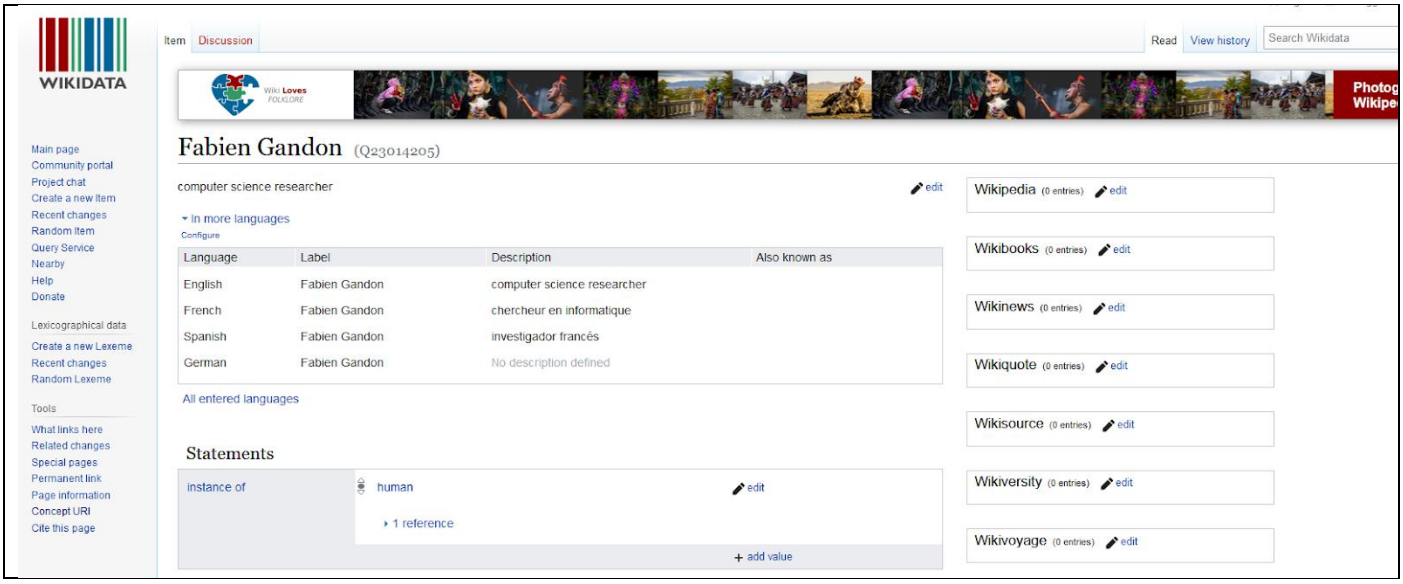
Q1.2 Provide 10 first lines

Get 10 first lines of the five results for:

http://www.wikidata.org/entity/Q23014205
http://www.wikidata.org/entity/Q23014205.json
http://www.wikidata.org/entity/Q23014205.rdf
http://www.wikidata.org/entity/Q23014205.ttl
http://www.wikidata.org/entity/Q23014205.nt

http://www.wikidata.org/entity/Q23014205

The URI changes (entity -> wiki) and we were redirected to a web page



http://www.wikidata.org/entity/Q23014205.json

{"entities":{"Q23014205":{"pageid":25028548,"ns":0,"title":"Q23014205","lastrevid":1840915886,"modified":"2023-02-24T23:58:53Z","type":"item","id":"Q23014205","labels":{"fr":{"language":"fr","value":"Fabien Gandon"},"en":{"language":"en","value":"Fabien Gandon"},"br":{"language":"br","value":"Fabien Gandon"},"de":{"language":"de","value":"Fabien Gandon"},"af":{"language":"af","value":"Fabien Gandon"},"an":{"language":"an","value":"Fabien Gandon"},"ast":{"language":"ast","value":"Fabien Gandon"},"bar":{"language":"bar","value":"Fabien Gandon"},"bm":{"language":"bm","value":"Fabien Gandon"},"ca":{"language":"ca","value":"Fabien Gandon"},"co":{"language":"co","value":"Fabien Gandon"},"cs":{"language":"cs","value":"Fabien Gandon"},"cy":{"language":"cy","value":"Fabien Gandon"},"da":{"language":"da","value":"Fabien Gandon"},"de-at":{"language":"de-at","value":"Fabien Gandon"},"de-ch":{"language":"de-ch","value":"Fabien Gandon"},"en-ca":{"language":"en-ca","value":"Fabien Gandon"},"en-gb":{"language":"en-gb","value":"Fabien Gandon"},"eo":{"language":"eo","value":"Fabien Gandon"},"es":{"language":"es","value":"Fabien Gandon"},"et":{"language":"et","value":"Fabien Gandon"},"eu":{"language":"eu","value":"Fabien Gandon"},"fi":{"language":"fi","value":"Fabien Gandon"},"frc":{"language":"frc","value":"Fabien Gandon"},

http://www.wikidata.org/entity/Q23014205.rdf

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:ontolex="http://www.w3.org/ns/lemon/ontolex#"
xmlns:dct="http://purl.org/dc/terms/" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:wikibase="http://wikiba.se/ontology#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:schema="http://schema.org/"
xmlns:cc="http://creativecommons.org/ns#" xmlns:geo="http://www.opengis.net/ont/geosparql#"
xmlns:prov="http://www.w3.org/ns/prov#" xmlns:wd="http://www.wikidata.org/entity/"
xmlns:data="https://www.wikidata.org/wiki/Special:EntityData/"
xmlns:s="http://www.wikidata.org/entity/statement/" xmlns:ref="http://www.wikidata.org/reference/"
xmlns:v="http://www.wikidata.org/value/" xmlns:wdt="http://www.wikidata.org/prop/direct/"
xmlns:wdtn="http://www.wikidata.org/prop/direct-normalized/" xmlns:p="http://www.wikidata.org/prop/"
xmlns:ps="http://www.wikidata.org/prop/statement/" xmlns:psv="http://www.wikidata.org/prop/statement/value/"
xmlns:psn="http://www.wikidata.org/prop/statement/value-normalized/"
xmlns:pq="http://www.wikidata.org/prop/qualifier/" xmlns:pqv="http://www.wikidata.org/prop/qualifier/value/"
xmlns:pqn="http://www.wikidata.org/prop/qualifier/value-normalized/"
xmlns:pr="http://www.wikidata.org/prop/reference/" xmlns:prv="http://www.wikidata.org/prop/reference/value/"

```
xmlns:prn="http://www.wikidata.org/prop/reference/value-normalized/"
xmlns:wdno="http://www.wikidata.org/prop/novalue/">
        <rdf:Description rdf:about="https://www.wikidata.org/wiki/Special:EntityData/Q23014205">
                <rdf:type rdf:resource="http://schema.org/Dataset"/>
                <schema:about rdf:resource="http://www.wikidata.org/entity/Q23014205"/>
                <cc:license rdf:resource="http://creativecommons.org/publicdomain/zero/1.0/"/>
                <schema:softwareVersion>1.0.0</schema:softwareVersion>
                <schema:version
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1840915886</schema:version>
                <schema:dateModified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2023-02-
24T23:58:53Z</schema:dateModified>
                <wikibase:statements
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">40</wikibase:statements>
```

http://www.wikidata.org/entity/Q23014205.ttl

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix wikibase: <http://wikiba.se/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix schema: <http://schema.org/> .
@prefix cc: <http://creativecommons.org/ns#> .
```

http://www.wikidata.org/entity/Q23014205.nt

```
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://schema.org/Dataset> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://schema.org/about>
<http://www.wikidata.org/entity/Q23014205> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://creativecommons.org/ns#license>
<http://creativecommons.org/publicdomain/zero/1.0/> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://schema.org/softwareVersion> "1.0.0" .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://schema.org/version>
"1840915886"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://schema.org/dateModified> "2023-02-
24T23:58:53Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/ontology#statements>
"40"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/ontology#sitelinks>
"0"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/ontology#identifiers>
"19"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://www.wikidata.org/entity/Q23014205> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://wikiba.se/ontology#Item> .
```

Q1.3 DBpedia

1. Find “London” on DBpedia.org; e.g. Google: "london site:dbpedia.org“
   make sure you are on the English chapter (dbpedia.org) as there are many others (fr.dbpedia.org, de.dbpedia.org)
2. Find dbp:populationDemonym and give its value
3. Find rdf:type and click on value yago:WikicatCapitalsInEurope

4. Find “Vienna” and get its URI
   (careful: with content negotiation and redirection, the URL of the page you are currently viewing may be different from the URI of the resource it describes)
5. ISO code of the Vienna region?

Answer:

1. https://dbpedia.org/page/London
2. Londoner
3. https://dbpedia.org/class/yago/WikicatCapitalsInEurope
   London is a subcategory of the category "Capitals in Europe" in the YAGO ontology
4. https://dbpedia.org/page/Vienna
5. AT-9

Q1.4 WHO.IS?

1. contact for inria.fr
2. contact for fabien.info
3. contact for lemonde.fr

Answer:

1. florian.dufour@inria.fr
2. REDACTED FOR PRIVACY
3. domain_names@lemonde.fr

Q1.5 CURL (or WGET)

1. Ten first lines:

```
curl -o Paris.html -L -H "Accept: text/html" http://dbpedia.org/resource/Paris

curl -o Paris-rdf.xml -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Paris
```

2. Ten first lines for HTML and RDF http://ns.inria.fr/fabien.gandon#me
3. Ten first lines for HTML and RDF for ‘Vienna’ on Dbpedia
4. Ten first lines for the “URI of the name of Victor Hugo” in the Library of Congress:
   http://id.loc.gov/authorities/names/n79091479
5. Ten first lines for HTML and RDF
   https://purl.uniprot.org/uniprot/P43121
6. What is the topic and format of data obtained with
   curl -o data.json -L -H "Accept: application/json" https://www.wikidata.org/wiki/Special:EntityData/Q551861
7. What is the topic and format of data obtained with
   curl -o data.ttl -L -H "Accept: text/turtle" http://dx.doi.org/10.1007/3-540-45741-0_18

1a.

```
<!DOCTYPE html>
<html
  prefix="
    dbp: http://dbpedia.org/property/
    dbo: http://dbedia.org/ontology/
```

```
    dct: http://purl.org/dc/terms/
    dbd: http://dbpedia.org/datatype/
  og:  https://ogp.me/ns#
  "
>
```

1b.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:skos="http://www.w3.org/2004/02/skos/core#"
        xmlns:dbp="http://dbpedia.org/property/"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
        xmlns:dbo="http://dbpedia.org/ontology/"
```

2a. HTML

```
<!DOCTYPE html>
<!-- saved from url=(0046)http://www-sop.inria.fr/members/Fabien.Gandon/ -->
<html class="no-js mdl-js" lang="en" pwa-launched="true" pwa-extension-
id="npnbdojkgkbcdfdjlfdmplppdphlhhcf" pwa-extension-url-root="chrome-
extension://npnbdojkgkbcdfdjlfdmplppdphlhhcf/"><!--<![endif]--><head><meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<title>Fabien Gandon - Homepage</title>

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="viewport" content="width=device-width">
<link rel="shortcut icon" type="image/x-icon" href="http://www-
sop.inria.fr/members/Fabien.Gandon/img/fabico.ico">
<link rel="shortcut icon" type="image/png" href="http://www-
sop.inria.fr/members/Fabien.Gandon/img/fabico.png">
<!-- link href='http://fonts.googleapis.com/css?family=Open+Sans:400italic,400,700' rel='stylesheet'
type='text/css'-->
```

2b. RDF

```
<?xml version='1.0' encoding='utf-8' ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
          xml:base="http://ns.inria.fr/fabien.gandon">
          <foaf:PersonalProfileDocument rdf:about="">
           <foaf:maker rdf:resource="#me"/>
                 <foaf:primaryTopic rdf:resource="#me"/>
```

3a. HTML

```
<!DOCTYPE html>
<html
```

```
  prefix="
    dbp: http://dbpedia.org/property/
    dbo: http://dbedia.org/ontology/
    dct: http://purl.org/dc/terms/
    dbd: http://dbpedia.org/datatype/
        og:  https://ogp.me/ns#
        "
>
```

3b. RDF

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:skos="http://www.w3.org/2004/02/skos/core#"
        xmlns:dbp="http://dbpedia.org/property/"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
        xmlns:dbo="http://dbpedia.org/ontology/"
```

4.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<!-- saved from url=(0051)https://id.loc.gov/authorities/names/n79091479.html -->
<html version="XHTML+RDFa 1.0" xmlns="http://www.w3.org/1999/xhtml" xmlns:madsrdf="http://www.loc.gov/mads/rdf/v1#" xmlns:ri="http://id.loc.gov/ontologies/RecordInfo#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:skosxl="http://www.w3.org/2008/05/skos-xl#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:cs="http://www.w3.org/2003/06/sw-vocab-status/ns#" xmlns:dcterms="http://purl.org/dc/terms/"><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Hugo, Victor, 1802-1885 - LC Linked Data Service: Authorities and Vocabularies | Library of Congress</title>
  <meta name="description" content=" The Linked Data Service provides access to commonly found standards and vocabularies promulgated by the Library of Congress. This includes data values and the controlled vocabularies that house them.  Datasets available include LCSH, BIBFRAME, LC Name Authorities, LC Classification, MARC codes, PREMIS vocabularies, ISO language codes, and more.">
  <link rel="schema.DC" href="http://purl.org/dc/elements/1.1/">
  <link rel="dc.relation.isPartOf" href="https://www.loc.gov/" title="Library of Congress">
  <meta name="dc.title" content="Hugo, Victor, 1802-1885 - LC Linked Data Service: Authorities and Vocabularies | Library of Congress, from LC Linked Data Service: Authorities and Vocabularies (Library of Congress)">
  <meta name="dc.contributor" content="The Library of Congress">
  <meta name="dc.subject" content="Library of Congress Subject Headings">
```

5a. HTML

```
<!doctype html><html lang="en"><head><meta charset="utf-8"/><title>UniProt</title><meta name="viewport" content="width=device-width,initial-scale=1"/><meta name="theme-color" content="#00639a"/><link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png"/><link rel="shortcut icon" type="image/png" sizes="144x144" href="/mstile-144x144.png"/><link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png"/><link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png"/><link rel="manifest"
```

```
href="/manifest.json"/><link rel="preconnect" href="https://fonts.gstatic.com/"/><link rel="preconnect"
href="https://rest.uniprot.org/"/><link rel="preload" as="style"
href="https://fonts.googleapis.com/css?family=Lato:400,700|Source+Sans+Pro:600,700&display=swap"/><script
async src="https://www.googletagmanager.com/gtag/js?id=G-V6TXEC4BDF"></script><script>window.dataLayer
= window.dataLayer || [];
    function gtag() {
      dataLayer.push(arguments);
    }
    gtag('js', new Date());

    // GA4
    gtag('config', 'G-V6TXEC4BDF', {
      anonymize_ip: true,
      allow_google_signals: false,
```


5b. RDF


```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xml:base="http://purl.uniprot.org/uniprot/" xmlns="http://purl.uniprot.org/core/"
xmlns:ECO="http://purl.obolibrary.org/obo/ECO_" xmlns:annotation="http://purl.uniprot.org/annotation/"
xmlns:citation="http://purl.uniprot.org/citations/" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:disease="http://purl.uniprot.org/diseases/" xmlns:enzyme="http://purl.uniprot.org/enzyme/"
xmlns:faldo="http://biohackathon.org/resource/faldo#" xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:go="http://purl.obolibrary.org/obo/GO_" xmlns:isoform="http://purl.uniprot.org/isoforms/"
xmlns:keyword="http://purl.uniprot.org/keywords/" xmlns:location="http://purl.uniprot.org/locations/"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:position="http://purl.uniprot.org/position/"
xmlns:pubmed="http://purl.uniprot.org/pubmed/" xmlns:range="http://purl.uniprot.org/range/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:taxon="http://purl.uniprot.org/taxonomy/" xmlns:tissue="http://purl.uniprot.org/tissues/">
<owl:Ontology rdf:about="http://purl.uniprot.org/uniprot/">
<owl:imports rdf:resource="http://purl.uniprot.org/core/"/>
</owl:Ontology>
<rdf:Description rdf:about="http://purl.uniprot.org/uniprot/P43121">
<rdf:type rdf:resource="http://purl.uniprot.org/core/Protein"/>
<reviewed rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</reviewed>
<created rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1995-11-01</created>
<modified rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2022-12-14</modified>
```


6.

- Topic: Xavier Dolan
- Format: RDF in JSON format file

7.

- Topic: “Distributed Artificial Intelligence for Distributed Corporate Knowledge Management” article.
- Format: RDF in Turtle format

Q1.6 Find the URIs of « Pedro Almodóvar » on the Spanish Dbpedia and on Wikidata.

Answer:
http://es.dbpedia.org/resource/Pedro_Almod%C3%B3var
https://www.wikidata.org/wiki/Q55171

Q1.7 Spotlight demo

Reproduce the demo:

1. Copy a text from Wikipedia (e.g. Muse Band page)
2. Find the DBpedia Spotlight service page
3. Paste the text and run the detection
4. Try with other texts and copy-paste one of the results you get.

Answer:

https://www.dbpedia-spotlight.org/

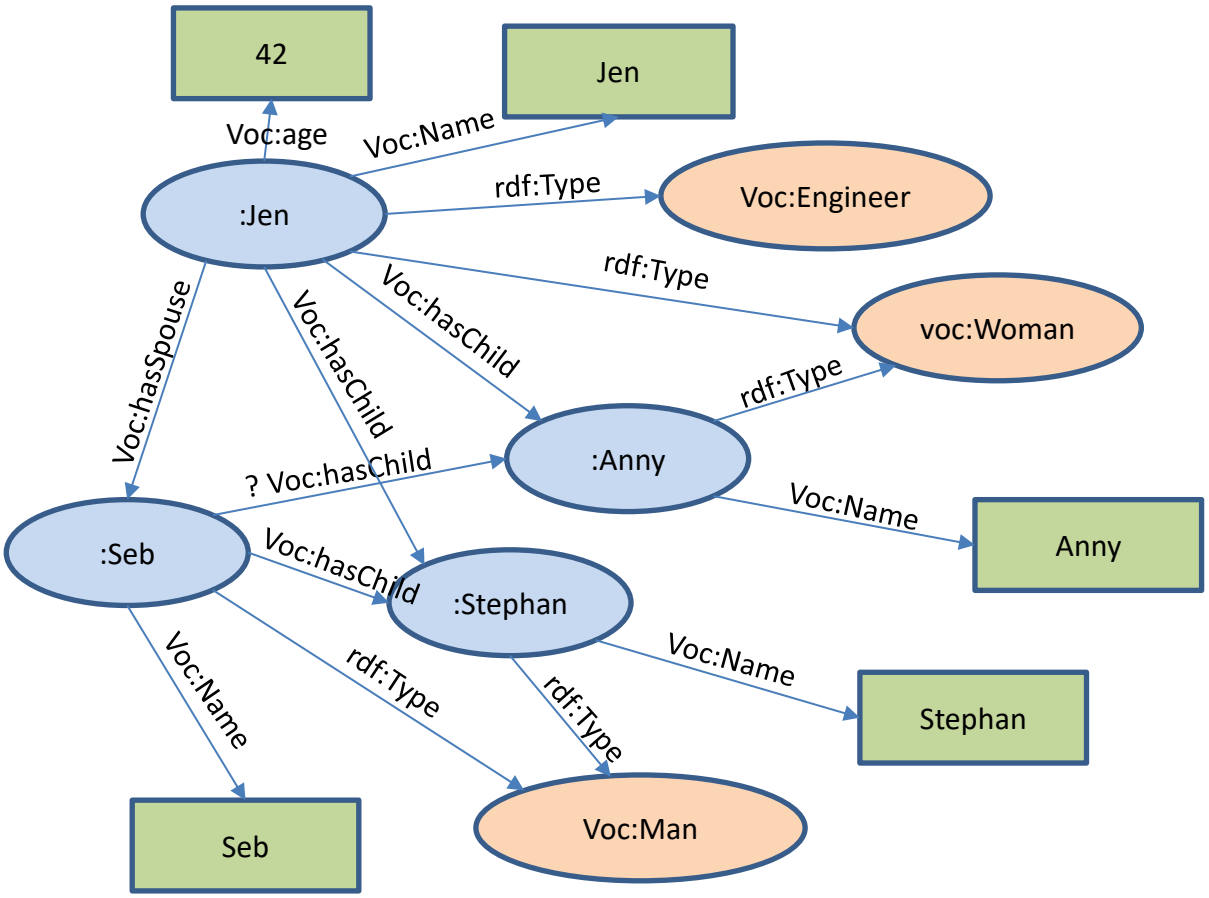
https://demo.dbpedia-spotlight.org/

Questions from the course on RDF.

Q2.0 Fill the blanks.

“Jen is an engineer woman, 42-year-old, married to Seb who is a man with whom she had two children: Anny who is a woman and Steffen who is a man". For each person we also explicitly specify the name.

To fill the blanks we use the values: :Seb, :Steffen, voc:name, voc:hasChild, voc:age, voc:hasSpouse, rdf:type, voc:Engineer, voc:Man, "Jen", "Seb", "Anny", "Steffen"
For each person we also explicitly specify the name



Q2.1 What is missing to say that “doc.html as for authors Catherine and Fabien and is about Music and Piano” ?

<http://inria.fr/rr/doc.html> <http://inria.fr/schema#author>
<http://ns.inria.fr/fabien.gandon#me> .

<http://inria.fr/rr/doc.html> <http://inria.fr/schema#theme> "Music" .

Answer:

<http://inria.fr/rr/doc.html> <http://inria.fr/schema#author>
<http://ns.inria.fr/catherine.faron#me> .
<http://inria.fr/rr/doc.html> <http://inria.fr/schema#theme> "Piano" .

Q2.2 Fill the blanks (N3/Turtle)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://www.unice.fr/voc#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://www.unice.fr/data#Jen> a voc:Engineer AAA BBB ;
   voc:age "42"^^xsd:string ;
   voc:hasChild <http://www.unice.fr/data#Anny>, <CCC>;
   voc:hasSpouse <http://www.unice.fr/data#Seb> ;
   voc:name "Jen" .
<http://www.unice.fr/data#Seb> DDD voc:Man ;
   voc:hasChild <http://www.unice.fr/data#Anny>,
      <http://www.unice.fr/data#Steffen> ;
   voc:name "Seb" .
<http://www.unice.fr/data#Anny> a voc:Woman ;
   voc:name "Anny" .
<EEE> a FFF ;
    GGG HHH .
```

Answer:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://www.unice.fr/voc#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://www.unice.fr/data#Jen> a voc:Engineer , voc:Woman ;
   voc:age "42"^^xsd:string ;
   voc:hasChild <http://www.unice.fr/data#Anny>, < http://www.unice.fr/data#Stephan>;
   voc:hasSpouse <http://www.unice.fr/data#Seb> ;
   voc:name "Jen" .
<http://www.unice.fr/data#Seb> rdf:type voc:Man ;
   voc:hasChild <http://www.unice.fr/data#Anny>,
      <http://www.unice.fr/data#Steffen> ;
   voc:name "Seb" .
<http://www.unice.fr/data#Anny> a voc:Woman ;
   voc:name "Anny" .
<http://www.unice.fr/data#Stephan> a voc:Man ;
    voc:name "Stephan".
```

Q2.3 Fill the blanks (RDF/XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [   <!ENTITY vocab "http://www.unice.fr/voc">    <!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#"> ]>
```

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:voc="&vocab;#"
xml:base="http://www.unice.fr/data">
 <voc:Woman rdf:about="#Jen">
   <voc:name>Jen</voc:name>
   <voc:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42 </voc:age>
   < voc:hasSpouse rdf:resource="#Seb"></voc:hasSpouse >
   <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
   <voc:hasChild>
    <rdf:Description rdf:about="#Anny">
     <voc:name>Anny</voc:name>
     <rdf:type rdf:resource="&vocab;#Woman"></rdf:type>
    </rdf:Description>
   </voc:hasChild>
   <rdf:type rdf:resource="&vocab;#Engineer"></rdf:type>
 </voc:Woman>
 <voc:Man rdf:about="#Seb">
   <voc:name>Seb</voc:name>
   <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
   <voc:hasChild rdf:resource="#Anny"></voc:hasChild>
 </voc:Man>
 <voc:Man rdf:about="#Steffen">
   <voc:name>Steffen</voc:name>
 </voc:Man>
</rdf:RDF>
```

Q2.4 Visit me please

Get the RDF data from: http://ns.inria.fr/fabien.gandon#me

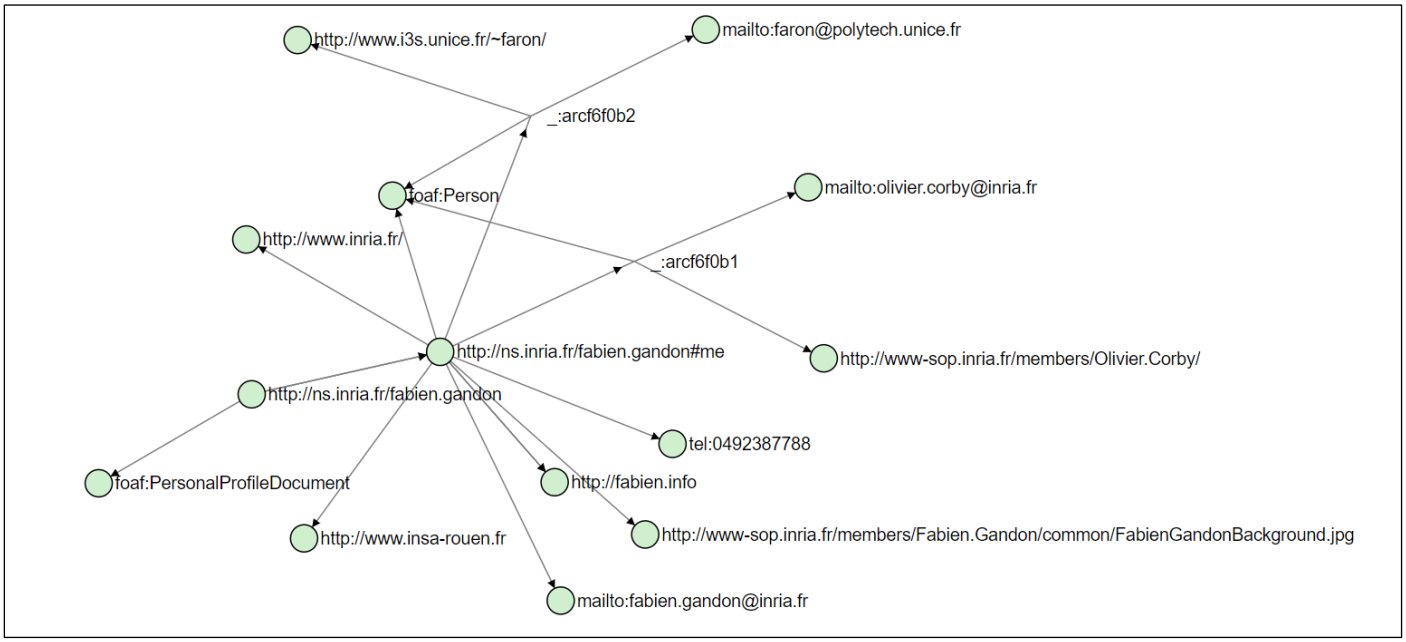
1. Get the RDF data from: http://ns.inria.fr/fabien.gandon#me
2. What is the syntax used?
3. Validate it and see the graph:
   http://www.w3.org/RDF/Validator/
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdf-translator.appspot.com/
5. Visualize it also with:
   http://cltl.nl/visualrdf/
   http://www.easyrdf.org/converter (PNG, SVG)
   https://www.ldf.fi/service/rdf-grapher
6. Adapt to your data and do it again

Turtle/N3 file:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://ns.inria.fr/fabien.gandon>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://ns.inria.fr/fabien.gandon#me> ;
  foaf:primaryTopic <http://ns.inria.fr/fabien.gandon#me> .

<http://ns.inria.fr/fabien.gandon#me>
  a foaf:Person ;
```

```
  foaf:name "Fabien Gandon" ;
  foaf:title "Dr" ;
  foaf:givenname "Fabien" ;
  foaf:family_name "Gandon" ;
  foaf:nick "Bafien" ;
  foaf:mbox <mailto:fabien.gandon@inria.fr> ;
  foaf:homepage <http://fabien.info> ;
  foaf:depiction <http://www-
sop.inria.fr/members/Fabien.Gandon/common/FabienGandonBackground.jpg> ;
  foaf:phone <tel:0492387788> ;
  foaf:workplaceHomepage <http://www.inria.fr/> ;
  foaf:workInfoHomepage <http://fabien.info> ;
  foaf:schoolHomepage <http://www.insa-rouen.fr> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Olivier Corby" ;
    foaf:mbox <mailto:olivier.corby@inria.fr> ;
    rdfs:seeAlso <http://www-sop.inria.fr/members/Olivier.Corby/>
  ], [
    a foaf:Person ;
    foaf:name "Catherine Faron-Zucker" ;
    foaf:mbox <mailto:faron@polytech.unice.fr> ;
    rdfs:seeAlso <http://www.i3s.unice.fr/~faron/>
  ] .
```

Visualization:



Adapt the Turtle file:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://personaldata.fr/lan.le>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://personaldata.fr/lan.le#me> ;
  foaf:primaryTopic <http://personaldata.fr/lan.le#me> .

<http://personaldata.fr/lan.le#me>
  a foaf:Person ;
  foaf:name "Lan LE" ;
  foaf:title "Miss" ;
  foaf:givenname "Lan" ;
  foaf:family_name "LE" ;
```

```
  foaf:nick "Havan" ;
  foaf:mbox <mailto:le.lan@edu.dsti.institute> ;
  foaf:homepage <http://personaldata.fr> ;
  foaf:phone <tel:0123456789> ;
  foaf:workplaceHomepage <http://www.personaldata.fr/> ;
  foaf:workInfoHomepage <http://www.personaldata.fr> ;
  foaf:schoolHomepage <http://www.personaldata.fr> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Duy Nguyen" ;
    foaf:mbox <mailto: duy.nguyen@edu.dsti.institute> ;
    rdfs:seeAlso < https://github.com/nguyenkduywork>
  ], [
    a foaf:Person ;
    foaf:name "Duy Khanh Nguyen" ;
    foaf:mbox <mailto: duy-khanh.nguyen@edu.dsti.institute> ;
    rdfs:seeAlso < https://github.com/dkhanhnguyen>
  ] .
```

Q2.5 what is the meaning of this RDF? What is this description saying?

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:exs="http://example.org/schema#">
 <rdf:Description rdf:about="http://example.org/doc.html">
  <rdf:type rdf:resource="http://example.org/schema#Report"/>
  <exs:theme rdf:resource="http://example.org#Music"/>
  <exs:theme rdf:resource="http://example.org#Danse"/>
  <exs:nbPages rdf:datatype="http://www.w3.org/2001/XMLSchema#int">73</exs:nbPages>
 </rdf:Description>
</rdf:RDF>
```

Answer:

This RDF describes a document located at "http://example.org/doc.html" using the XML syntax.

- The document is of type "Report";
- 2 themes are associated with this report: "Music" & "Danse";
- The report has 73 pages.

Q2.6 Visit to Victor Hugo

1. See HTML data from:
   http://id.loc.gov/authorities/names/n79091479.html
2. Get RDF data from:
   http://id.loc.gov/authorities/names/n79091479.rdf
3. What is the syntax?
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/
5. Any remark about the values of the properties of Victor Hugo?

Answer:

The used syntax is XML.

The error is that we don't have any language mentioned.

```
ns0:hasVariant [
  a ns0:PersonalName, ns0:Variant ;
  ns0:variantLabel "Hiwkō, Vik‘t‘or, 1802-1885" ;
  ns0:elementList (
   _:genid6
   _:genid8
 )
], [
  a ns0:PersonalName, ns0:Variant ;
  ns0:variantLabel "Hījū, Fīktūr, 1802-1885" ;
  ns0:elementList (
   _:genid11
   _:genid13
 )
], [
  a ns0:PersonalName, ns0:Variant ;
  ns0:variantLabel "Giūgo, Viktor, 1802-1885" ;
  ns0:elementList (
   _:genid16
   _:genid18
 )
], [
  a ns0:PersonalName, ns0:Variant ;
  ns0:variantLabel "Hsiao-o, 1802-1885" ;
  ns0:elementList (
```

Q2.7 What is the syntax of the following RDF statement? What does it mean?

@prefix dcterms: <http://purl.org/dc/terms/>.

GRAPH <http://inria.fr/data/algebra>

{

<http://inria.fr/rr/doc.html>

dcterms:subject

<http://data.bnf.fr/ark:/12148/cb121105993> .

}

Answer:

This RDF uses TriG syntax (because we are using "GRAPH ") with the first line is the declaration and the rest is the triple statement.

This triple statement specifies details of a graph:

- The graph is named http://inria.fr/data/algebra;
- The graph's resource is identified by the URI "http://inria.fr/rr/doc.html"
- The graph has the property is "dcterms:subject" with the value

Q2.8 Visit Leukocyte surface antigen CD53

1. See HTML data from:
   http://www.uniprot.org/uniprot/Q61451
2. Get RDF data from:
   http://www.uniprot.org/uniprot/Q61451.rdf
3. What is the syntax?
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter

http://rdfvalidator.mybluemix.net/
http://rdf-translator.appspot.com/

5. Any remark about the structure of the data?

The used syntax is XML.
The error is : Reification is used instead of named graphs

```
<http://purl.uniprot.org/uniprot/#_kb.Q61451_up.isolatedFrom_tissue.449>
  a rdf:Statement ;
  rdf:subject <http://purl.uniprot.org/uniprot/Q61451> ;
  rdf:predicate ns0:isolatedFrom ;
  rdf:object <http://purl.uniprot.org/tissues/449> ;
  ns0:citation <http://purl.uniprot.org/citations/15489334> .

<http://purl.uniprot.org/uniprot/#_kb.Q61451_up.isolatedFrom_tissue.595>
  a rdf:Statement ;
  rdf:subject <http://purl.uniprot.org/uniprot/Q61451> ;
  rdf:predicate ns0:isolatedFrom ;
  rdf:object <http://purl.uniprot.org/tissues/595> ;
  ns0:citation <http://purl.uniprot.org/citations/15489334> .

<http://purl.uniprot.org/uniprot/#_kb.Q61451_up.annotation_C41195D4B5E5CBEB>
  a rdf:Statement ;
  rdf:subject <http://purl.uniprot.org/uniprot/Q61451> ;
  rdf:predicate ns0:annotation ;
  rdf:object <http://purl.uniprot.org/uniprot/Q61451#SIPB6CC6D00F2446EB3> ;
  ns0:attribution <http://purl.uniprot.org/uniprot/Q61451#attribution-89AC1B682EEB440D50C4AEBB24FCA860> .
```

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 . }
```

	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>

Dataset

Data:
/data/tutorial/human1.rdf
/data/tutorial/human2.rdf
/data/tutorial/pragma.ttl
Query:
/data/tutorial/tutorial.ttl

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 ?x ?p ?v WHERE {
  ?x rdfs:label "Paris"@fr .
  ?x ?p ?v .
}
LIMIT 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.w3.org/2003/01/geo/wgs84_pos#SpatialThing>
:Paris	rdf:type	dbpedia:ontology/City
:Paris	rdf:type	dbpedia:ontology/Settlement
:Paris	rdf:type	<http://schema.org/Place>
:Paris	rdf:type	<http://www.wikidata.org/entity/Q486972>
:Paris	rdf:type	dbpedia:class/yago/WikicatArchaeologicalSitesInFrance
:Paris	rdf:type	dbpedia:ontology/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/>

PREFIX ps: <http://www.wikidata.org/prop/statement/>

Find q-name of the property “given name”

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

The query retrieves the USA presidents’ name, with Q30 = United States of America and P6 = Head of Government. Results: https://w.wiki/6Psi

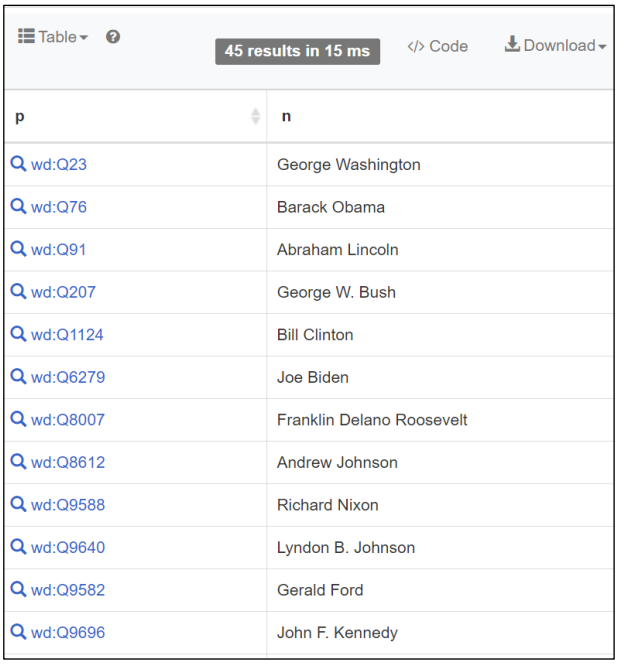


Table | 45 results in 15 ms | Code | Download

p	n
wd:Q23	George Washington
wd:Q76	Barack Obama
wd:Q91	Abraham Lincoln
wd:Q207	George W. Bush
wd:Q1124	Bill Clinton
wd:Q6279	Joe Biden
wd:Q8007	Franklin Delano Roosevelt
wd:Q8612	Andrew Johnson
wd:Q9588	Richard Nixon
wd:Q9640	Lyndon B. Johnson
wd:Q9582	Gerald Ford
wd:Q9696	John F. Kennedy

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’

```
SELECT ?age
WHERE { ?x c:name ‘Fabien’ ; ?x c:age ?age }
```

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

```
SELECT ?name
WHERE { ?x c:name ?name ; ?x c:age ?age . FILTER(?age < 50) }
```

3. Find properties and their values for resources whose name is ‘Fabien’ and whose age is less than 50

```
SELECT ?p ?y
WHERE {
  ?x c:name ‘Fabien’ ; c:age ?age ; ?p ?y .
  FILTER(?age < 50)
```

```
}
```

4.Find other names of resources with name ‘Fabien’, at least, a second name

```
SELECT ?name
WHERE {
  ?x c:name ‘Fabien’ , ?name . FILTER(?name != ‘Fabien’) }
```

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

```
SELECT ?x
WHERE { ?x ?p1 ?y ; ?p2 ?y . FILTER(?p1 != ?p2) }
```

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

```
SELECT ?x
WHERE {?x ?p ?y , ?y2 . FILTER(?y1 != ?y2) }
```

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 }
}
```

Yes, because we do not use the same variable so the minus will be ignored.

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

```
SELECT ?x
WHERE {
?x c:memberOf ?org .
FILTER (! EXISTS { ?x c:author ?doc} )
}
```

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))
```

This query retrieves a list of authors and the number of documents they wrote, sorted in descending order by the count of documents authored (i.e., the most prolific author appears first).

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

The correct expression is the first one: " ?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 }
}
```

This query retrieves the names of x's friend. If the name variable is bound, then give this name. Otherwise, print "John Doe" if no name is bound.

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
```

This query retrieves the estimation of x's age based on the average age of x's friends.

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

1. SELECT * WHERE {{?x c:study c:Informatics } UNION {?x c:study c:Mathematics}}

2. SELECT * WHERE {{?x c:study c:Informatics } UNION {?x c:study c:Mathematics}
   OPTIONAL {?x c:name ?name} }

3. SELECT * WHERE {
   {?x c:study c:Informatics } UNION {?x c:study c:Mathematics}
   OPTIONAL {graph ?g {?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}  }
```

?x ?p ?y is calculated on g1

graph ?g { ?y ?q ?z } is calculated on g2

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

INSERT {?x rdfs:label ?n}

DELETE {?x foaf:name ?n }

WHERE {?x foaf:name ?n}

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 ?a (xsd:integer(?a) as ?i)
 where {
   ?x ex:age ?a
   filter(datatype(?a) = xsd:string)
 }
}
```

The query locates all the 'age' properties in string format, converts them into integer format, deletes the identified triple, and inserts a new one with the 'age' property converted into an integer.

This query makes sure all 'age' properties are 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 {…} …

We can use the CONSTRUCT clause to obtain results as RDF triples that follow a specific pattern.

Q3.18 What is the difference between these two queries?

```
prefix ex: <http://example.org/>
insert { ?x a ex:Parent }
where  { ?x ex:hasChild  ?y }


prefix ex: <http://example.org/>
construct { ?x a ex:Parent }
where  { ?x ex:hasChild  ?y }
```

Query 1: we modify the base with INSERT clause

Query 2: we don’t modify the base with CONSTRUCT clause

Questions from the course on Ontologies.

Q4.0 Choose among the following assertions one or more you consider to be true:

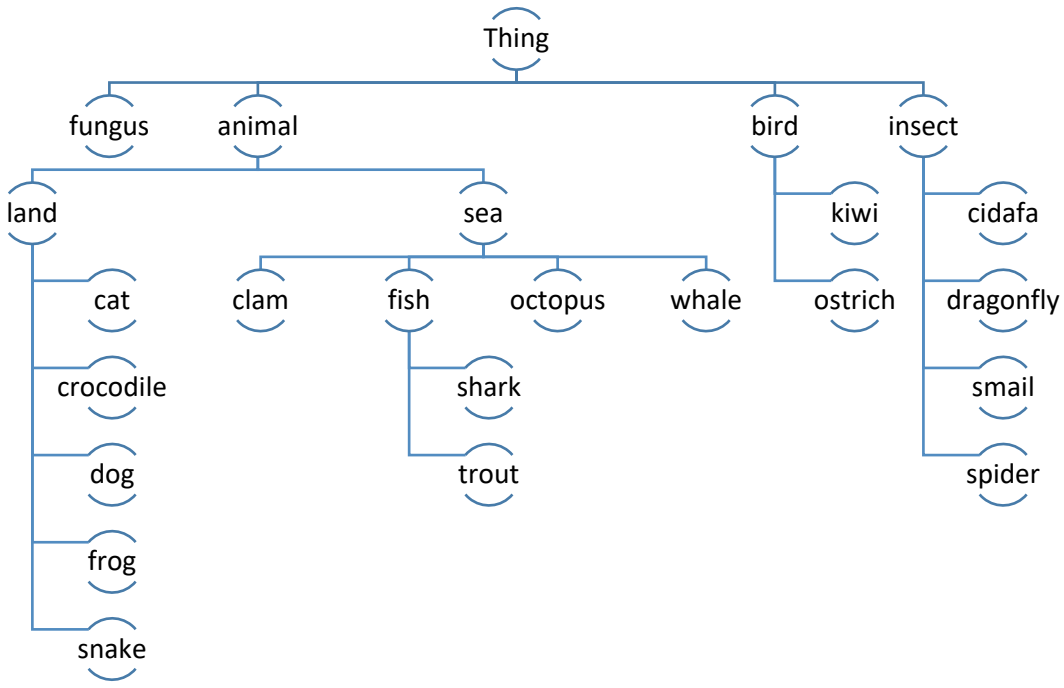
- ☐ an ontology is necessarily formalized in first-order logic
- ☐ **an ontology may allow inferences on data that uses it**
- ☐ **conceptual graphs can represent an ontology**
- ☐ **a shared ontology promotes interoperability**
- ☐ **description logics can represent an ontology**

an ontology may allow inferences on data that uses it
conceptual graphs can represent an ontology
a shared ontology promotes interoperability
description logics can represent an ontology

Q4.1 work alone for 10 minutes

From real example of ontology engineering: you are designing a system to assist (data) management of species presented in a museum of natural sciences. In particular you need to organize at least the species below. Structure them and add categories as needed.

Species: animal, bird, cat, cicada, clam, crocodile, dog, dragonfly, fish, frog, fungus, insect, kiwi, octopus, ostrich, shark, snail, snake, spider, thing, trout, whale



Questions from the course on RDFS

Q4.2 RDFS contains primitives to (several answers possible)…

- ☐ **describe classes of resources**
- ☐ describe formulas of calculation for values of properties
- ☐ **describe types of properties of resources**
- ☐ **document definitions in natural language**
- ☐ sign and authenticate the authors of the definitions of classes and properties

describe classes of resources
describe types of properties of resources
document definitions in natural language

Q4.3. What is defined and derived from these definitions?

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <http://inria.fr/devices#>
:Phone rdfs:subClassOf  :Device .
:Computer rdfs:subClassOf  :Device .
:Smartphone rdfs:subClassOf :Computer .
:Smartphone rdfs:subClassOf :Phone .
```

We have Smartphone is subclass of Computer and also is subclass of Phone. Phone and Computer are subclass of Device.

Then Smartphone is subclass of Device by transitivity.

Q4.4. What is defined and derived from these definitions?

```
@prefix rdfs:  < http://www.w3.org/2000/01/rdf-schema# >
@prefix : <http://inria.fr/member#>
:employeeOf rdfs:subPropertyOf  :proRelationWith .
:hasControlOver rdfs:subPropertyOf :proRelationWith .
:isShareholderOf rdfs:subPropertyOf :hasControlOver .
:isCEOof rdfs:subPropertyOf :employeeOf, :hasControlOver .
```

We have

":employeeOf" and ":hasControlOver" are sub-properties of ":proRelationWith"

":isShareholderOf" is a sub-property of ":hasControlOver"

":isCEOof" is a sub-property of both ":employeeOf" and ":hasControlOver"

Then

":isCEOof" is sub-property of ":proRelationWith"

and ":isShareholderOf" is also a sub-property of ":proRelationWith"

Q4.5. Download the ontology Schema.org founded by Google, Microsoft, Yahoo and Yandex: https://schema.org/version/latest/schemaorg-current-https.ttl

Find the Class schema:AboutPage and identify its super-class

Can a document be of type schema:abstract ?

3 The super-class of schema:AboutPage is schema:WebPage

4 A document cannot be of type schema:abstract because an abstract is a property that can be attached to a document to give it a short summary.

```
abstract a rdf:Property ;
    rdfs:label "abstract" ;
    rdfs:comment "An abstract is a short description that summarizes a
[[CreativeWork]]."
```

Q4.6. What can be said about the types of the resources that will be linked by the properties defined below?

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <http://inria.fr/humans#>
:driverOf  rdfs:subPropertyOf  :isControling .
:piloteOf  rdfs:subPropertyOf  :isControling .
:isControling  rdfs:domain  :Human ;  rdfs:range :Object .
:driverOf rdfs:range  :Car .
:piloteOf rdfs:domain  :Adult ; rdfs:range  :Plane .
```

All of them will apply.

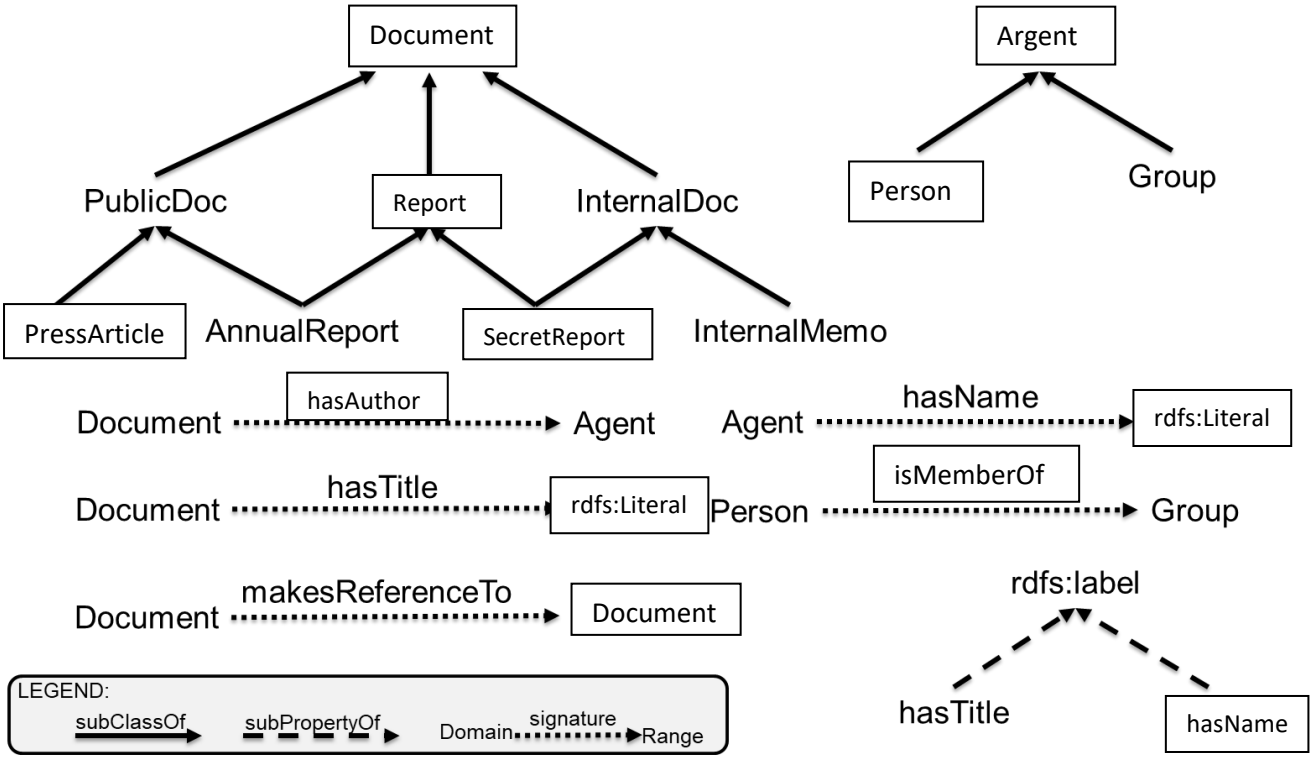
Q4.7. What could we add to this schema (several answers are possible)?

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@ prefix :    <http://ns.inria.fr/humans/schema#>
:p1  a rdf:Property ;  rdfs:label  "age"@en .
:c1  a rdfs:Class ; rdfs:comment "a human being"@en .
```

- ☐ :p1 rdfs:label "firstname"@en, "prénom"@fr .
- ☐ :c1 rdfs:comment "un être humain"@en .
- ☐ **:c1 rdfs:label "person"@en, "personne"@fr .**
- ☐ **:p1 rdfs:label "âge"@fr .**
- ☐ :c1 rdfs:label "woman"@en .
- ☐ **:c1 rdfs:label "persona"@es .**
- ☐ **:p1 rdfs:comment "the length of time something has existed."@en .**

:c1 rdfs:label "person"@en, "personne"@fr .
:p1 rdfs:label "âge"@fr .
:c1 rdfs:label "persona"@es .
:p1 rdfs:comment "the length of time something has existed."@en .

Q4.8. (a) Fill the blanks with: Document, PublicDoc, PressArticle, Report, AnnualReport, InternalDoc, SecretReport, InternalMemo, Agent, Person, Group, hasTitle, hasAuthor, makesReferenceTo, hasName, isMemberOf + rdf / rdfs primitives.
(b) Write it in RDFS and validate the RDF.



Questions from the course on OWL.

Q5.1 What is asserted and what can we deduce?

ex:Man owl:intersectionOf (ex:Male ex:Human) .
ex:Woman owl:intersectionOf (ex:Female ex:Human) .
ex:Human owl:unionOf (ex:Man ex:Woman) .
ex:Jane a ex:Human .
ex:John a ex:Man .
ex:James a ex:Male .
ex:Jane a ex:Female .

“John is Man”, then he is in the intersection of Male of Human

➔ Deduction: He is a Male and also a Human.

Since “Jane is Female” and “Jane is Human”, then she is in the intersection of Female and Human

➔ Deduction: She is Woman

“James is Male”: we cannot deduce anything more about James.

Q5.2 What are we defining and inferring?

```
@prefix ex: <http://example.org/>

ex:GrandFather rdfs:subClassOf [
  a owl:Class ;
  owl:intersectionOf ( ex:Parent ex:Man )
] .

ex:Jim a ex:Man, ex:Parent .
ex:Jack a ex:GrandFather .
```

We have: Every time we have GrandFather, we’ll have Parent and Man

Then:

- Jack is GrandFather, so he is also a Parent and a Man
- We can not deduct about Jim.

Q5.3 What is asserted and what can we deduce?

```
ex:hasSpouse a owl:SymmetricProperty .
ex:hasChild  owl:inverseOf ex:hasParent .
ex:hasParent rdfs:subPropertyOf ex:hasAncestor .
ex:hasAncestor a owl:TransitiveProperty .

ex:Jim  ex:hasChild  ex:Jane .
ex:Jane ex:hasSpouse ex:John .
ex:Jim  ex:hasParent ex:James .
```

Jane hasParent Jim

John hasSpouse Jane

Jane has Ancestor James

Jim has Ancestor James

James hasChild Jim

Q5.4 What is asserted and what can we deduce?

```
ex:Human owl:equivalentClass foaf:Person .
foaf:name owl:equivalentProperty ex:name .

ex:JimmyPage a ex:Human ;
        owl:sameAs ex:JamesPatrickPage .
ex:JimmyHendrix owl:differentFrom ex:JimmyPage .
```

What is asserted :

- Class Human is equivalent to the class Person.
- In the foaf ontology, there is the property name is equivalent to name in the ex ontology.

We can deduce:

- JimmyPage a foaf:Person
- JamesPatrickPage a foaf:Person
- JamesPatrickPage a ex:Human
- "ex:JimmyHendrix owl:differentFrom ex:JimmyPage" does not allow us to deduce anything

Q5.5 What are we defining and inferring?

```
ex:UnhappyPerson owl:equivalentClass [
 a owl:Class ;
 owl:intersectionOf (
  ex:Person
  [ a owl:Class ; owl:complementOf ex:Happy ]
 )
] .
```

To be an unhappy Person, you need to be a Person and in the complement of happy.

Q5.6 What is asserted and what can we deduce?

```
ex:Human rdfs:subClassOf
 [ a owl:Restriction ;
  owl:onProperty ex:hasParent ;
  owl:allValuesFrom ex:Human ] .
ex:Tom a ex:Human .
ex:Tom ex:hasParent ex:James, ex:Jane.
```

What is asserted:

- Human is the sub class of Restriction, onProperty of hasParent.
- All the values from this property must come from Human.

We can deduce that:

- Since Tom is Human and Tom has James and Jane as parent, so James and Jane must be Human.

Q5.7 What are we defining and inferring?

```
@prefix ex: <http://example.org/>
ex:PersonList rdfs:subClassOf
 [
  a owl:Restriction ;
  owl:onProperty rdf:first ;
  owl:allValuesFrom ex:Person
 ] ,  [
  a owl:Restriction ;
  owl:onProperty rdf:rest ;
  owl:allValuesFrom ex:PersonList
 ] .

ex:value rdfs:range ex:PersonList .
ex:abc ex:value (ex:a ex:b ex:c) .
```

We have: In a list of person, the first element must be person and the rest must be a person list.

Then: (ex:a ex:b ex:c) becomes person list and a & b & c become automatically a person.

Q5.8 What are we defining and inferring?

```
@prefix ex: <http://example.org/>
ex:Human rdfs:subClassOf [
 owl:intersectionOf (
  [
   a owl:Restriction ;
    owl:onProperty ex:hasBiologicalFather ;
    owl:maxCardinality 1
  ] , [
   a owl:Restriction ;
    owl:onProperty ex:hasBiologicalMother ;
    owl:maxCardinality 1
  ] )
] .
ex:Jane a ex:Human ;
     ex:hasBiologicalFather ex:James , ex:Jhon .
```

The system will conclude ex:James and ex:Jhon are the same person

Q5.9 Visit the financial Industry Business Ontology (FIBO)
https://spec.edmcouncil.org/fibo/
with a lot of companies using it: https://edmcouncil.org/page/listofmembersreview

Negociate the turtle version of the part about governments:
https://spec.edmcouncil.org/fibo/ontology/BE/GovernmentEntities/GovernmentEntities/

Explain the formal definitions of Government, NationalGovernment, FederalGovernment, and isJurisdictionOf.

Negotiate the turtle version:

```
curl -L -H "Accept: text/turtle" -o govt_entities.ttl
https://spec.edmcouncil.org/fibo/ontology/BE/GovernmentEntities/GovernmentEntities/ -k
```

Explain the formal definitions:

- *Government :*

Government is a class and is a subclass of FunctionalEntity. Every government is also a functional entity and must have a value of the geopolitical entity type, indicating that it governs a geopolitical entity. The government may have Parts which must be BranchOfGovernment, GovernmentAgency, or GovernmentDepartment.

- *NationalGovernment :*

The national government is a class, and it has a label and a definition. The national government is a subclass of Government with a restriction that it must have a relation “governs”, meaning it must have at least one value from country (the national government is ruling at least one country).

- *FederalGovernment :*

The federal government is a type of national government, which means that a federal government is also a national government and a form of governance. The property of governance on the class of 'FederalCapitalArea' has a minimum qualified cardinality of 0, which means that there can be zero or more FederalCapitalAreas. The 'governs' property must have a minimum qualified cardinality of 2 for the 'FederalState' class.

A federal government must govern at least two states in order to be considered a federation. If only one state is governed, then it would be considered a national government rather than a federal government.

- *isJurisdictionOf :*

"Jurisdiction" is a property that connects Jurisdiction to anything (because no specific range is defined). It has an inverse property called ‘hasJurisdiction’.

Questions from the course on Vocabularies.

Q6.0 Visit schema.org and

1. Check the documentation of schema:Accommodation
   https://schema.org/Accommodation
   and identify the more specific types in the page
2. Download the current version in Turtle (.ttl):
   https://schema.org/docs/developers.html
   extract the Turtle definition of schema:Recipe
   and find the name of the class it inherits from

1. Identification of more specific types:

   Apartment
   CampingPitch
   House
   Room
   Suite

2. Turtle definition of schema:Recipe is: *"A recipe. For dietary restrictions covered by the recipe, a few common restrictions are enumerated via [[suitableForDiet]]. The [[keywords]] property can also be used to add more detail."*

   Name of the class it inherits from: "schema:HowTo"

```
schema:Recipe a rdfs:Class ;
    rdfs:label "Recipe" ;
    rdfs:comment "A recipe. For dietary restrictions covered by the recipe, a few
common restrictions are enumerated via [[suitableForDiet]]. The [[keywords]]
property can also be used to add more detail." ;
    rdfs:subClassOf schema:HowTo .
```

Q6.1 What do you think of the annotation?

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
<#B-A-Ba> a skos:Concept ;
 skos:prefLabel   "B.A.-BA"@en , "b.a.-ba"@en ;
 skos:altLabel    "B-A-BA"@en ,  "b-a-ba"@en ;
 skos:hiddenLabel "BABA"@en , "baba"@en .
```

B-A-Ba is skos:Concept which has 2 preferred label (-> error – because we can have only 1 Label in @en), 2 altLabel and hiddenLabel.

Q6.2 practice:

1. Using the site prefix.cc find back the namespace usually associated to the SKOS prefix
2. Access the URL of the namespace and find the RDF source file defining the SKOS vocabulary
3. Find the definition of the property narrowMatch and give all the relations it has with other properties

http://www.w3.org/TR/skos-reference/skos.rdf

skos:narrowMatch is used to state a hierarchical mapping link between two conceptual resources in different concept schemes.

It is a sub property of mappintrelation and narrower.

It is the inverse property of broadMatch.

```
<owl:ObjectProperty rdf:about="http://www.w3.org/2004/02/skos/core#narrowMatch">
  <rdfs:label xml:lang="en">has narrower match</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2004/02/skos/core"/>
  <skos:definition xml:lang="en">skos:narrowMatch is used to state a hierarchical mapping link between two conceptual resources in differ
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2004/02/skos/core#mappingRelation"/>
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2004/02/skos/core#narrower"/>
  <owl:inverseOf rdf:resource="http://www.w3.org/2004/02/skos/core#broadMatch"/>
```

Q6.3 practice:

1. Find and open the source file of Dublin Core Terms: https://dublincore.org/schemas/rdfs/
   Look at the definition of the class FileFormat and find the class it inherits from.
2. Choose your preferred book on Amazon, Fnac, etc. and describe it in an RDF annotation using as many DC primitives as necessary.
3. Add the most restrictive CC license to your preferred book ; is this license appropriate?

1. <rdfs:subClassOf rdf:resource="http://purl.org/dc/terms/**MediaType**"/>

```
    <rdfs:label xml:lang="en">File Format</rdfs:label>
  </rdf:Description>
  <rdf:Description rdf:about="http://purl.org/dc/terms/FileFormat">
    <rdfs:comment xml:lang="en">A digital resource format.</rdfs:comment>
  </rdf:Description>
  <rdf:Description rdf:about="http://purl.org/dc/terms/FileFormat">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://purl.org/dc/terms/FileFormat">
    <rdfs:subClassOf rdf:resource="http://purl.org/dc/terms/MediaType"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://purl.org/dc/terms/FileFormat">
    <dcterms:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2008-01-14</dcterms:issued>
```

2 & 3.

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ex: <http://example.com/>.
@prefix cc: <http://creativecommons.org/ns#>.

<http://www.amazon.com/dp/2070367233>
        cc:requiresPermission [
                cc:Reproduction;
                cc:Distribution;
                cc:DerivativeWorks;
```

```
        ];
        dc:title "L'insoutenable Légèreté De L'être";
        dc:creator "Milan Kundera";
        dc:language "fr";
        dcterms:issued "1984";
        dc:publisher "Gallimard";
        dc:description "L'insoutenable Légèreté De L'être is a novel by Milan Kundera, published in 1984 by Gallimard.";
```

Q6.4 practice:

1. Get the source of the FoaF schema: http://xmlns.com/foaf/spec/index.rdf
2. Find the property weblog
3. What are the types of this property?
4. Does it inherit from other properties?
5. What is its signature?

Types of this property: owl:InverseFunctionalProperty

It is subPropertyOf foaf:page

rdf:resource="http://xmlns.com/foaf/0.1/page"/>

Signature : Domain: Agent & Range: Document

Property: foaf:weblog

weblog - A weblog of some thing (whether person, group, company etc.).
Status: testing
OWL Type: An InverseFunctionalProperty (uniquely identifying property)
Domain: foaf:Agent
Range: foaf:Document

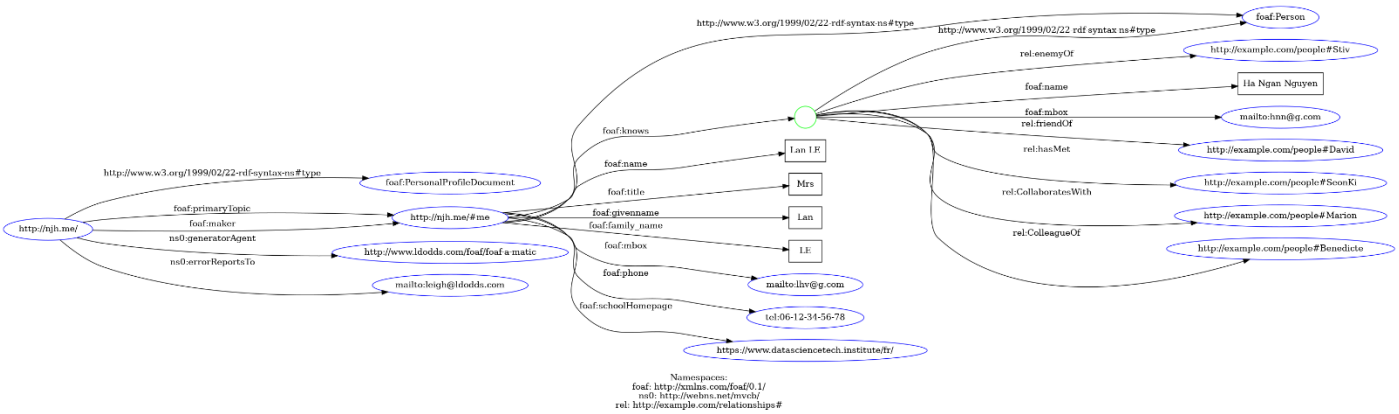
The foaf:weblog property relates a foaf:Agent to a weblog of that agent.

Q6.5 practice:

1. Find the FOAF-a-Matic web page
2. Use this tool to generate your FOAF profile in RDF/XML
3. Translate it into Turtle, save and give the result in your answers.
4. Add five specific relationships to your FOAF file using RELATIONSHIPS: http://purl.org/vocab/relationship/

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ns0: <http://webns.net/mvcb/> .
@prefix rel: <http://example.com/relationships#> .

<http://njh.me/>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://njh.me/#me> ;
  foaf:primaryTopic <http://njh.me/#me> ;
```

```
  ns0:generatorAgent <http://www.ldodds.com/foaf/foaf-a-matic> ;
  ns0:errorReportsTo <mailto:leigh@ldodds.com> .

<http://njh.me/#me>
  a foaf:Person ;
  foaf:name "Lan LE" ;
  foaf:title "Mrs" ;
  foaf:givenname "Lan" ;
  foaf:family_name "LE" ;
  foaf:mbox <mailto:lhv@g.com> ;
  foaf:phone <tel:06-12-34-56-78> ;
  foaf:schoolHomepage <https://www.datasciencetech.institute/fr/> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Ha Ngan Nguyen" ;
    foaf:mbox <mailto:hnn@g.com> ;
    rel:friendOf <http://example.com/people#David> ;
    rel:hasMet <http://example.com/people#SeonKi> ;
    rel:CollaboratesWith <http://example.com/people#Marion> ;
    rel:ColleagueOf <http://example.com/people#Benedicte> ;
    rel:enemyOf <http://example.com/people#Stiv>
  ] .
```



Q6.6 What does this mean?

```
:BioRDF2DBLP a void:Linkset;
        void:target :BioRDF;
        void:target :DBLP;
        void:linkPredicate skos:exactMatch;
        void:triples 8936 .
```

BioRDF2DBLP is linking 2 datasets (BioRDF and DBLP) and listing triples in the two datasets by using the predicate exactMatch. We have 8936 of triples / 'exactMatch' relations between these 2 datasets.

Q6.7 practice:

1. Connect to the Void Store SPARQL endpoint: http://lod.openlinksw.com/sparql/
2. Find the void:Dataset with a property dct:subject and retrieve their URI and subjects.
3. Find the void:Dataset with a dct:subject "covid19".

```
PREFIX dct: <http://purl.org/dc/terms/>
SELECT * WHERE { ?s dct:subject ?s } LIMIT 1000
```

```
SELECT * WHERE { ?x a void:Dataset } LIMIT 1000
```

```
PREFIX dct: <http://purl.org/dc/terms/>
SELECT * WHERE { ?x dct:subject "covid19" ; ?p ?v } limit 1000
```

Q6.8 What does this mean?

```
ex:plot prov:used ex:stats1998 .
ex:bar-chart prov:wasGeneratedBy ex:plot .
ex:stats1998 a dcat:Distribution ;
        dcat:format [ rdfs:label "CSV" ] ;
        dcat:mediaType "text/csv" .
```

The Plot used stats1998 and then generated Bar-chart (out-put).

Stats1998 are distributed in CSV format.

Q6.9 What does this mean?

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix dct:  <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://inria.fr/data#> .

:db-employ
 a dcat:Distribution ;
 dcat:downloadURL <http://wimmics.inria.fr/docs/employ-2014.sql> ;
 dct:title "SQL Dump of the employees" ;
 dct:spatial <http://www.geonames.org/6640252> ;
 dct:issued "2015-01-12"^^xsd:date ;
 dct:temporal <http://reference.data.gov.uk/id/year/2014> ;
 dct:publisher <http://inria.fr> ;
 dcat:mediaType "application/sql" ;
 dcat:format [ rdfs:label "SQL" ] ;
 dct:language <http://id.loc.gov/vocabulary/iso639-1/fr> ;
 dcat:byteSize "38729"^^xsd:decimal .

:R2RTransform12 prov:used :db-employ ;
            prov:used :R2R-employ-mapping ;
            prov:used <http://xmlns.com/foaf/0.1/> .

:FoaFDump a void:Dataset;
        void:feature <http://www.w3.org/ns/formats/RDF_XML>;
        void:dataDump <http://wimmics.inria.fr/docs/employ-2014.rdf>;
        void:exampleResource <http://ns.inria.fr/fabien.gandon#me> ;
        void:vocabulary <http://xmlns.com/foaf/0.1/>;
        void:triples 12875;
        dct:title "RDF Dump of the employees" ;
        prov:wasGeneratedBy :R2RTransform12 ;
        prov:generatedAtTime "2015-01-14T11:38:27"^^xsd:dateTime ;
        prov:wasDerivedFrom :db-employ .
```

‘db-employ’ is a Distribution. We can find it with the mentioned URL. The title is “SQL Dump of the employees”, came from “http://www.geonames.org/6640252” at 2015-01-12, published by Inria. It has the media type of SQL. Its size is 38729 bytes.

The ‘R2RTransform12’ algorithm consumes the SQL Dump of the employees and transformed it into RDF. It used FoaF to represent the employees (12875 triples).

Q6.10 practice:

1. Here is the namespace of Ontology for Media Resources:
   http://www.w3.org/ns/ma-ont#
   Negotiate the turtle version of the ontology with curl/wget
2. Study the ontology to know if I can use the property hasLanguage on a Person

1. Negotiate the turtle version :

   curl -o media-ont.ttl -L -H "Accept: text/turtle" www.w3.org/ns/ma-ont#

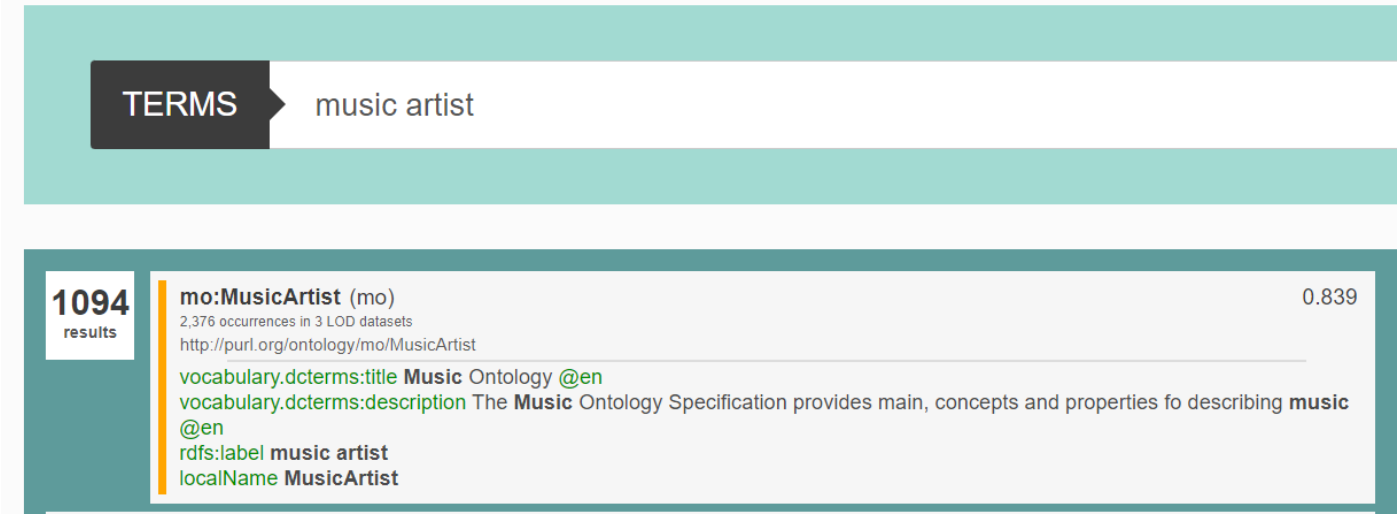
2. Study the ontology :

   We cannot use the property hasLanguage on a Person because the domain of hasLanguage is Media Resources.

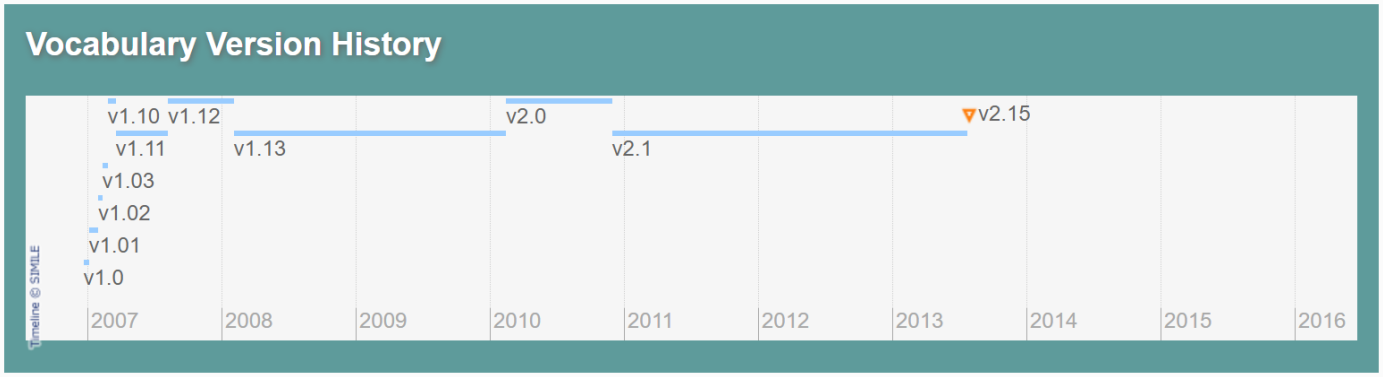
Q6.11 practice:

3. Connect to the LOV directory: https://lov.linkeddata.es/
4. Search for schemas talking about “music artist”.
5. What is the top ontology you find?
6. What is its version number?
7. Is it reused by other ontologies?
8. How many classes and properties does it have?
9. What expressivity does it use? (RDFS, OWL)

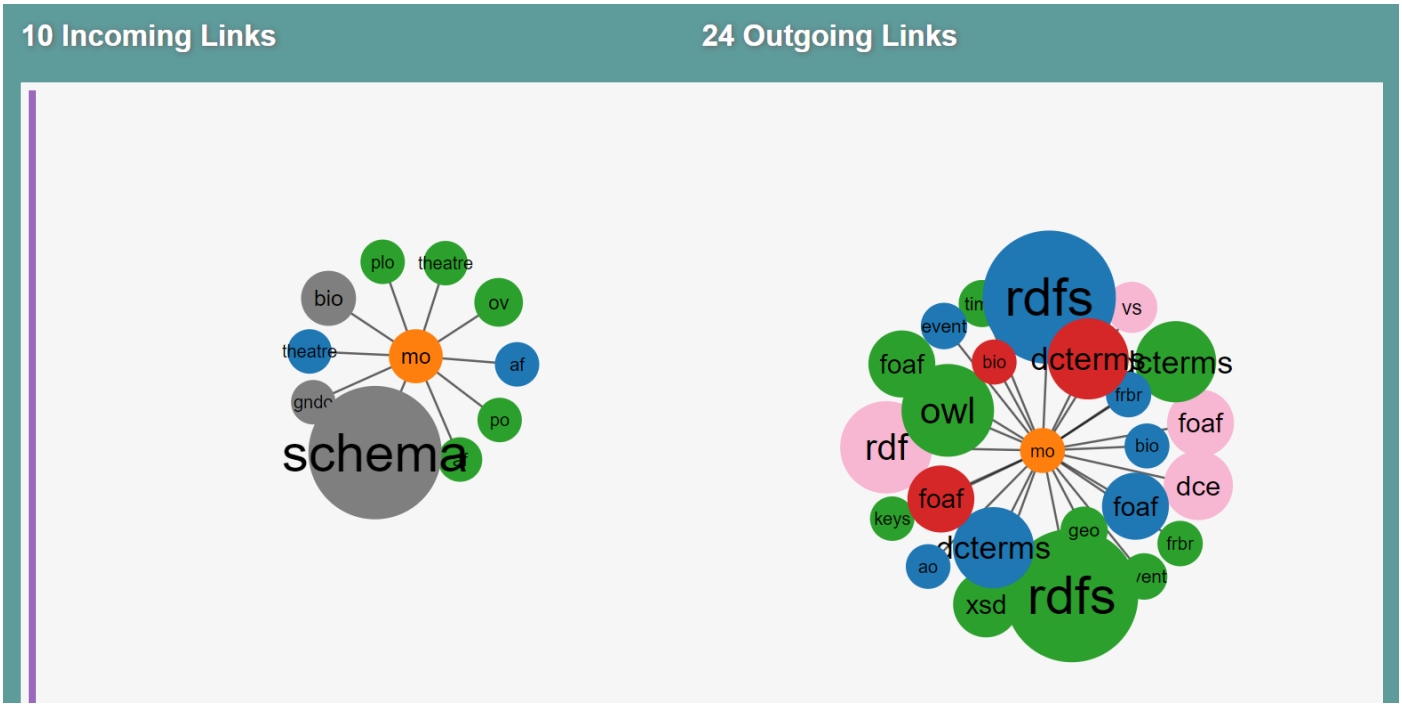
The top ontology: Linked Open Vocabularies (linkeddata.es)



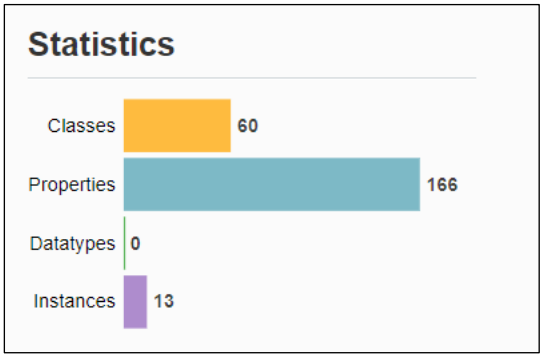
The version number is: V2.15



It is reused by the following ontologies : Schema Ontology, Audio Features Ontology, Theatre Ontology , Playlist Ontology, Programmes ontology, GND Ontology



Classes and properties it has:



Expressivity it uses:

Questions from the course on other data formats.

Q7.1 What are the triples produced with this mapping and this table?

```
:My_Table rdf:type rr:TriplesMap ;
 rr:subjectMap [  rr:template "https://www.ietf.org/rfc/rfc{NUM}.txt"; ];
 rr:predicateObjectMap [
  rr:predicateMap [  rr:predicate dc:title  ];
  rr:objectMap [     rr:column "ttl"     ]
 ].
```

ID	NUM	ttl
87	2616	Hypertext Transfer Protocol -- HTTP/1.1
88	2396	Uniform Resource Identifiers (URI): Generic Syntax

2 triples are constructed as follows:

First triple

Subject: "https://www.ietf.org/rfc/rfc{2616}.txt"

Predicate : dc :title

Object : Hypertext Transfer Protocol -- HTTP/1.1

Second triple

Subject: "https://www.ietf.org/rfc/rfc{2396}.txt"

Predicate : dc :title

Object: Uniform Resource Identifiers (URI): Generic Syntax

Q7.2 What are the triples encoded in this HTML?

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#cathy" typeof="Person">
 <p> <span property="name">Catherine Faron</span>
 (mail: <span property="mbox">faron@i3s.unice.fr</span>) is a friend of
 <span property="knows" resource="http://ns.inria.fr/fabien.gandon#me">Fabien Gandon</span>
 </p>
</div>
```

This HTML code encodes the following RDF triples using the FOAF vocabulary:

- Cathy foaf:name Catherine Faron
- Cathy foaf:mbox faron@i3s.unice.fr
- Cathy foaf:knows <Fabien gandon>

Q7.3 practice:

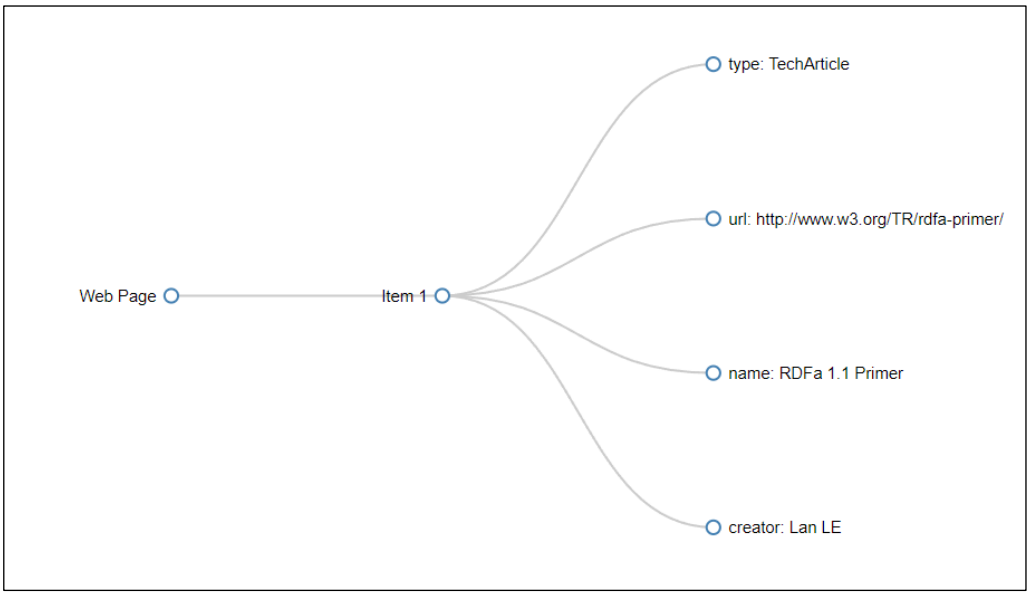
1. Look at the Web Page
   https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html
2. Call the translator on this Web page to get Turtle:
   https://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
3. What does the extracted triple say?
4. Repeat the question with the page:
   https://coffeecode.net/rdfa/codelab/exercises/check_3b.html
5. Test the page:
   http://ns.inria.fr/humans/humans_rdfa.html

```
@prefix dc: <http://purl.org/dc/terms/> .

<https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html> dc:creator "Paul"@en .
```

```
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix schema: <http://schema.org/> .

<https://coffeecode.net/rdfa/codelab/exercises/check_3b.html> rdfa:usesVocabulary schema: .
[]
  a schema:TechArticle ;
  schema:name "Structured data with schema.org codelab" ;
  schema:image <https://coffeecode.net/rdfa/codelab/exercises/squares.png> ;
  schema:educationalUse "codelab" ;
  schema:author <http://example.com/AuthorName> ;
  schema:datePublished "20140129" ;
  schema:description """
    About this codelab
""" ;
  schema:articleBody """
    Exercise 1: From basic HTML to RDFa: first steps
    Exercise 2: Embedded types
    Exercise 3: From strings to things
""" .
```

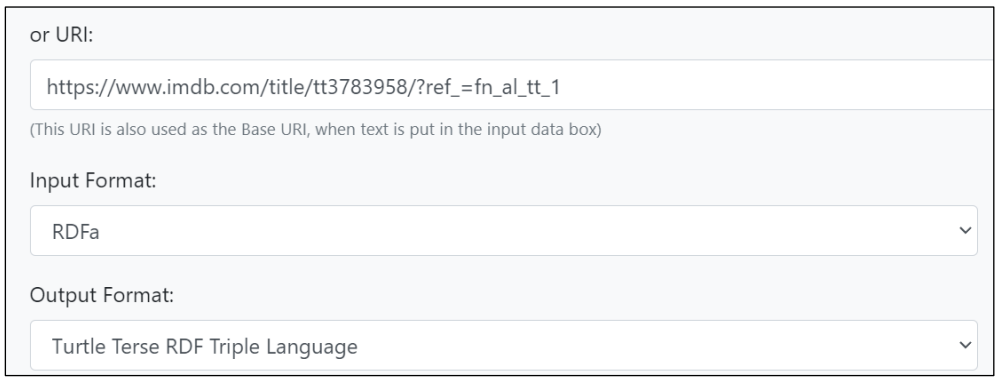
Q7.4 Use the online tool to play with RDFa adding for instance a “creator” property
https://rdfa.info/play/

Lan LE.



Q7.5 IMDB uses RDFa – OGP for the I like button

1. Choose a movie on IMDB http://www.imdb.com
2. Copy the URL of the page of the movie
3. Go to the RDFa 1.0 RDFa Distiller and Parser: https://www.w3.org/2007/08/pyRdfa/
4. Open the URI option, past the URL of the movie page and configure and perform the extraction to get Turtle
5. Try also the transformation on a translator:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/




```
@prefix og: <http://ogp.me/ns#> .
@prefix ns0: <twitter:> .
@prefix ns1: <imdb:> .

<https://www.imdb.com/title/tt3783958/?ref_=fn_al_tt_1>
  og:url "https://www.imdb.com/title/tt3783958/"@en-US ;
  og:site_name "IMDb"@en-US ;
  og:title "La La Land (2016) - IMDb"@en-US ;
  og:description "La La Land: Directed by Damien Chazelle. With Ryan Gosling, Emma Stone, Amiee Conn, Terry
Walters. While navigating their careers in Los Angeles, a pianist and an actress fall in love while attempting to
reconcile their aspirations for the future."@en-US ;
  og:type "video.movie"@en-US ;
```

```
  og:image "https://m.media-
amazon.com/images/M/MV5BMzUzNDM2NzM2MV5BMl5BanBnXkFtZTgwNTM3NTg4OTE@._V1_FMjpg_UX1000_.jpg"@en-US ;
  og:image:height "1481.4814814814813"@en-US ;
  og:image:width "1000"@en-US ;
  og:locale "en_US"@en-US ;
  og:locale:alternate "es_ES"@en-US, "es_MX"@en-US, "fr_FR"@en-US, "fr_CA"@en-US, "it_IT"@en-US, "pt_BR"@en-US,
"hi_IN"@en-US, "de_DE"@en-US ;
  ns0:site "@IMDb"@en-US ;
  ns0:title "La La Land (2016) - IMDb"@en-US ;
  ns0:description "La La Land: Directed by Damien Chazelle. With Ryan Gosling, Emma Stone, Amiee Conn, Terry
Walters. While navigating their careers in Los Angeles, a pianist and an actress fall in love while attempting to
reconcile their aspirations for the future."@en-US ;
  ns0:card "summary_large_image"@en-US ;
  ns0:image "https://m.media-
amazon.com/images/M/MV5BMzUzNDM2NzM2MV5BMl5BanBnXkFtZTgwNTM3NTg4OTE@._V1_FMjpg_UX1000_.jpg"@en-US ;
  ns0:image:alt "La La Land: Directed by Damien Chazelle. With Ryan Gosling, Emma Stone, Amiee Conn, Terry
Walters. While navigating their careers in Los Angeles, a pianist and an actress fall in love while attempting to
reconcile their aspirations for the future."@en-US ;
  ns1:pageType "title"@en-US ;
  ns1:subPageType "main"@en-US ;
  ns1:pageConst "tt3783958"@en-US .
```

Q7.6 Test JSON-LD online

1. Transform your FOAF profile in JSON-LD with the translator:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/

2. Use the following online tool to generate different variations of JSON-LD of your profile (expanded, collapsed, flattened, etc.)
   http://json-ld.org/playground/


```
{
 "@graph": [
  {
   "@id": "_:b0",
   "@type": "http://xmlns.com/foaf/0.1/Person",
   "http://xmlns.com/foaf/0.1/mbox_sha1sum": "667f3a1b28fdf8d4d0940e9600bca2673fa5ee13",
   "http://xmlns.com/foaf/0.1/name": "Ha Ngan Nguyen"
  },
  {
   "@id": "http://njh.me/",
   "@type": "http://xmlns.com/foaf/0.1/PersonalProfileDocument",
   "http://webns.net/mvcb/errorReportsTo": {
    "@id": "mailto:leigh@ldodds.com"
   },
   "http://webns.net/mvcb/generatorAgent": {
    "@id": "http://www.ldodds.com/foaf/foaf-a-matic"
   },
   "http://xmlns.com/foaf/0.1/maker": {
    "@id": "http://njh.me/#me"
   },
   "http://xmlns.com/foaf/0.1/primaryTopic": {
    "@id": "http://njh.me/#me"
   }
  },
  {
   "@id": "http://njh.me/#me",
   "@type": "http://xmlns.com/foaf/0.1/Person",
```

```
    "http://xmlns.com/foaf/0.1/family_name": "LE",
    "http://xmlns.com/foaf/0.1/givenname": "Lan",
    "http://xmlns.com/foaf/0.1/knows": {
     "@id": "_:b0"
    },
    "http://xmlns.com/foaf/0.1/mbox_sha1sum": "c96f0e62ddf7b50186955fc00ee9776d7fcc8b73",
    "http://xmlns.com/foaf/0.1/name": "Lan LE",
    "http://xmlns.com/foaf/0.1/phone": {
     "@id": "tel:06-12-34-56-78-"
    },
    "http://xmlns.com/foaf/0.1/schoolHomepage": {
     "@id": "https://www.datasciencetech.institute/fr/"
    },
    "http://xmlns.com/foaf/0.1/title": "Mrs"
   }
  ]
}
```


Q7.7 To provide the metadata of a CSV file I can...

- ☐ include them in a special column of the CSV.
- ☐ **put them in a file with the same name plus “-metadata.json”.**
- ☐ put them in the first line of my CSV file.
- ☐ **put them in a file called “csv-metadata.json” in the same directory.**
- ☐ add the URL of the metadata file to the content of my CSV file.

put them in a file with the same name plus “-metadata.json”.

put them in a file called “csv-metadata.json” in the same directory.

Q7.8 TV Catalog : Imagine we submit the following call to an LDP platform

```
GET /catalog/tv/ HTTP/1.1
Host: example.org
Accept: text/turtle; charset=UTF-8
```

and we receive the following answer:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type", <http://www.w3.org/ns/ldp#DirectContainer>; rel="type"
Allow: OPTIONS,HEAD,GET,POST,PUT
Accept-Post: text/turtle, application/ld+json
Content-Length: 232
ETag: W/"90231678"
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix cat: <http://example.org/vocab/catalog#> .
<> a ldp:DirectContainer;   ldp:membershipResource <#cat>;   ldp:hasMemberRelation cat:hasProduct;
  dcterms:title "Container of the TV descriptions";
  ldp:contains <tv1>, <tv2> .
<#cat> a cat:Catalog;   dcterms:title "Catalog of TVs";   cat:hasProduct <tv1>, <tv2> .
```

Which ones of the following statements are true?

- ☐ the container is just a basic container.
- ☐ **the container is a direct container.**
- ☐ the container is an indirect container.
- ☐ **the platform accepts the GET calls.**
- ☐ the platform accepts the PATCH calls.
- ☐ the platform accepts RDF/XML format.
- ☐ **the platform accepts RDF Turtle.**
- ☐ **the platform accepts RDF JSON-LD.**
- ☐ **a link hasProduct is automatically created between the resource #cat and the resources of this container**

the container is a direct container.
the platform accepts the GET calls.
the platform accepts RDF Turtle.
the platform accepts RDF JSON-LD.
a link hasProduct is automatically created between the resource #cat and the resources of this container

LAB SESSIONS

Remember: just like for programming, it is a good practice to write & validate step by step, incrementally, and to start from copy-pasted examples from the course.

Lab session on RDF.

Software requirements

- A real text editor (e.g. Notepad++, Gedit, Sublime Text, Emacs, etc.) do not use Word!
- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- An RDF online translator:
  - http://www.easyrdf.org/converter
  - http://rdf.greggkellogg.net/distiller
  - https://issemantic.net/rdf-converter
  - http://rdfvalidator.mybluemix.net/
  - http://rdf-translator.appspot.com/
- The SPARQL Corese engine (CORESE-GUI) : https://project.inria.fr/corese/

Understand existing data

1, If you haven't do it yet during the course, get the RDF/XML about http://ns.inria.fr/fabien.gandon#me and translate the RDF/XML into Turtle/N3 syntax using one of the online translators.

Code of validated RDF in N3 syntax:

Answer:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://ns.inria.fr/fabien.gandon> a foaf:PersonalProfileDocument ;
   foaf:maker <http://ns.inria.fr/fabien.gandon#me> ;
   foaf:primaryTopic <http://ns.inria.fr/fabien.gandon#me> .

<http://ns.inria.fr/fabien.gandon#me> a foaf:Person ;
   foaf:depiction <http://www-sop.inria.fr/members/Fabien.Gandon/common/FabienGandonBackground.jpg> ;
   foaf:family_name "Gandon" ;
   foaf:givenname "Fabien" ;
   foaf:homepage <http://fabien.info> ;
   foaf:knows [ a foaf:Person ;
         rdfs:seeAlso <http://www-sop.inria.fr/members/Olivier.Corby/> ;
         foaf:mbox <mailto:olivier.corby@inria.fr> ;
         foaf:name "Olivier Corby" ],
      [ a foaf:Person ;
         rdfs:seeAlso <http://www.i3s.unice.fr/~faron/> ;
         foaf:mbox <mailto:faron@polytech.unice.fr> ;
```

```
        foaf:name "Catherine Faron-Zucker" ] ;
    foaf:mbox <mailto:fabien.gandon@inria.fr> ;
    foaf:name "Fabien Gandon" ;
    foaf:nick "Bafien" ;
    foaf:phone <http://ns.inria.fr/tel:0492387788> ;
    foaf:schoolHomepage <http://www.insa-rouen.fr> ;
    foaf:title "Dr" ;
    foaf:workInfoHomepage <http://fabien.info> ;
    foaf:workplaceHomepage <http://www.inria.fr/> .
```

In the RDF Turtle file, can you identify all the links between the two resources http://ns.inria.fr/fabien.gandon and http://ns.inria.fr/fabien.gandon#me ? What do they represent?

```
<http://ns.inria.fr/fabien.gandon> a foaf:PersonalProfileDocument ;
<http://ns.inria.fr/fabien.gandon#me> a foaf:Person ;
```

2, Get the Turtle data of Paris on DBpedia.org then in the file find the triple that declares it as a capital in Europe.

The triple is:

```
dbr:Paris       rdf:type        geo:SpatialThing ,
                yago:WikicatPrefecturesInFrance ,
                yago:WikicatWorldHeritageSitesInFrance ,
                yago:WikicatCitiesInFrance ,
                yago:WikicatCitiesWithMillionsOfInhabitants ,
                yago:WikicatCapitals ,
                yago:WikicatCapitalsInEurope ,
```

Humans Knowledge Graph and its namespace http://ns.inria.fr/humans/data#

The major part of this practical session is using a small dataset about a few persons.

The namespace of the dataset is http://ns.inria.fr/humans/data#

1. With your Web browser, visit that namespace and spot the age of Gaston in the graph

Answer: 102

2. Use the command curl or wget to obtain the **XML version** of this dataset from the same address and download it in a file named "**humans_data.xml**" then use the W3C RDF online validation service to validate the RDF/XML and see the triples and the graph.

Validation Results

Your RDF document validated successfully.

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://ns.inria.fr/humans/data#Harry	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Man
2	http://ns.inria.fr/humans/data#Harry	http://ns.inria.fr/humans/schema#name	"Harry"
3	http://ns.inria.fr/humans/data#Harry	http://ns.inria.fr/humans/schema#hasChild	http://ns.inria.fr/humans/data#John
4	http://ns.inria.fr/humans/data#Harry	http://ns.inria.fr/humans/schema#hasSpouse	http://ns.inria.fr/humans/data#Sophie
5	http://ns.inria.fr/humans/data#John	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Person
6	http://ns.inria.fr/humans/data#John	http://ns.inria.fr/humans/schema#name	"John"
7	http://ns.inria.fr/humans/data#John	http://ns.inria.fr/humans/schema#shoesize	"14"^^http://www.w3.org/2001/XMLSchema#integer
8	http://ns.inria.fr/humans/data#John	http://ns.inria.fr/humans/schema#age	"37"^^http://www.w3.org/2001/XMLSchema#integer
9	http://ns.inria.fr/humans/data#Mark	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Person
10	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#name	"Mark"
11	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#shoesize	"8"^^http://www.w3.org/2001/XMLSchema#integer
12	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#age	"14"^^http://www.w3.org/2001/XMLSchema#integer
13	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#shirtsize	"9"^^http://www.w3.org/2001/XMLSchema#integer
14	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#trouserssize	"36"^^http://www.w3.org/2001/XMLSchema#integer
15	http://ns.inria.fr/humans/data#Mark	http://ns.inria.fr/humans/schema#hasFather	http://ns.inria.fr/humans/data#John
16	http://ns.inria.fr/humans/data#Eve	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Person
17	http://ns.inria.fr/humans/data#Eve	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.inria.fr/2007/09/11/humans.rdfs#Lecturer
18	http://ns.inria.fr/humans/data#Eve	http://ns.inria.fr/humans/schema#hasSpouse	http://ns.inria.fr/humans/data#David
19	http://ns.inria.fr/humans/data#Eve	http://ns.inria.fr/humans/schema#name	"Eve"
20	http://ns.inria.fr/humans/data#Eve	http://ns.inria.fr/humans/schema#hasFriend	http://ns.inria.fr/humans/data#Alice
21	http://ns.inria.fr/humans/data#David	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Person
22	http://ns.inria.fr/humans/data#David	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.inria.fr/2007/09/11/humans.rdfs#Researcher
23	http://ns.inria.fr/humans/data#David	http://ns.inria.fr/humans/schema#hasFriend	http://ns.inria.fr/humans/data#Gaston
24	http://ns.inria.fr/humans/data#David	http://ns.inria.fr/humans/schema#name	"David"
25	http://ns.inria.fr/humans/data#Alice	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Woman
26	http://ns.inria.fr/humans/data#Alice	http://ns.inria.fr/humans/schema#hasFriend	http://ns.inria.fr/humans/data#John
27	http://ns.inria.fr/humans/data#Alice	http://ns.inria.fr/humans/schema#name	"Alice"
28	http://ns.inria.fr/humans/data#Jack	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Man
29	http://ns.inria.fr/humans/data#Jack	http://ns.inria.fr/humans/schema#hasFriend	http://ns.inria.fr/humans/data#Alice
30	http://ns.inria.fr/humans/data#Jack	http://ns.inria.fr/humans/schema#hasChild	http://ns.inria.fr/humans/data#Harry
31	http://ns.inria.fr/humans/data#Jack	http://ns.inria.fr/humans/schema#name	"Jack"
32	http://ns.inria.fr/humans/data#Flora	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ns.inria.fr/humans/schema#Woman

3. Use the command curl or wget to obtain the **Turtle version** of this dataset from the same address and download it in a file named “humans_data.ttl”

4. What is the namespace used for instances created / resources described in this file?

@prefix d: <http://ns.inria.fr/humans/data#> .

5. In the file how is the association between resources described (Gaston, Laura, etc.) and their namespace done i.e. how is the namespace of these resources specified?

d: <http://ns.inria.fr/humans/data#> .

6. What is the namespace of the vocabulary used to describe the resources (hasParent, name, etc.) in the dataset and how is it specified?

default prefix declaration: @prefix : <http://ns.inria.fr/humans/schema#> .

7. Find *everything* about John in the turtle file, all available information:

```
d:John a :Person ;
   :age 37 ;
   :hasParent d:Sophie ;
   :name "John" ;
   :shirtsize 12 ;
   :shoesize 14 ;
   :trouserssize 44 .

d:Jennifer a :Woman ;
   :hasSpouse d:John ;

d:Mark a :Person ;
   :hasFather d:John ;

d:Harry a :Man ;
```

```
                    :hasChild d:John ;

                d:Alice a :Woman ;
                    :hasFriend d:John ;
```

Create RDF

Here is a statement extracted from the course:

> "Jen is an engineer woman, 42-year old, married to Seb who is a man with whom she had two children: Anny who is a woman and Steffen who is a man".

1. Use your text editor and write the above statements in RDF in N3 syntax inventing your own vocabulary. Save you file as "Jen.ttl"
2. Use your favorite text or XML editor and write the above statements in RDF in XML syntax reusing the same vocabulary "Jen.rdf"
3. Use the RDF XML online validation service to validate your XML and see the triples https://www.w3.org/RDF/Validator/
4. In the validator use the option to visualize the graph
    - Use the RDF online translator to validate your N3 and translate it into RDF/XML: http://www.easyrdf.org/converter
    - http://rdf.greggkellogg.net/distiller
    - https://issemantic.net/rdf-converter
    - http://rdfvalidator.mybluemix.net/
    - http://rdf-translator.appspot.com/
5. Compare your RDF/XML with the result of the N3 translation
6. Translate in other formats to see the results.

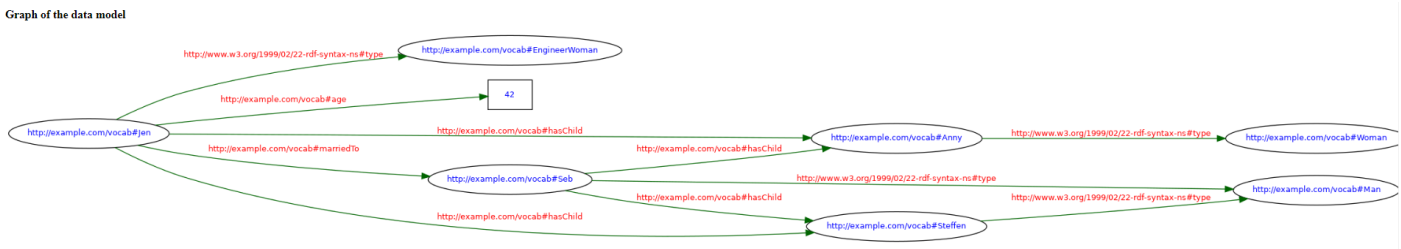
Code of validated RDF in N3 syntax:

```
@prefix vocab: <http://example.com/vocab#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

vocab:Jen a vocab:EngineerWoman ;
   vocab:age 42 ;
   vocab:hasChild vocab:Anny,
      vocab:Steffen ;
   vocab:marriedTo vocab:Seb .

vocab:Seb a vocab:Man ;
   vocab:hasChild vocab:Anny,
      vocab:Steffen .

vocab:Steffen a vocab:Man .

vocab:Anny a vocab:Woman .
```

Code of validated RDF in XML syntax:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ns0="http://example.com/vocab#">

  <rdf:Description rdf:about="http://example.com/vocab#Jen">
```

```
    <rdf:type rdf:resource="http://example.com/vocab#EngineerWoman"/>
    <ns0:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42</ns0:age>
    <ns0:hasChild rdf:resource="http://example.com/vocab#Anny"/>
    <ns0:hasChild rdf:resource="http://example.com/vocab#Steffen"/>
    <ns0:marriedTo>
      <ns0:Man rdf:about="http://example.com/vocab#Seb">
        <ns0:hasChild rdf:resource="http://example.com/vocab#Anny"/>
        <ns0:hasChild rdf:resource="http://example.com/vocab#Steffen"/>
      </ns0:Man>
    </ns0:marriedTo>

  </rdf:Description>

  <ns0:Man rdf:about="http://example.com/vocab#Steffen">
  </ns0:Man>

  <ns0:Woman rdf:about="http://example.com/vocab#Anny">
  </ns0:Woman>

</rdf:RDF>
```



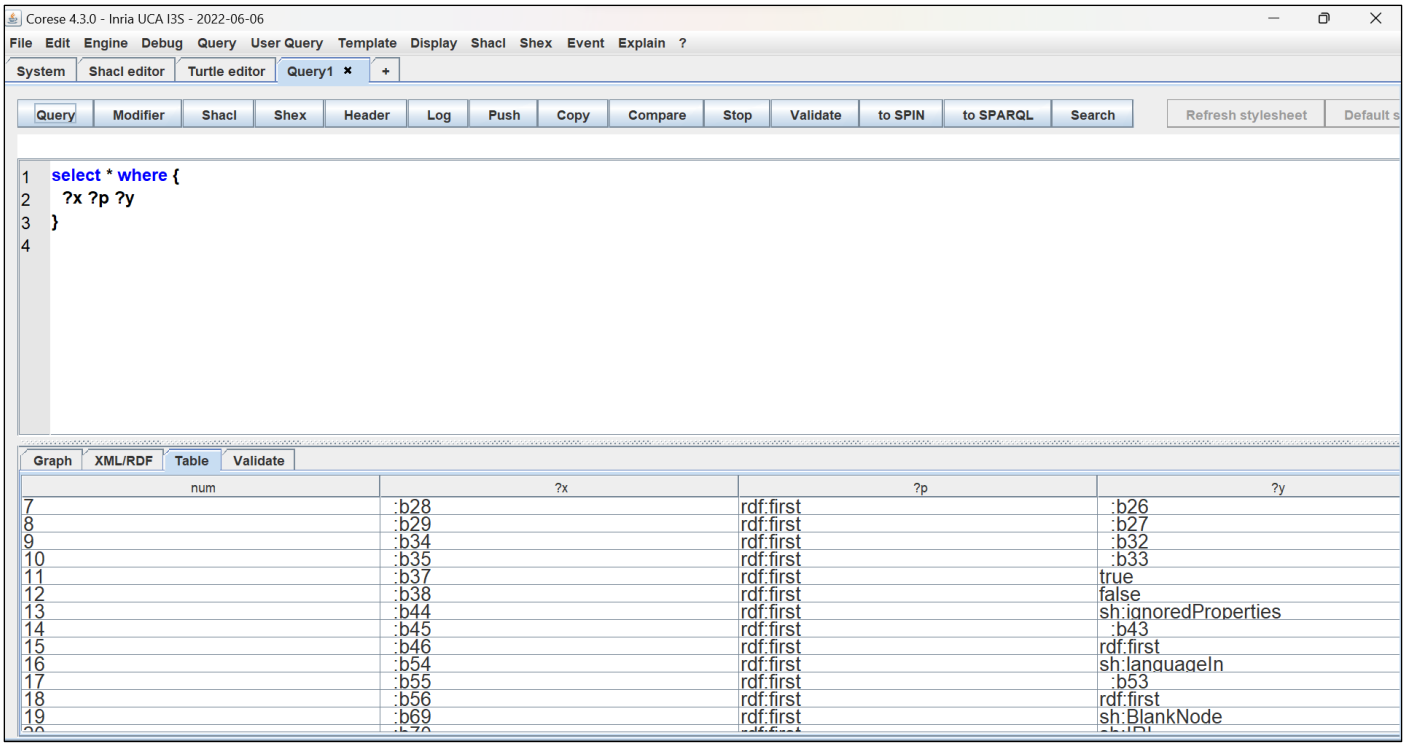
Query your data

Download the Corese.jar library and start it as a standalone application: 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 several tabs: (1) the System tab for traces of execution, (2) a SHACL editor tab (3) a Turtle Editor tab and (4) A "+" tab to create as many queries as you want. Load the annotations contained in the file "Jen.rdf" you created and validated before. Click on the "+" tab to create a new query and the interface contains a default SPARQL query:

```
select * where {  ?x ?p ?y}
```

The SPARQL language will be presented in the next course. Just know that this query can find all the triples of your data. Launch the query and check the results.


Corese 4.3.0 - Inria UCA I3S - 2022-06-06
File Edit Engine Debug Query User Query Template Display Shacl Shex Event Explain ?
System Shacl editor Turtle editor Query1 +
Query Modifier Shacl Shex Header Log Push Copy Compare Stop Validate to SPIN to SPARQL Search
Refresh stylesheet
1 select * where {
2 ?x ?p ?y
3 }
4
Graph XML/RDF Table Validate
num ?x ?p ?y
7 :b28 rdf:first :b26
8 :b29 rdf:first :b27
9 :b34 rdf:first :b32
10 :b35 rdf:first :b33
11 :b37 rdf:first true
12 :b38 rdf:first false
13 :b44 rdf:first sh:ignoredProperties
14 :b45 rdf:first :b43
15 :b46 rdf:first rdf:first
16 :b54 rdf:first sh:languageIn
17 :b55 rdf:first :b53
18 :b56 rdf:first rdf:first
19 :b69 rdf:first sh:BlankNode

Lab session on SHACL.

Software requirements

- A real text editor (e.g. Notepad++, Gedit, Sublime Text, Emacs, etc.)
- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The SPARQL Corese engine (Corese-GUI jar file): https://project.inria.fr/corese/
- The human dataset file http://ns.inria.fr/humans/data#
- The SHACL file http://ns.inria.fr/humans/humans_shape.ttl

What is that shape

With you text editor open the file humans_shape.ttl and look at the content

```
@prefix :     <http://ns.inria.fr/humans/schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:  <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:PersonShape a sh:NodeShape ;

        sh:targetClass :Person;


        sh:property [
                sh:message "a Person must have a name"@en;
                sh:severity sh:Violation;
                sh:path :name ;
                sh:minCount 1
        ] .
```

What is the qualified name of the main shape being defined:

:PersonShape

What is the type of that shape:

NodeShape

What is the target of that shape:

Class of individuals called "Person"

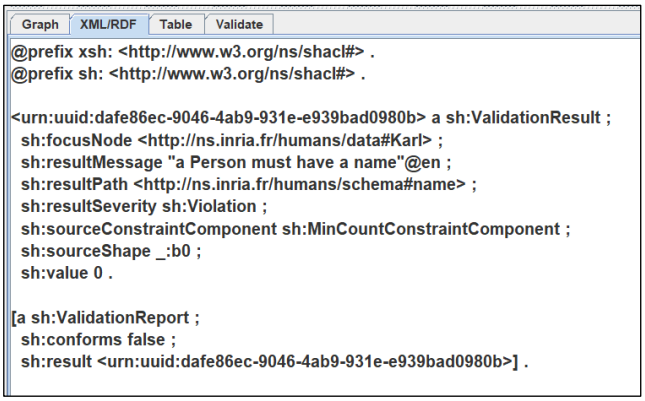
Explain in English the constraint it places on the focus node:

The shape has a single property constraint, requiring that a person must have at least 1 name.

What is the severity level of that constraint?

The constraint requires that a Person need to have one "name". If not, it will violate the shape and a validation report will be generated with an error message "a Person must have a name".

In Corese load the dataset humans_data.ttl (menu "file > load > Dataset") and this shape (menu "file > load > SHACL") and run the validation in a query tab (button "SHACL" in a query tab). Explain in English what the report is saying:



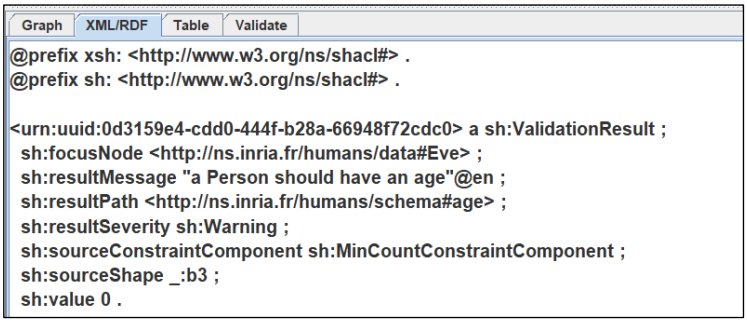
The message says "a Person should have an age" for Karl

Add your constraints

Extend the shape to add a constraint of severity level “Warning” enforcing that a Person should have an age:

```
@prefix :     <http://ns.inria.fr/humans/schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:  <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:PersonShape a sh:NodeShape ;

    sh:targetClass :Person ;

    sh:property [
        sh:message "a Person must have a name"@en ;
        sh:severity sh:Violation ;
        sh:path :name ;
        sh:minCount 1
    ] ;

    sh:property [
        sh:message "a Person should have an age"@en ;
        sh:severity sh:Warning ;
        sh:path :age ;
        sh:minCount 1 ;
        sh:datatype xsd:integer ;
    ] .
```

In Corese load the dataset humans_data.ttl (menu “file > load > Dataset”) and this shape (menu “file > load > SHACL”) and run the validation in a query tab (button “SHACL” in a query tab). Explain in English what the report is saying:



The message says 'a person should have an age' for each person who does not have an age (Eve, Laura, David). And it’s noted that one person doesn’t have a name.

Extend the shape to add a constraint of severity level “Info” enforcing that a person’s name should be in English:

```
@prefix :     <http://ns.inria.fr/humans/schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:  <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:PersonShape a sh:NodeShape ;

    sh:targetClass :Person ;

    sh:property [
        sh:message "a Person must have a name"@en ;
        sh:severity sh:Violation ;
        sh:path :name ;
        sh:minCount 1
    ] ;
    sh:property [
        sh:message "a Person should have an age"@en ;
        sh:severity sh:Warning ;
        sh:path :age ;
        sh:minCount 1 ;
        sh:datatype xsd:integer ;
    ] ;
    sh:property [
        sh:message "a Person's name should be in English"@en ;
        sh:severity sh:Info ;
        sh:path :name ;
        sh:languageIn "en" ;
    ] .
```

In Corese load the dataset humans_data.ttl (menu “file > load > Dataset”) and this shape (menu “file > load > SHACL”) and run the validation in a query tab (button “SHACL” in a query tab). Explain in English what the report is saying:

The message says besides the above mentioned non conforms nodes, there are also non-conformity for language of name.

Lab session on SPARQL.

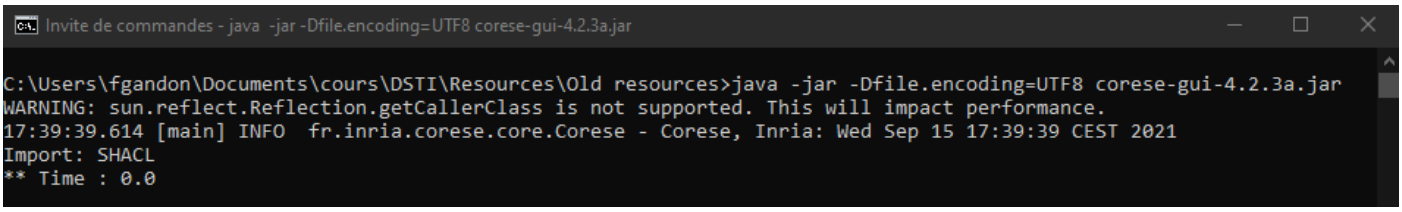
Software requirements

- 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 (Corese-GUI jar file): https://project.inria.fr/corese/

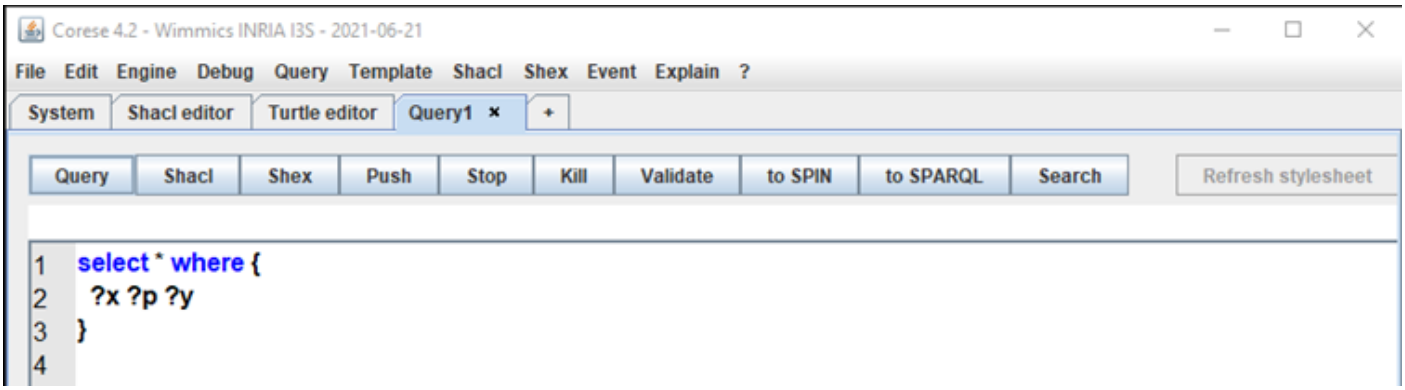
Basic query on RDF humans dataset

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

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. Example:



This interface provides several tabs: (1) the System tab for traces of execution, (2) a SHACL editor tab (3) a Turtle Editor tab and (4) A "+" tab to create as many queries as you want.



You should have the dataset humans_data.ttl from the previous practical session.

Load the file humans_data.ttl as RDF data in CORESE.

NB: CORESE reads all the formats/syntaxes of RDF.

Question 1:

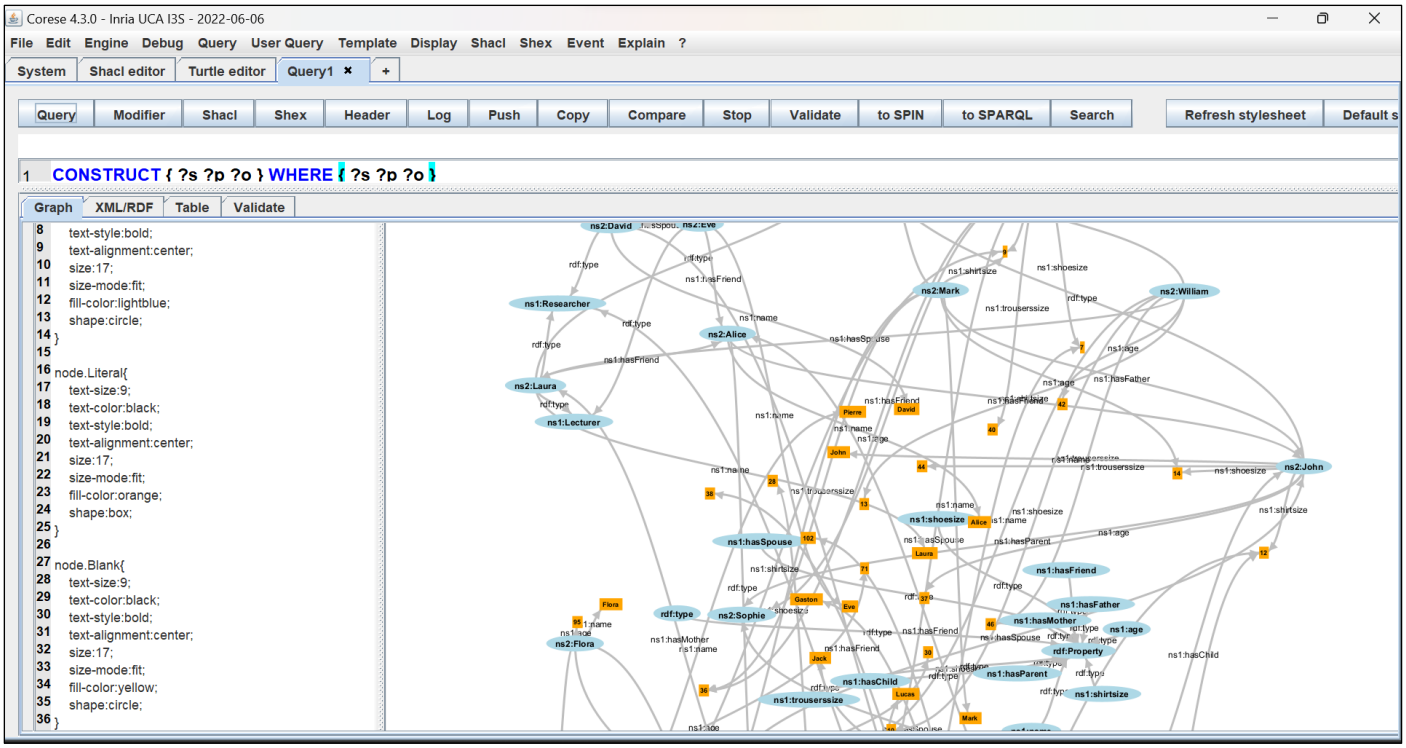
Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

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

Explanation:

This query constructs a new RDF graph by matching patterns in the input RDF graph.

Screenshot:



Question 2:

Create a new tab to enter the following query:

prefix h: <http://ns.inria.fr/humans/schema#>

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

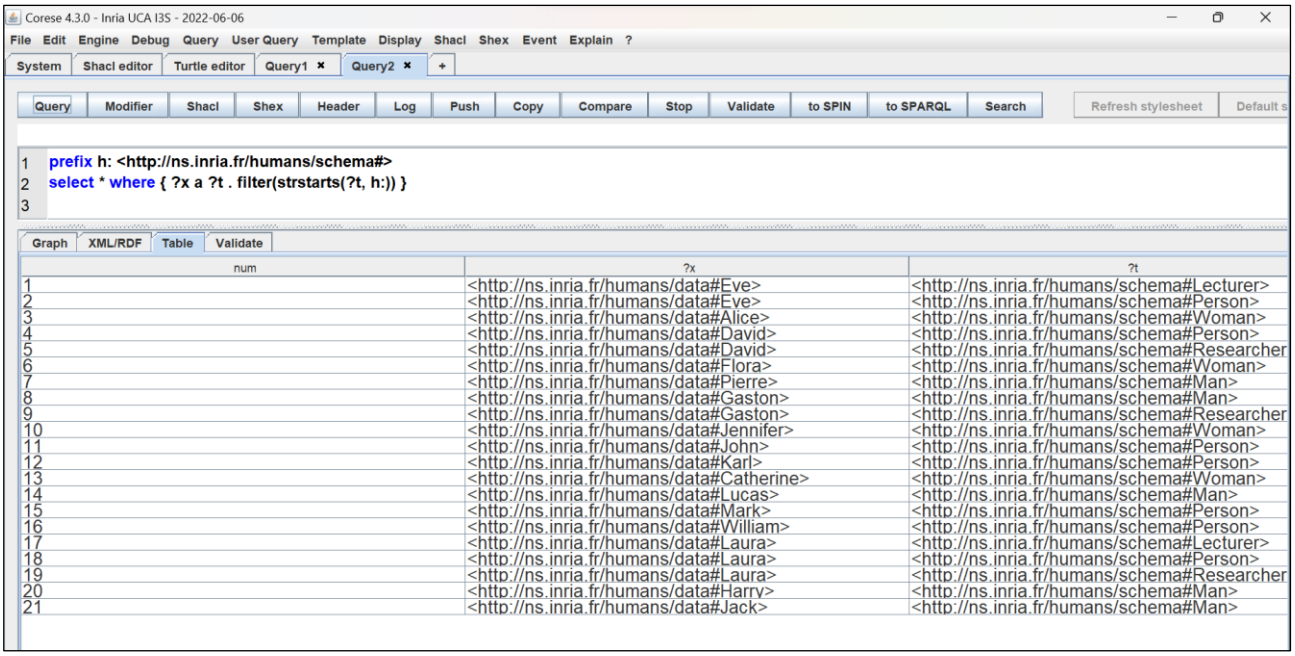
Translate this query in plain English.

This query retrieves all the resources x which have a type of t with the prefix h: <http://ns.inria.fr/humans/schema#>.

The “filter” clause restricts the results to the above-mentioned type, meaning excludes all other types.

Run this query. How many answers do you get?

21

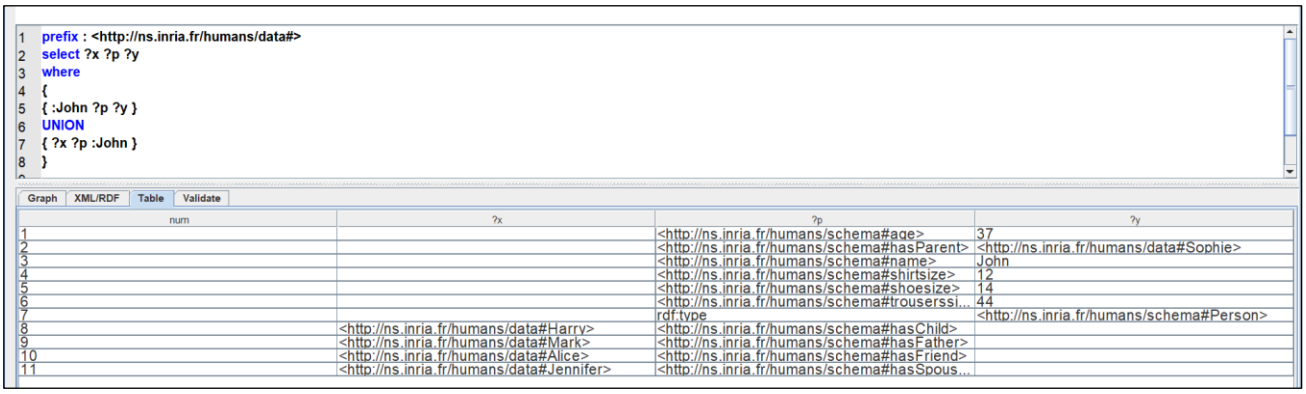


num	?x	?t
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/schema#Lecturer>
2	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/schema#Person>
3	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/schema#Woman>
4	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/schema#Person>
5	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/schema#Researcher
6	<http://ns.inria.fr/humans/data#Flora>	<http://ns.inria.fr/humans/schema#Woman>
7	<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/schema#Man>
8	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/schema#Man>
9	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/schema#Researcher
10	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/schema#Woman>
11	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/schema#Person>
12	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/schema#Person>
13	<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/schema#Woman>
14	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/schema#Man>
15	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/schema#Person>
16	<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/schema#Person>
17	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Lecturer>
18	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Person>
19	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Researcher
20	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/schema#Man>
21	<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/schema#Man>

Find John and his types in the answers.

John's types:

<http://ns.inria.fr/humans/schema#Person>



num	?x	?p	?y
1		<http://ns.inria.fr/humans/schema#age>	37
2		<http://ns.inria.fr/humans/schema#hasParent>	<http://ns.inria.fr/humans/data#Sophie>
3		<http://ns.inria.fr/humans/schema#name>	John
4		<http://ns.inria.fr/humans/schema#shirtsize>	12
5		<http://ns.inria.fr/humans/schema#shoesize>	14
6		<http://ns.inria.fr/humans/schema#trouserssi...	44
7		rdf:type	<http://ns.inria.fr/humans/schema#Person>
8	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/schema#hasChild>	
9	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/schema#hasFather>	
10	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/schema#hasFriend>	
11	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/schema#hasSpous...	

Question 3:

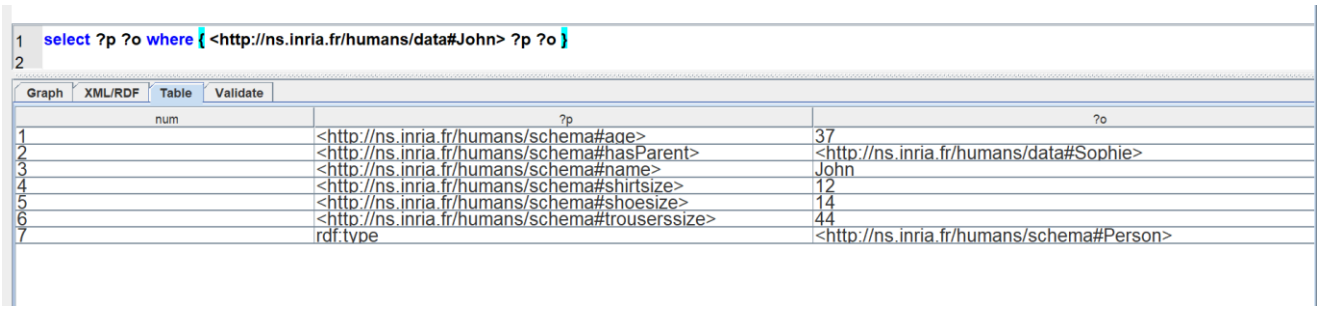
In the previous answer, locate the URI of John.

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

   Query:

   SELECT ?p ?o WHERE { <http://ns.inria.fr/humans/data#John> ?p ?o }

   Results:



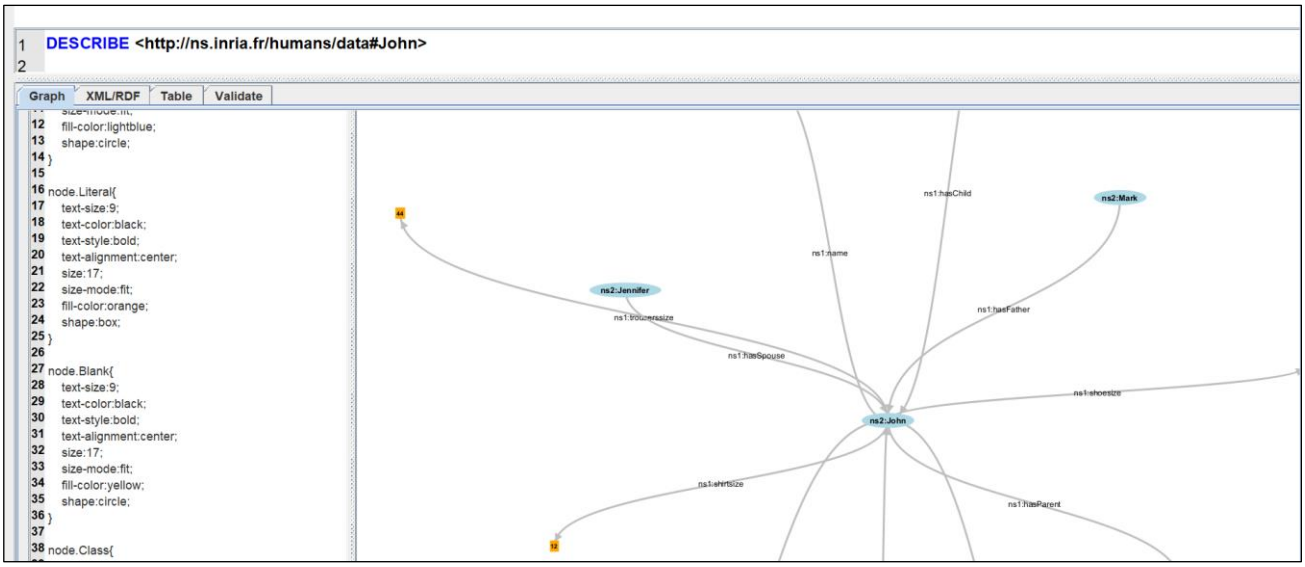
num	?p	?o
1	<http://ns.inria.fr/humans/schema#age>	37
2	<http://ns.inria.fr/humans/schema#hasParent>	<http://ns.inria.fr/humans/data#Sophie>
3	<http://ns.inria.fr/humans/schema#name>	John
4	<http://ns.inria.fr/humans/schema#shirtsize>	12
5	<http://ns.inria.fr/humans/schema#shoesize>	14
6	<http://ns.inria.fr/humans/schema#trouserssize>	44
7	rdf:type	<http://ns.inria.fr/humans/schema#Person>

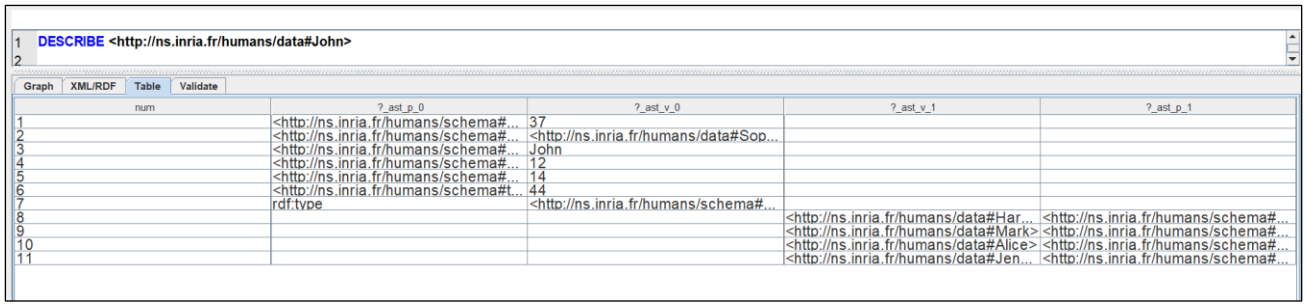
2. request a description of John using the SPARQL clause for this.

Query:

DESCRIBE http://ns.inria.fr/humans/data#John

Results:





Question 4

Create a new tab to enter the following query:

prefix h: <http://ns.inria.fr/humans/schema#>

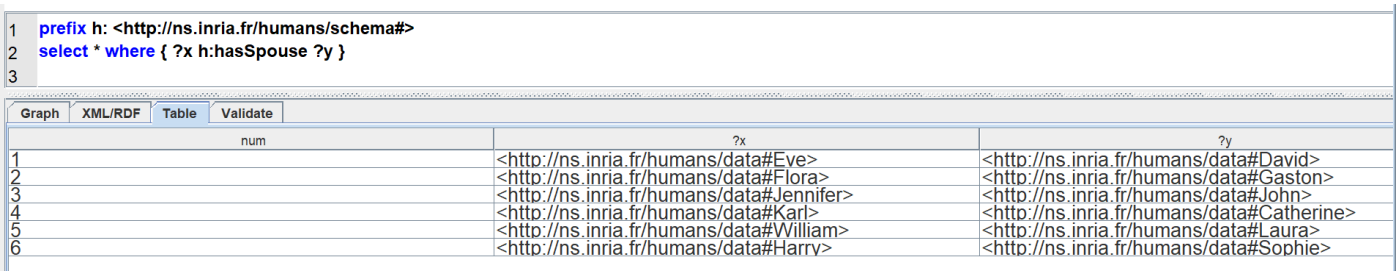
select * where { ?x h:hasSpouse ?y }

Translate this query in plain English.

This query finds all the resources having the property “hasSpouse” and their values.

Run this query. How many answers do you get?

6



Question 5:

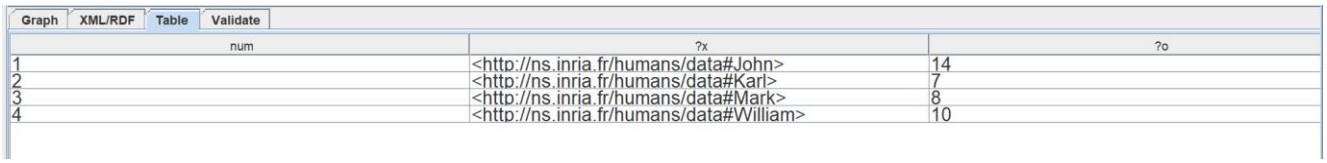
In the RDF file, find the name of the property that is used to give the shoe size of a person.

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT * WHERE { ?x h:shoesize ?o. ?x a h:Person }
```

Result:



Graph | XML/RDF | Table | Validate

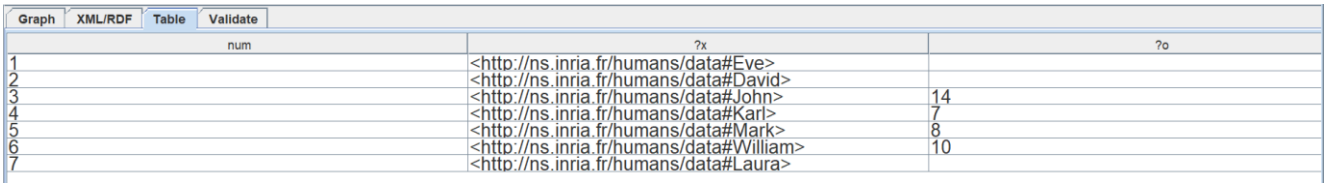
num	?x	?o
1	<http://ns.inria.fr/humans/data#John>	14
2	<http://ns.inria.fr/humans/data#Karl>	7
3	<http://ns.inria.fr/humans/data#Mark>	8
4	<http://ns.inria.fr/humans/data#William>	10

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT * WHERE { ?x a h:Person OPTIONAL {?x h:shoesize ?o } }
```

Result:



Graph | XML/RDF | Table | Validate

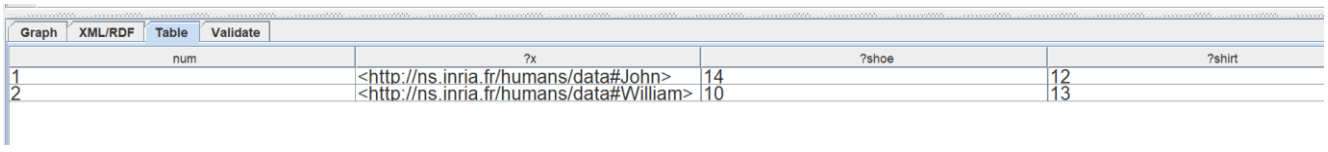
num	?x	?o
1	<http://ns.inria.fr/humans/data#Eve>	
2	<http://ns.inria.fr/humans/data#David>	
3	<http://ns.inria.fr/humans/data#John>	14
4	<http://ns.inria.fr/humans/data#Karl>	7
5	<http://ns.inria.fr/humans/data#Mark>	8
6	<http://ns.inria.fr/humans/data#William>	10
7	<http://ns.inria.fr/humans/data#Laura>	

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT * WHERE { ?x a h:Person . ?x h:shoesize ?shoe. ?x h:shirtsize ?shirt. FILTER ( (?shoe > 8) || (?shirt > 12)) }
```

Result:



Graph | XML/RDF | Table | Validate

num	?x	?shoe	?shirt
1	<http://ns.inria.fr/humans/data#John>	14	12
2	<http://ns.inria.fr/humans/data#William>	10	13

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://ns.inria.fr/humans/schema#>
SELECT ?x WHERE { ?x h:hasChild ?c}
```

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

We got 5 answers.

The answer “Gaston” was duplicated.

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Flora>
2	<http://ns.inria.fr/humans/data#Gaston>
3	<http://ns.inria.fr/humans/data#Gaston>
4	<http://ns.inria.fr/humans/data#Harry>
5	<http://ns.inria.fr/humans/data#Jack>

2. Find a way to avoid duplicates.

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT DISTINCT ?x WHERE { ?x h:hasChild ?c}
```

How many answers do you get then?

We got only 4 answers this time.

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Flora>
2	<http://ns.inria.fr/humans/data#Gaston>
3	<http://ns.inria.fr/humans/data#Harry>
4	<http://ns.inria.fr/humans/data#Jack>

3. Rewrite a query to find the Persons, Men and Women who have no child.

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x
WHERE { {?x a h:Person} MINUS {?x h:hasChild ?c} }
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#David>
3	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Karl>
5	<http://ns.inria.fr/humans/data#Mark>
6	<http://ns.inria.fr/humans/data#William>
7	<http://ns.inria.fr/humans/data#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 persons with their age.

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT * WHERE { ?x h:age ?age }
```

Result:

Graph | XML/RDF | Table | Validate

num	?x	?age
1	<http://ns.inria.fr/humans/data#Flora>	95
2	<http://ns.inria.fr/humans/data#Pierre>	71
3	<http://ns.inria.fr/humans/data#Gaston>	102
4	<http://ns.inria.fr/humans/data#John>	37
5	<http://ns.inria.fr/humans/data#Karl>	36
6	<http://ns.inria.fr/humans/data#Lucas>	12
7	<http://ns.inria.fr/humans/data#Mark>	14
8	<http://ns.inria.fr/humans/data#William>	42

2. Formulate a query to find person who are not adults (here and Adult is a person at least 18 years old).

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT * WHERE { ?x h:age ?age filter (?age <18) }
```

How many answers do you get?

2 answers.

Graph | XML/RDF | Table | Validate

num	?x	?age
1	<http://ns.inria.fr/humans/data#Lucas>	12
2	<http://ns.inria.fr/humans/data#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://ns.inria.fr/humans/schema#>
SELECT ?x ?age (if(?age >= 18, true, false) as ?isAdult)
WHERE {
?x h:name "Mark" .
?x h:age ?age .
}
```

```
1 PREFIX h: <http://ns.inria.fr/humans/schema#>
2 SELECT ?x ?age (if(?age >= 18, true, false) as ?isAdult)
3 WHERE {
4 ?x h:name "Mark" .
5 ?x h:age ?age .
6 }
7
```

Graph | XML/RDF | Table | Validate

num	?x	?age	?isAdult
1	<http://ns.inria.fr/humans/data#Mark>	14	false

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?age (IF (FLOOR(?age/2 )*2= ?age, true, false) as ?even)
WHERE { ?x h:age ?age }
```

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?age (IF (FLOOR(?age/2 )*2= ?age, true, false) as ?even)
WHERE { ?x h:age ?age }
```

Graph | XML/RDF | Table | Validate

num	?x	?age	?even
1	<http://ns.inria.fr/humans/data#Flora>	95	false
2	<http://ns.inria.fr/humans/data#Pierre>	71	false
3	<http://ns.inria.fr/humans/data#Gaston>	102	true
4	<http://ns.inria.fr/humans/data#John>	37	false
5	<http://ns.inria.fr/humans/data#Karl>	36	true
6	<http://ns.inria.fr/humans/data#Lucas>	12	true
7	<http://ns.inria.fr/humans/data#Mark>	14	true
8	<http://ns.inria.fr/humans/data#William>	42	true

OR

PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?age ?even
WHERE { ?x h:age ?age . BIND ((FLOOR(?age/2)*2 = ?age) AS ?even)}

```
prefix h: <http://ns.inria.fr/humans/schema#>
select ?x ?age ?even
where { ?x h:age ?age .
      BIND ((FLOOR(?age/2)*2 = ?age) AS ?even)
}
```

Graph | XML/RDF | Table | Validate

num	?x	?age	?even
1	<http://ns.inria.fr/humans/data#Flora>	95	false
2	<http://ns.inria.fr/humans/data#Pierre>	71	false
3	<http://ns.inria.fr/humans/data#Gaston>	102	true
4	<http://ns.inria.fr/humans/data#John>	37	false
5	<http://ns.inria.fr/humans/data#Karl>	36	true
6	<http://ns.inria.fr/humans/data#Lucas>	12	true
7	<http://ns.inria.fr/humans/data#Mark>	14	true
8	<http://ns.inria.fr/humans/data#William>	42	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://ns.inria.fr/humans/schema#>
CONSTRUCT {?y h:hasFriend ?x } WHERE {?x h:hasFriend ?y}

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#Alice>
2	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#John>
3	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/data#Gaston>
4	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Sophie>
5	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/data#Alice>
6	<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Alice>

2. **Insert** 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://ns.inria.fr/humans/schema#>
INSERT {?y h:hasFriend ?x } WHERE {?x h:hasFriend ?y}

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#Alice>
2	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Eve>
3	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Laura>
5	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Jack>
6	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/data#Gaston>
7	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/data#David>
8	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Alice>
9	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Sophie>
10	<http://ns.inria.fr/humans/data#Sophie>	<http://ns.inria.fr/humans/data#Karl>
11	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/data#Alice>
12	<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Alice>

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:

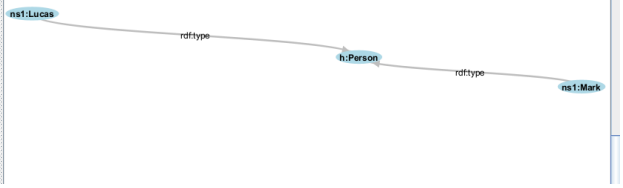
```
PREFIX h: <http://ns.inria.fr/humans/schema#>
CONSTRUCT {?x a h:Person} WHERE { ?x h:age ?age filter (?age <18) }
```

Result:

```
1  prefix h: <http://ns.inria.fr/humans/schema#>
2  construct {?x a h:Person} where { ?x h:age ?age filter (?age <18) }
3
```

Graph | XML/RDF | Table | Validate

```
23    fill-color:orange;
24    shape:box;
25  }
26
27  node.Blank{
28    text-size:9;
29    text-color:black;
30    text-style:bold;
31    text-alignment:center;
32    size:17;
33    size-mode:fit;
34    fill-color:yellow;
35    shape:circle;
36  }
37
38  node.Class{
```



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:

```
1  prefix h: <http://ns.inria.fr/humans/schema#>
2  SELECT ?p ?o WHERE { <http://ns.inria.fr/humans/data#Lan> ?p ?o }
3
```

Graph | XML/RDF | Table | Validate

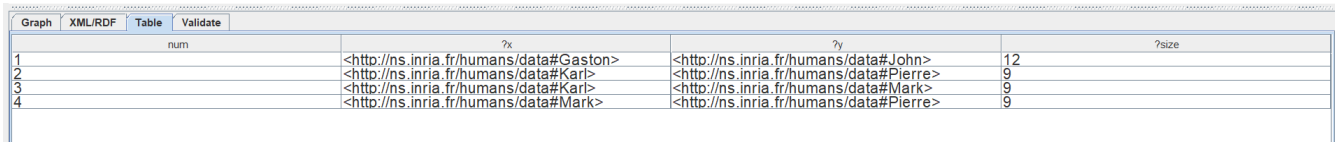
num	?p	?o
1	<http://ns.inria.fr/humans/schema#age>	30
2	<http://ns.inria.fr/humans/schema#hasFriend>	<http://ns.inria.fr/humans/data#Hangan>
3	<http://ns.inria.fr/humans/schema#hasMother>	<http://ns.inria.fr/humans/data#Ban>
4	<http://ns.inria.fr/humans/schema#shoesize>	4
5	rdf:type	<http://ns.inria.fr/humans/schema#Woman>

Question 11

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?y ?size
WHERE {
?x h:shirtsize ?size .
?y h:shirtsize ?size
FILTER (?x < ?y)
}
```



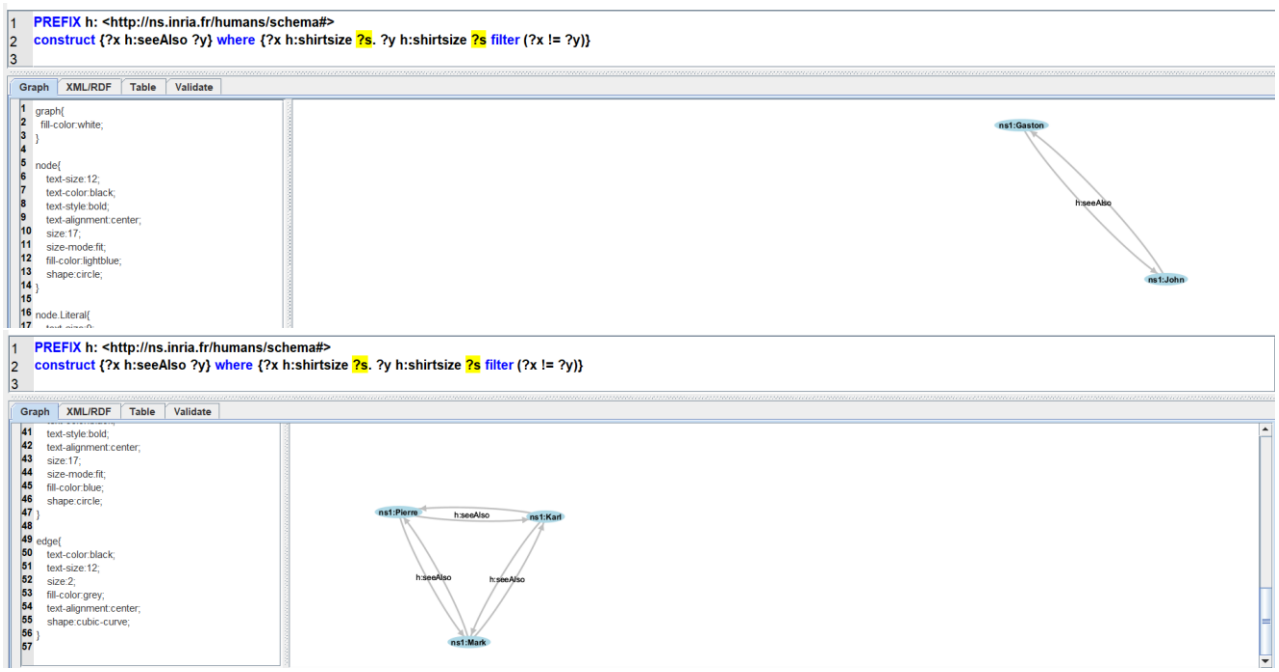
Graph | XML/RDF | Table | Validate

num	?x	?y	?size
1	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/data#John>	12
2	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Pierre>	9
3	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Mark>	9
4	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Pierre>	9

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
CONSTRUCT {?x h:seeAlso ?y}
WHERE {
?x h:shirtsize ?s .
?y h:shirtsize ?s
FILTER (?x != ?y)
}
```



3. Change the query into an insert.

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
```

```
INSERT {?x h:seeAlso ?y}
WHERE {
?x h:shirtsize ?s .
?y h:shirtsize ?s
FILTER (?x != ?y)
}
```

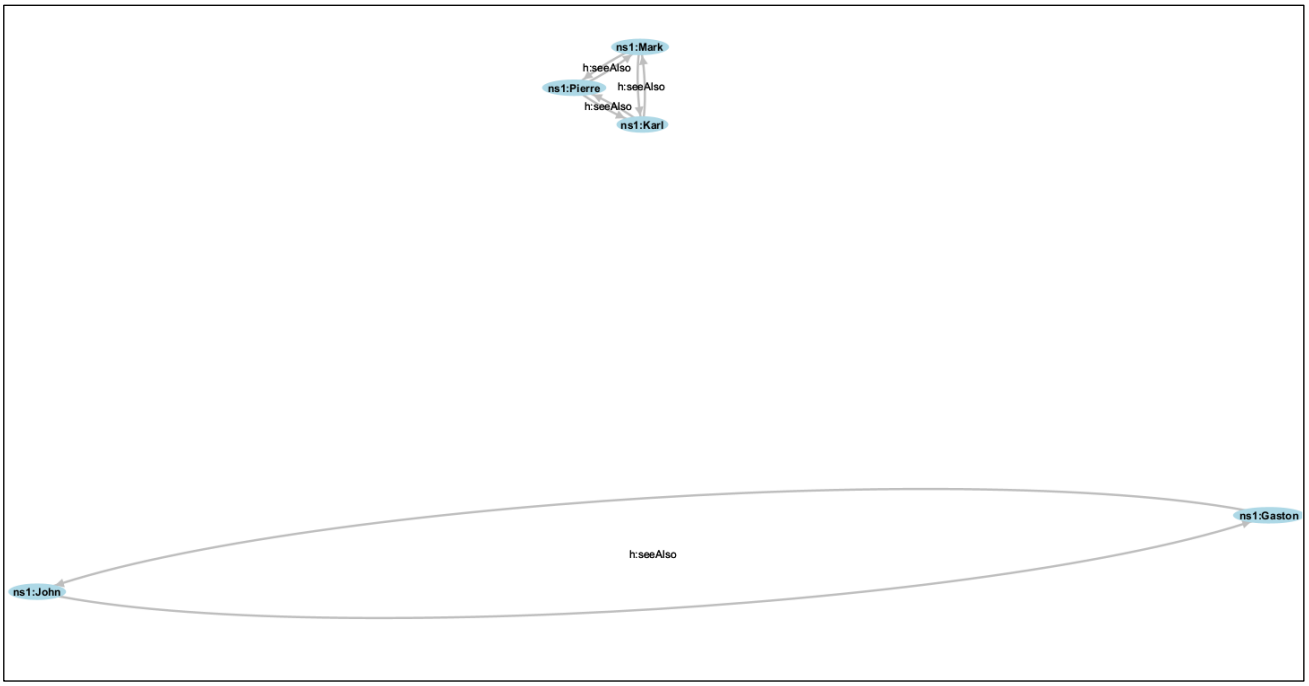
```
1  PREFIX h: <http://ns.inria.fr/humans/schema#>
2  INSERT {?x h:seeAlso ?y}
3  WHERE {
4  ?x h:shirtsize ?s .
5  ?y h:shirtsize ?s
6  FILTER (?x != ?y)
7  }
8
```

Graph | XML/RDF | Table | Validate

num	?x	?s	?y
1	<http://ns.inria.fr/humans/data#Pierre>	9	<http://ns.inria.fr/humans/data#Karl>
2	<http://ns.inria.fr/humans/data#Pierre>	9	<http://ns.inria.fr/humans/data#Mark>
3	<http://ns.inria.fr/humans/data#Gaston>	12	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#John>	12	<http://ns.inria.fr/humans/data#Gaston>
5	<http://ns.inria.fr/humans/data#Karl>	9	<http://ns.inria.fr/humans/data#Pierre>
6	<http://ns.inria.fr/humans/data#Karl>	9	<http://ns.inria.fr/humans/data#Mark>
7	<http://ns.inria.fr/humans/data#Mark>	9	<http://ns.inria.fr/humans/data#Pierre>
8	<http://ns.inria.fr/humans/data#Mark>	9	<http://ns.inria.fr/humans/data#Karl>

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

Screenshot:



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://ns.inria.fr/humans/schema#>
SELECT ?x ?y ?size
WHERE {
?x h:shoesize ?size .
?y h:shoesize ?size
FILTER (?x != ?y)
}

PREFIX h: <http://ns.inria.fr/humans/schema#>
```

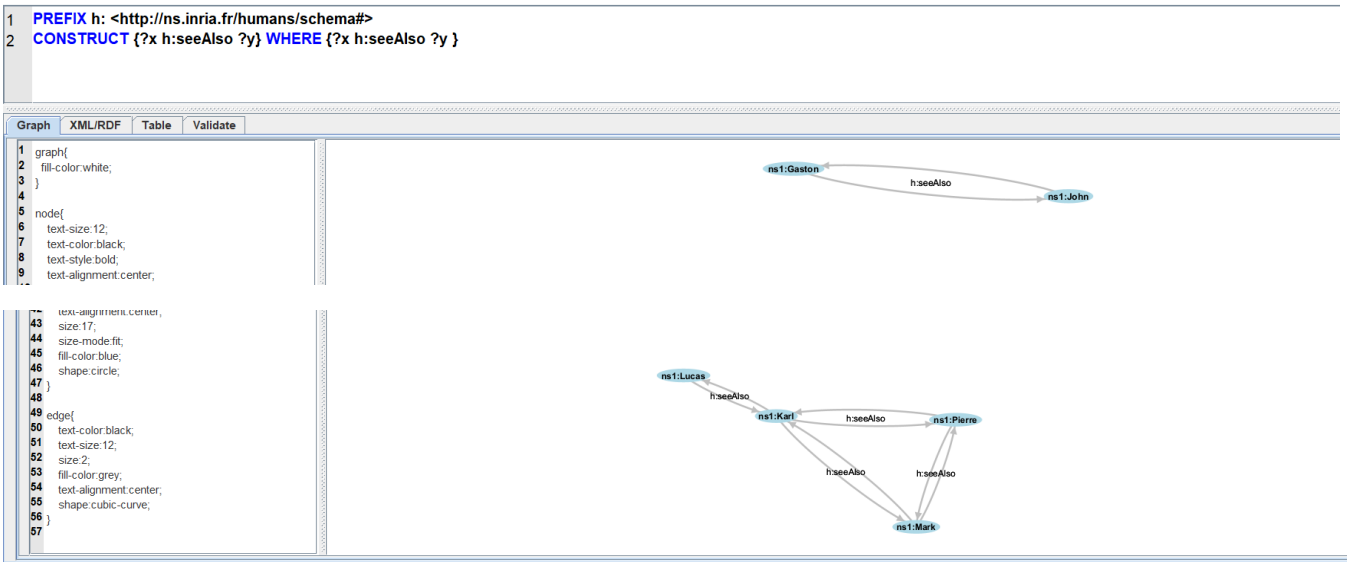
```
INSERT { ?x h:seeAlso ?y}
WHERE {?x h:shoesize ?size .
?y h:shoesize ?size
FILTER (?x != ?y)
}
```

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
CONSTRUCT {?x h:seeAlso ?y} WHERE {?x h:seeAlso ?y }
```

Screenshot:



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

Query:

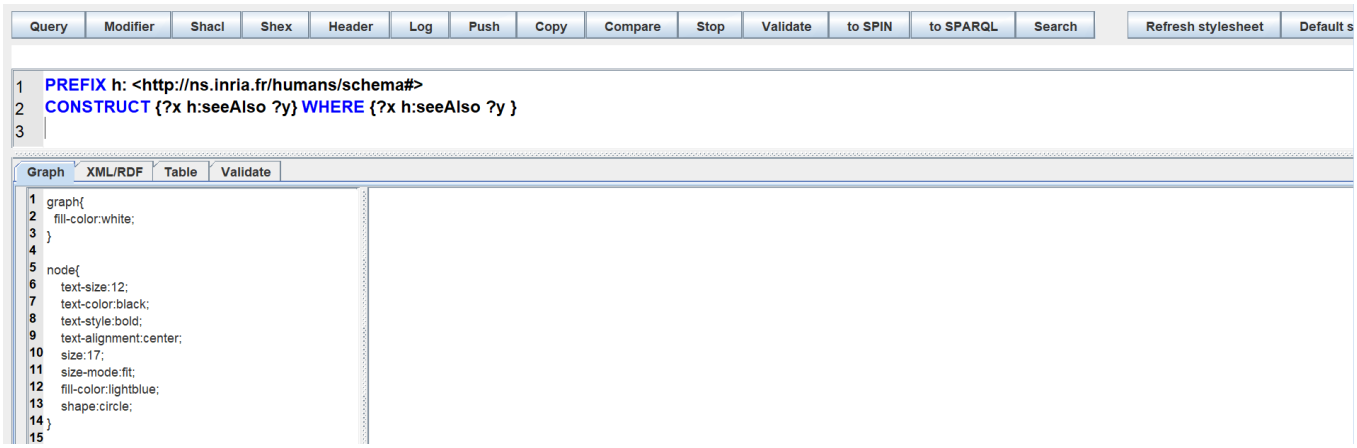
```
PREFIX h: <http://ns.inria.fr/humans/schema#>
 SELECT DISTINCT ?x ?y
 WHERE {
   ?x h:seeAlso* ?y .
   FILTER (?x != ?y)
 }
```

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT DISTINCT ?x ?y
WHERE {
  ?x h:seeAlso* ?y .
  FILTER (?x != ?y)
}
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Karl>
2	<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Lucas>
3	<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Mark>
4	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/data#John>
5	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Gaston>
6	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Pierre>
7	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Mark>
8	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Lucas>
9	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Karl>
10	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Pierre>
11	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Mark>
12	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Pierre>
13	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Karl>
14	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Lucas>

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

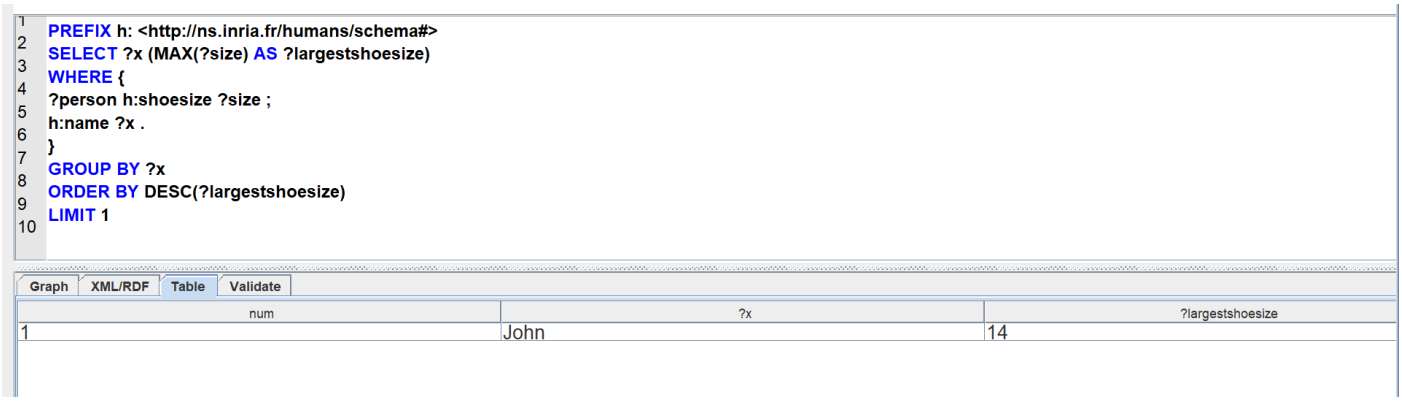


Question 12

1. Find the largest shoe size

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x (MAX(?size) AS ?largestshoesize)
WHERE {
?person h:shoesize ?size ;
h:name ?x .
}
GROUP BY ?x
ORDER BY DESC(?largestshoesize)
LIMIT 1
```



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

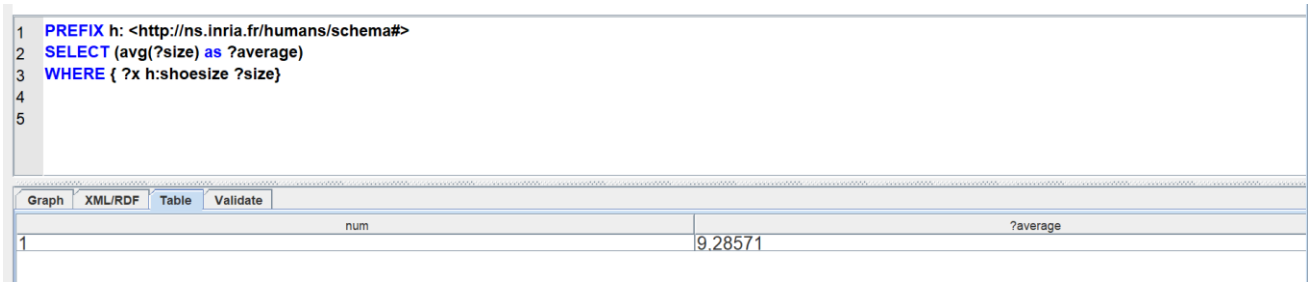
Query: **See above**

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

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT (avg(?size) as ?average)
```

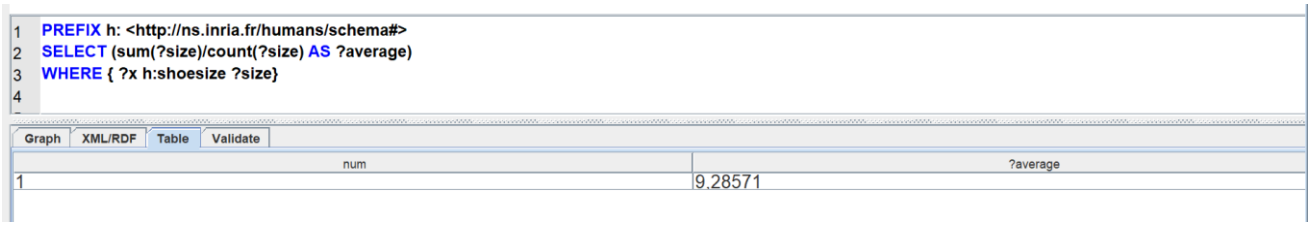
WHERE { ?x h:shoesize ?size}



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

Query:

PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT (sum(?size)/count(?size) AS ?average)
WHERE { ?x h:shoesize ?size}



Question 13

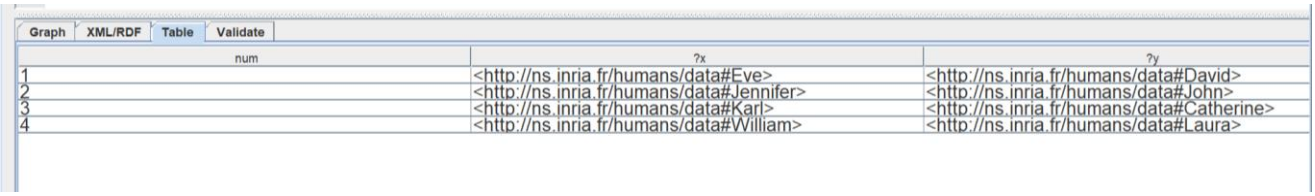
Find couples without children

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?y
WHERE {
?x h:hasSpouse ?y
    MINUS {
        {?x h:hasChild ?z}
        UNION {?y h:hasChild ?u}
        }
}
```

Or

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?y
WHERE {
  ?x h:hasSpouse ?y .
  OPTIONAL {?x h:hasChild ?z .}
  OPTIONAL {?y h:hasChild ?u .}
  FILTER (!bound(?z) && !bound(?u))
}
```



num	?x	?y
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#David>
2	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
3	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
4	<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>

Question 14

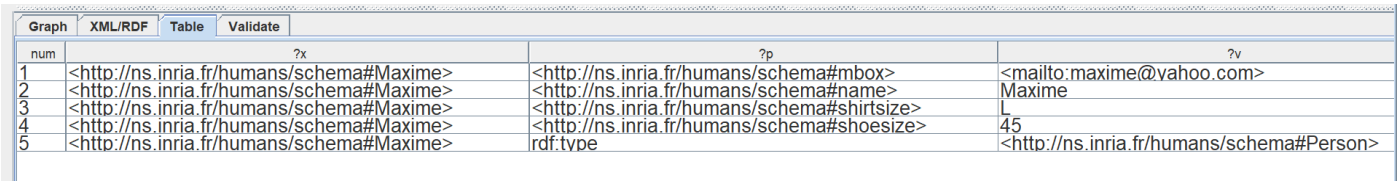
Using INSERT DATA, create a new person with its properties. Then, check that it has been created.

Insert:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
INSERT DATA {
  h:Maxime a h:Person ;
        h:name "Maxime" ;
        h:shoesize "45" ;
        h:shirtsize "L" ;
        h:mbox <mailto:maxime@yahoo.com> .
}
```

Screenshot result:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?p ?v
WHERE {
  ?x a h:Person ;
    h:name "Maxime" ;
    ?p ?v .
}
```



num	?x	?p	?v
1	<http://ns.inria.fr/humans/schema#Maxime>	<http://ns.inria.fr/humans/schema#mbox>	<mailto:maxime@yahoo.com>
2	<http://ns.inria.fr/humans/schema#Maxime>	<http://ns.inria.fr/humans/schema#name>	Maxime
3	<http://ns.inria.fr/humans/schema#Maxime>	<http://ns.inria.fr/humans/schema#shirtsize>	L
4	<http://ns.inria.fr/humans/schema#Maxime>	<http://ns.inria.fr/humans/schema#shoesize>	45
5	<http://ns.inria.fr/humans/schema#Maxime>	rdf:type	<http://ns.inria.fr/humans/schema#Person>

Question 15

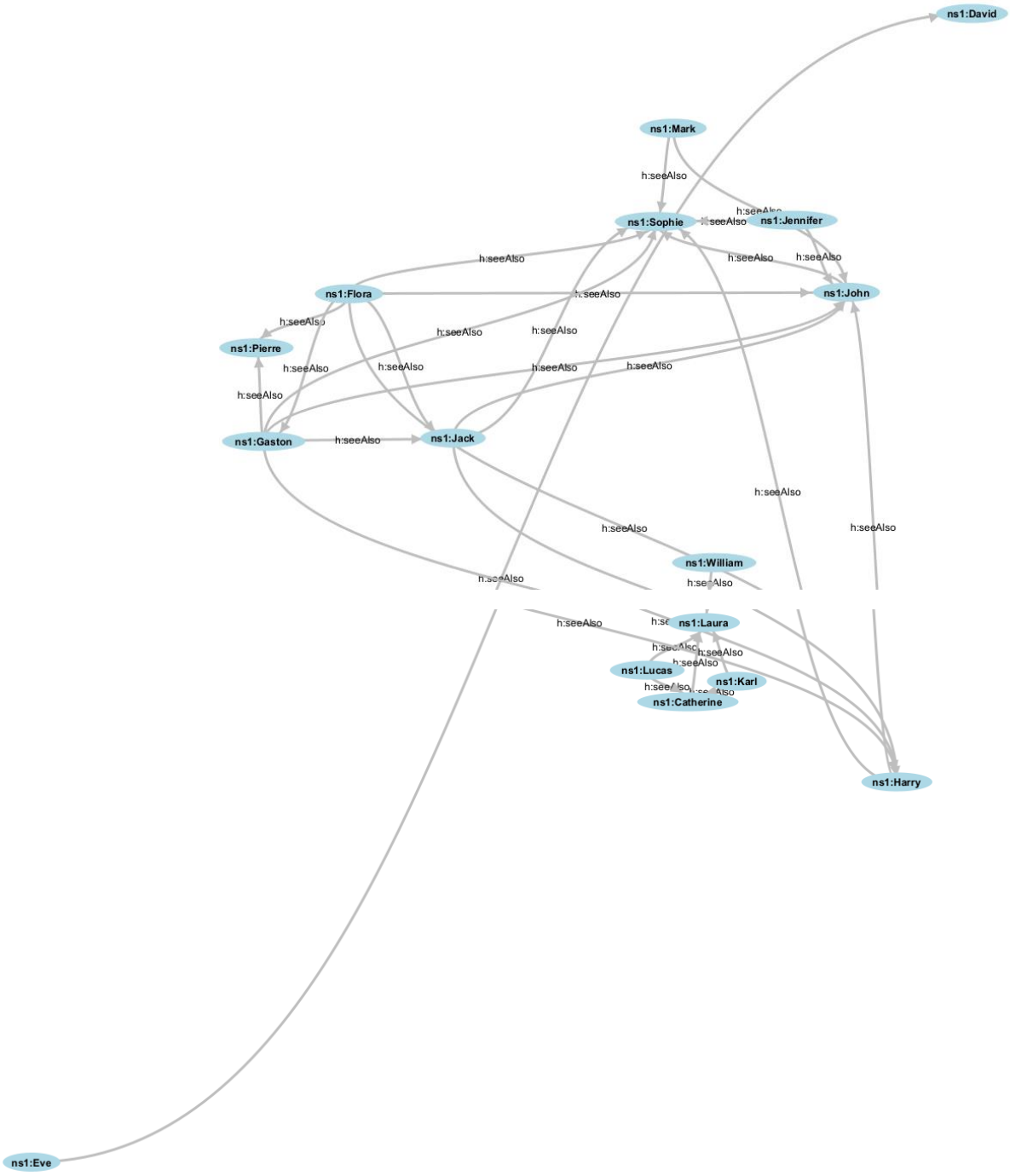
Find the persons connected by paths of any family links. Construct an arc seeAlso between them to visualize the result.

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
SELECT ?x ?y
WHERE { ?x (h:hasSpouse | h:hasChild | h:hasFather | h:hasMother | h:hasParent)+ ?y }
```

Screenshot:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
CONSTRUCT {?x h:seeAlso ?y }
WHERE { ?x (h:hasSpouse | h:hasChild | h:hasFather | h:hasMother | h:hasParent)+ ?y }
```



Question 16

Run the following query:

```
prefix db: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix h: <http://ns.inria.fr/humans/schema#>
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

This query constructs a graph that includes the names and spouse relationships of 20 French DBpedia persons.

Modify it to insert new persons in the base and check the results.

Query:

```
INSERT  { ?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
  }
}
```

Result:

The query was not executed successfully because the INSERT clause is performed on the local RDF dataset (not the dbpedia one) and the prefixes "h:", "db:", and "foaf:" used in the query are not defined in this local dataset.

Lab session on RDFS.

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 (Corese-GUI jar file): https://project.inria.fr/corese/

About the humans schema

1. If you don't have the human schema file yet use the command curl or wget to obtain the Turtle version of this schema from its namespace and download it in a file named "humans_schema.ttl" http://ns.inria.fr/humans/schema#

   curl -o humans_schema.ttl -L -H "Accept: text/turtle" http://ns.inria.fr/humans/schema#

2. What is the namespace of this ontology? How was it specified?

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

3. Locate the use of the terms of the RDF (S) language: Class, Property, label, comment, range, domain, subClassOf, subPropertyOf, etc. What are their namespaces?

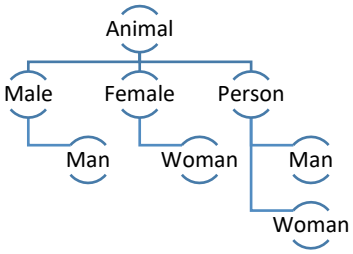
   The namespace of the RDFS language is associated to namespace:
   xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#

4. What are the classes of resources that can have the age property? Explain

   The age property has no domain or range specified, which means there are no restriction. Hence, any resource can use this property.

5. Look at the beginning of the file and draw the subgraph of the hierarchy containing the classes Animal, Man and Woman.

   :Animal a rdfs:Class
   :Person rdfs:subClassOf :Animal
   :Male rdfs:subClassOf :Animal
   :Female rdfs:subClassOf :Animal
   :Man rdfs:subClassOf :Male, :Person
   :Woman rdfs:subClassOf :Female, :Person



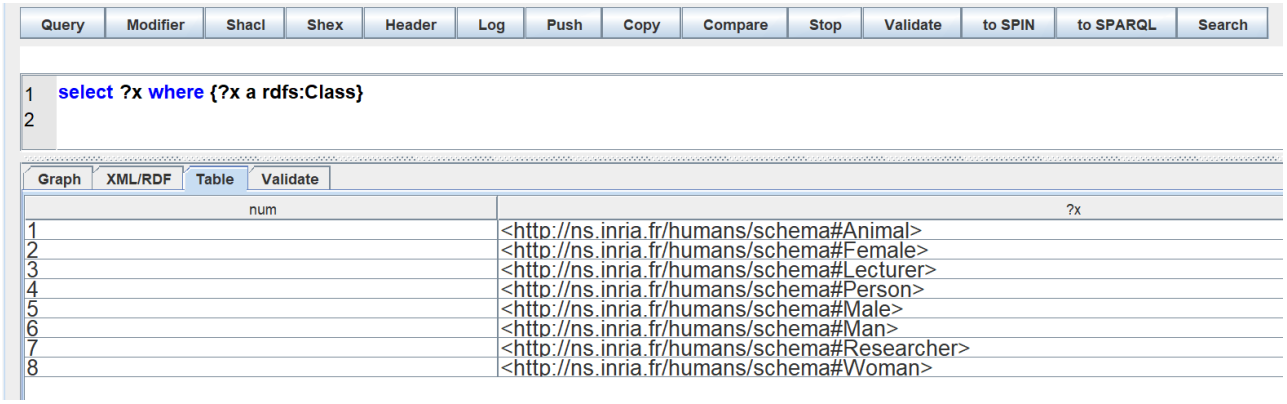
Query the schema itself

Reset or relaunch the standalone Corese search engine interface and load the file humans_**schema.**ttl (and only this one).

1. Write a query to find all the classes of the ontology.

Query:

```
SELECT ?x WHERE {?x a rdfs:Class}
```

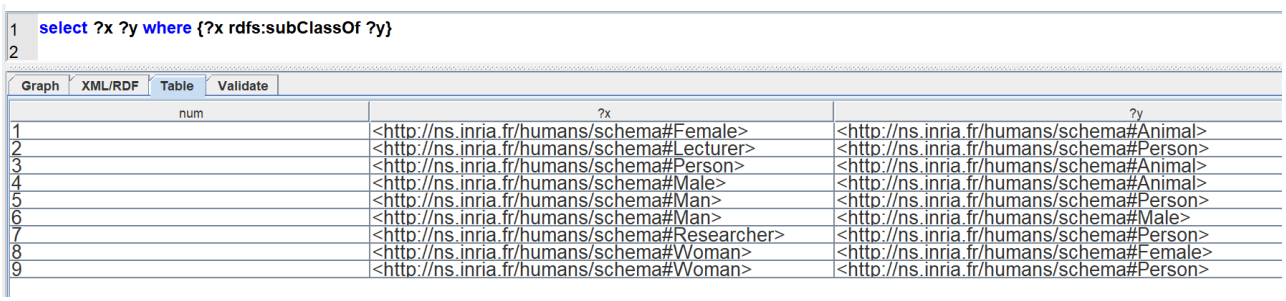


num	?x
1	<http://ns.inria.fr/humans/schema#Animal>
2	<http://ns.inria.fr/humans/schema#Female>
3	<http://ns.inria.fr/humans/schema#Lecturer>
4	<http://ns.inria.fr/humans/schema#Person>
5	<http://ns.inria.fr/humans/schema#Male>
6	<http://ns.inria.fr/humans/schema#Man>
7	<http://ns.inria.fr/humans/schema#Researcher>
8	<http://ns.inria.fr/humans/schema#Woman>

2. Write a query to find all the links subClassOf in the ontology.

Query:

```
SELECT ?x ?y WHERE {?x rdfs:subClassOf ?y}
```



num	?x	?y
1	<http://ns.inria.fr/humans/schema#Female>	<http://ns.inria.fr/humans/schema#Animal>
2	<http://ns.inria.fr/humans/schema#Lecturer>	<http://ns.inria.fr/humans/schema#Person>
3	<http://ns.inria.fr/humans/schema#Person>	<http://ns.inria.fr/humans/schema#Animal>
4	<http://ns.inria.fr/humans/schema#Male>	<http://ns.inria.fr/humans/schema#Animal>
5	<http://ns.inria.fr/humans/schema#Man>	<http://ns.inria.fr/humans/schema#Person>
6	<http://ns.inria.fr/humans/schema#Man>	<http://ns.inria.fr/humans/schema#Male>
7	<http://ns.inria.fr/humans/schema#Researcher>	<http://ns.inria.fr/humans/schema#Person>
8	<http://ns.inria.fr/humans/schema#Woman>	<http://ns.inria.fr/humans/schema#Female>
9	<http://ns.inria.fr/humans/schema#Woman>	<http://ns.inria.fr/humans/schema#Person>

3. Write a query to find the definitions and translations of "shoe size" (*other* labels and comments in different languages for the resource labeled "shoe size").

Query:

```
SELECT *
WHERE {
?x rdfs:label "shoe size"@en .
?x rdfs:label ?label
}
```

Answers:

```
select * where {
?x rdfs:label "shoe size"@en .
?x rdfs:label ?label
}
```

Graph | XML/RDF | Table | Validate

num	?x	?label
1	<http://ns.inria.fr/humans/schema#shoesize>	"size"@en
2	<http://ns.inria.fr/humans/schema#shoesize>	"shoe size"@en
3	<http://ns.inria.fr/humans/schema#shoesize>	"pointure"@fr

```
SELECT * WHERE {
  ?x rdfs:label ?label .
  ?x rdfs:label "shoe size"@en, ?label .
  FILTER (lang(?label) = "fr")
}
```

Graph | XML/RDF | Table | Validate

num	?x	?label
1	<http://ns.inria.fr/humans/schema#shoesize>	"pointure"@fr

4. Write a query to find the synonyms in French of the word 'personne' in French (*other* labels in the same language for the same resource/class/property). What are the answers?

Query:

SELECT *
WHERE {?x rdfs:label "personne"@fr. ?x rdfs:label ?y FILTER (lang(?y) = "fr")}

Answers:

```
select *
where {?x rdfs:label "personne"@fr. ?x rdfs:label ?y filter (lang(?y) = "fr" )}
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/schema#Person>	"homme"@fr
2	<http://ns.inria.fr/humans/schema#Person>	"humain"@fr
3	<http://ns.inria.fr/humans/schema#Person>	"personne"@fr
4	<http://ns.inria.fr/humans/schema#Person>	"être humain"@fr

5. Write a query to find the different meaning of the term "size" (disambiguation using the different comments attached to different resources/classes/properties having the label "size"). What are the answers?

Query:

SELECT *
WHERE {?x rdfs:label "size"@en . ?x rdfs:label ?y}

Answers:

```
select * where {?x rdfs:label "size"@en . ?x rdfs:label ?y}
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/schema#shirtsize>	"shirt size"@en
2	<http://ns.inria.fr/humans/schema#shirtsize>	"size"@en
3	<http://ns.inria.fr/humans/schema#shirtsize>	"taille"@fr
4	<http://ns.inria.fr/humans/schema#shirtsize>	"taille de chemise"@fr
5	<http://ns.inria.fr/humans/schema#shoesize>	"size"@en
6	<http://ns.inria.fr/humans/schema#shoesize>	"shoe size"@en
7	<http://ns.inria.fr/humans/schema#shoesize>	"pointure"@fr
8	<http://ns.inria.fr/humans/schema#trouserssize>	"size"@en
9	<http://ns.inria.fr/humans/schema#trouserssize>	"taille"@fr
10	<http://ns.inria.fr/humans/schema#trouserssize>	"trousers size"@en
11	<http://ns.inria.fr/humans/schema#trouserssize>	"taille de pantalon"@fr

6. Write a query to find the properties that use the class Person in their signatures?

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
 { ?x rdfs:domain h:Person }
 UNION
 { ?y rdfs:range h:Person }
}
```

```
1 PREFIX h: <http://ns.inria.fr/humans/schema#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 SELECT * WHERE {
4  { ?x rdfs:domain h:Person }
5  UNION
6  { ?y rdfs:range h:Person }
7 }
8
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/schema#hasFriend>	
2	<http://ns.inria.fr/humans/schema#hasSpouse>	
3	<http://ns.inria.fr/humans/schema#shirtsize>	
4	<http://ns.inria.fr/humans/schema#shoesize>	
5	<http://ns.inria.fr/humans/schema#trouserssize>	
6		<http://ns.inria.fr/humans/schema#hasFriend>
7		<http://ns.inria.fr/humans/schema#hasSpouse>

7. Make CORESE draw the graph of the hierarchy of Classes using a CONSTRUCT query considering only the classes in the humans schema

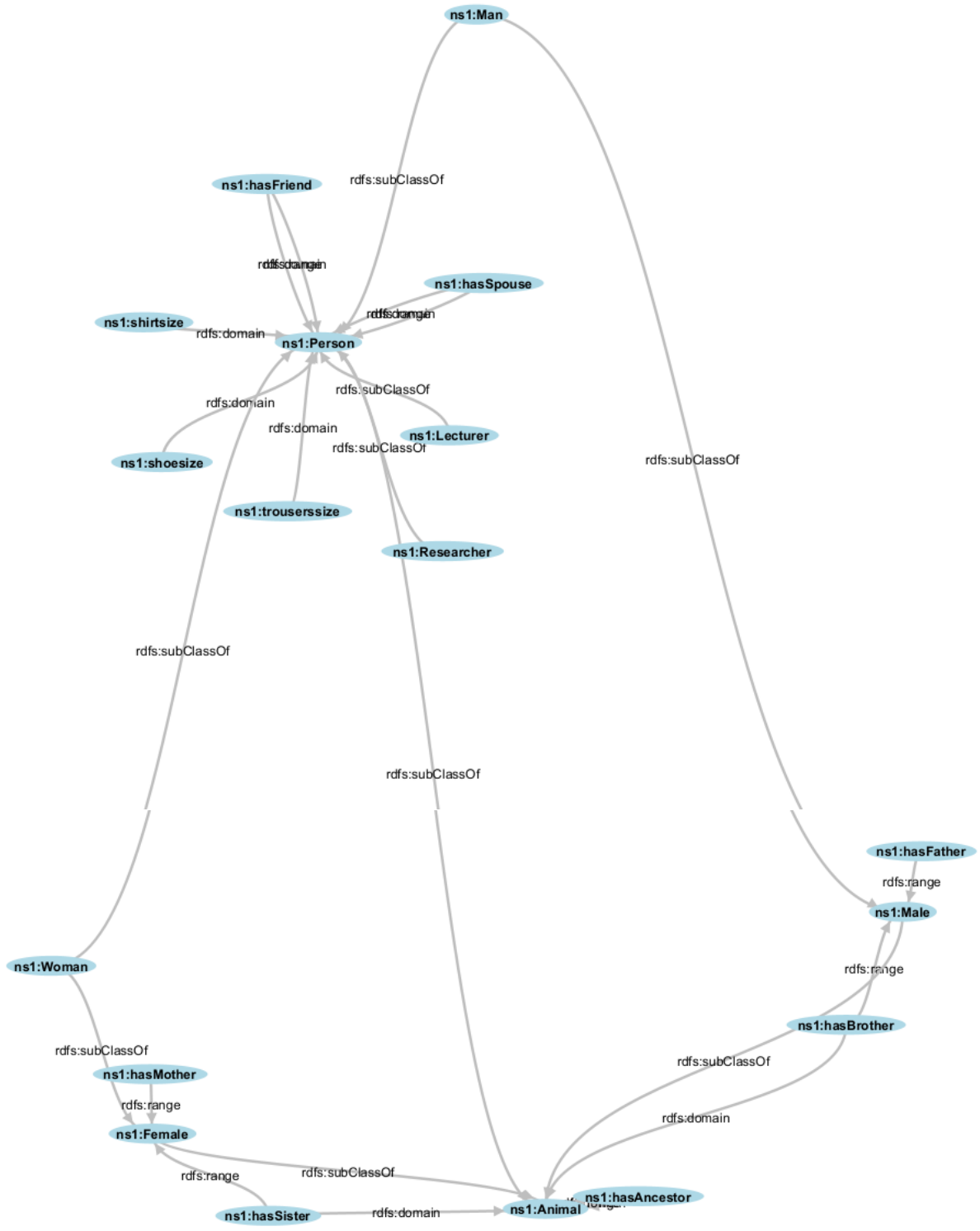
Query:

```
CONSTRUCT {
?x rdfs:range ?r
?x rdfs:domain ?d
?x rdfs:subClassOf ?y
}
WHERE {
{ ?x rdfs:range ?r}
UNION
{ ?x rdfs:domain ?d }
UNION
{ ?x rdfs:subClassOf ?y }
}
```

Answers:

Graph | XML/RDF | Table | Validate

num	?x	?r	?d	?y
1	<http://ns.inria.fr/humans/schema#hasBrother>	<http://ns.inria.fr/humans/schema#Male>		
2	<http://ns.inria.fr/humans/schema#hasFather>	<http://ns.inria.fr/humans/schema#Male>		
3	<http://ns.inria.fr/humans/schema#hasFriend>	<http://ns.inria.fr/humans/schema#Person>		
4	<http://ns.inria.fr/humans/schema#hasMother>	<http://ns.inria.fr/humans/schema#Female>		
5	<http://ns.inria.fr/humans/schema#hasSister>	<http://ns.inria.fr/humans/schema#Female>		
6	<http://ns.inria.fr/humans/schema#hasSpouse>	<http://ns.inria.fr/humans/schema#Person>		
7	<http://ns.inria.fr/humans/schema#hasAncestor>	<http://ns.inria.fr/humans/schema#Animal>		
8	<http://ns.inria.fr/humans/schema#hasBrother>		<http://ns.inria.fr/humans/schema#Animal>	
9	<http://ns.inria.fr/humans/schema#hasFriend>		<http://ns.inria.fr/humans/schema#Person>	
10	<http://ns.inria.fr/humans/schema#hasSister>		<http://ns.inria.fr/humans/schema#Animal>	
11	<http://ns.inria.fr/humans/schema#hasSpouse>		<http://ns.inria.fr/humans/schema#Person>	
12	<http://ns.inria.fr/humans/schema#shirtsize>		<http://ns.inria.fr/humans/schema#Person>	
13	<http://ns.inria.fr/humans/schema#shoesize>		<http://ns.inria.fr/humans/schema#Person>	
14	<http://ns.inria.fr/humans/schema#trouserssize>		<http://ns.inria.fr/humans/schema#Person>	
15	<http://ns.inria.fr/humans/schema#hasAncestor>		<http://ns.inria.fr/humans/schema#Animal>	
16	<http://ns.inria.fr/humans/schema#Female>			<http://ns.inria.fr/humans/schema#Animal>
17	<http://ns.inria.fr/humans/schema#Lecturer>			<http://ns.inria.fr/humans/schema#Person>
18	<http://ns.inria.fr/humans/schema#Person>			<http://ns.inria.fr/humans/schema#Animal>
19	<http://ns.inria.fr/humans/schema#Male>			<http://ns.inria.fr/humans/schema#Animal>
20	<http://ns.inria.fr/humans/schema#Man>			<http://ns.inria.fr/humans/schema#Person>
21	<http://ns.inria.fr/humans/schema#Man>			<http://ns.inria.fr/humans/schema#Male>
22	<http://ns.inria.fr/humans/schema#Researcher>			<http://ns.inria.fr/humans/schema#Person>
23	<http://ns.inria.fr/humans/schema#Woman>			<http://ns.inria.fr/humans/schema#Female>
24	<http://ns.inria.fr/humans/schema#Woman>			<http://ns.inria.fr/humans/schema#Person>

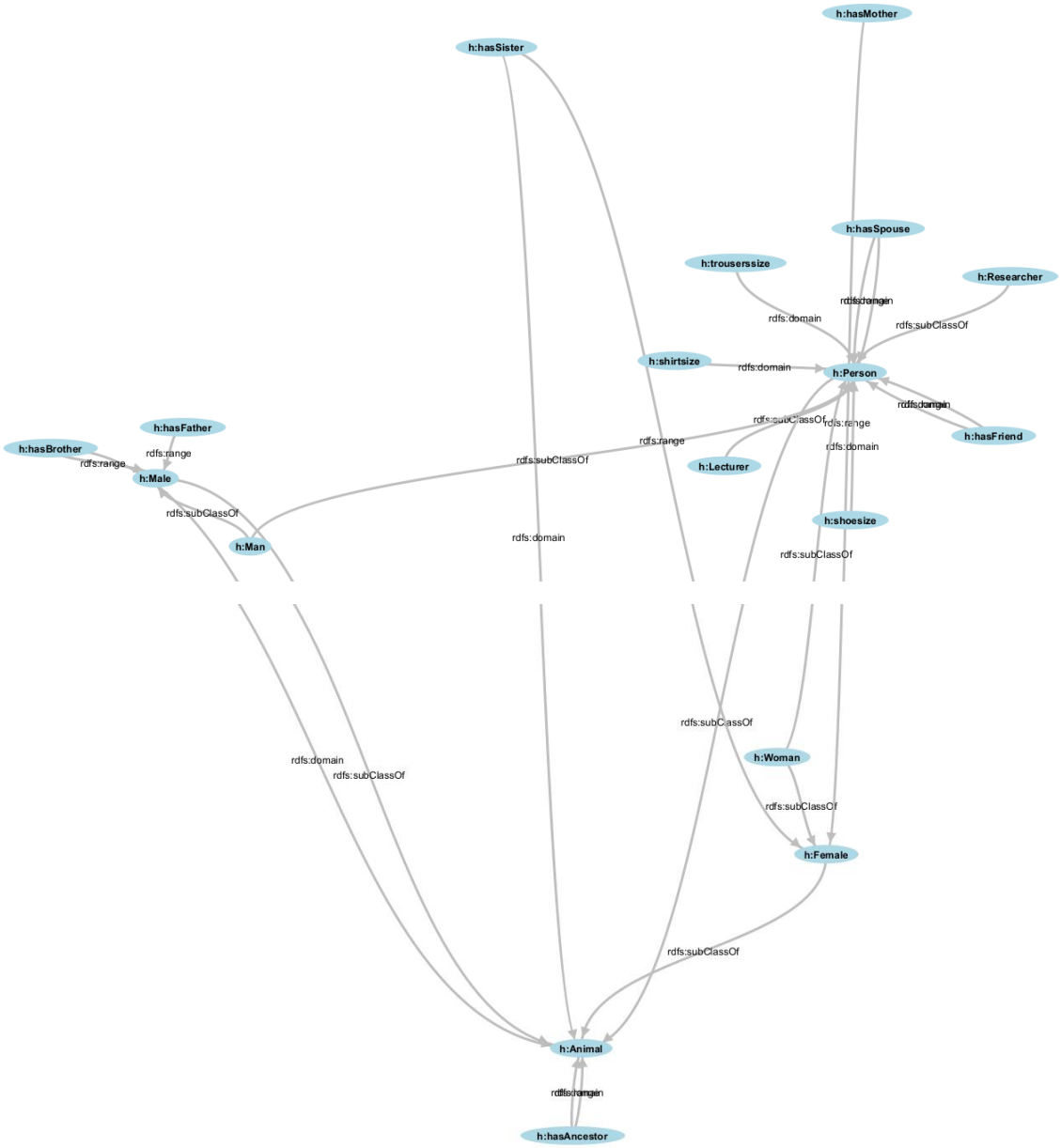


8. To the previous CONSTRUCT add the signatures of the relations.

Query:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT {?x rdfs:subClassOf ?y. ?z rdfs:domain ?domain. ?z rdfs:range ?range}
WHERE {
{?x rdfs:subClassOf ?y }
UNION {?z rdfs:domain ?domain.}
UNION {?z rdfs:range ?range}
}
```

Answers:



You now know how to query schemas on the semantic Web!

Query data augmented by an RDFS schema

Question 1

1. Reset the Corese engine and load only the data (humans_**data**.ttl)
2. Write a query to find the Persons.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT *
WHERE {?x a :Person}
```

Number of results before:

7

```
@prefix :     <http://ns.inria.fr/humans/schema#>
SELECT *
WHERE {?x a :Person}
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#David>
3	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Karl>
5	<http://ns.inria.fr/humans/data#Mark>
6	<http://ns.inria.fr/humans/data#William>
7	<http://ns.inria.fr/humans/data#Laura>

3. Load the schema (humans_**schema**.ttl)
4. Rerun the query to find the Persons and explain the result.

New number of results after and your explanation:

17 results because the schema file provides information about classes and sub-classes.

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT *
WHERE {?x a :Person}
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#Alice>
3	<http://ns.inria.fr/humans/data#David>
4	<http://ns.inria.fr/humans/data#Flora>
5	<http://ns.inria.fr/humans/data#Pierre>
6	<http://ns.inria.fr/humans/data#Gaston>
7	<http://ns.inria.fr/humans/data#Jennifer>
8	<http://ns.inria.fr/humans/data#John>
9	<http://ns.inria.fr/humans/data#Karl>
10	<http://ns.inria.fr/humans/data#Sophie>
11	<http://ns.inria.fr/humans/data#Catherine>
12	<http://ns.inria.fr/humans/data#Lucas>
13	<http://ns.inria.fr/humans/data#Mark>
14	<http://ns.inria.fr/humans/data#William>
15	<http://ns.inria.fr/humans/data#Laura>
16	<http://ns.inria.fr/humans/data#Harry>
17	<http://ns.inria.fr/humans/data#Jack>

Question 2

1. Write a query to find Males and their wives. How many answers do you get? Explain this result.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {
  ?x a :Male .
  ?x :hasSpouse ?s .
}
```

Number of results and explanation:

We got 1 result because according to the schema file, Man is a subclass of Male.

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {
  ?x a :Male .
  ?x :hasSpouse ?s .
}
```

Graph | XML/RDF | Table | Validate

num	?x	?s
1	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

2. In the data declare that Lucas has father Karl. Reset Corese, reload the ontology and the data, and then rerun the query to find Males and their wives. Explain the new result.

Line added in RDF:

```
d:Lucas a :Man ;
    :age 12 ;
    :hasMother d:Catherine ;
    :hasFather d:Karl;
    :name "Lucas" ;
    :shirtsize 8 ;
    :shoesize 7 ;
    :trouserssize 28 .
```

Number of results before and after and explanation:

```
1 PREFIX : <http://ns.inria.fr/humans/schema#>
2 SELECT * WHERE {
3   ?x a :Male .
4   ?x :hasSpouse ?s .
5 }
```

Graph | XML/RDF | Table | Validate

num	?x	?s
1	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
2	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

We now have 2 results because according to the schema file, the range of the "hasFather" property is the class ":Male", which means that the object of a "hasFather" triple must be an instance of the ":Male" class.

```
:hasFather a rdf:Property ;
    rdfs:label "has for father"@en, "a pour père"@fr ;
    rdfs:comment "to have for parent a male."@en,
                 "avoir pour parent un mâle."@fr ;
    rdfs:range :Male ;
    rdfs:subPropertyOf :hasParent .
```

Question 3

1. Write a query to find the Lecturers and their types. How many answers do you get? See how this typing is declared in the data and explain the result.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {?x a :Lecturer .}
```

Number of results and your explanation:

```
1 PREFIX : <http://ns.inria.fr/humans/schema#>
2 SELECT * WHERE {?x a :Lecturer .}
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#Laura>

2. Write a query to find common resources both of type Person and of type Male (instances of both classes). See how this typing is declared in the data and explain the presence of Jack.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT ?x WHERE {
  ?x rdf:type :Person, :Male .
}
```

Your explanation of the result: Jack is showed as a result because Jack is declared as a Man and Man is a subclass of Male.

```
1 PREFIX : <http://ns.inria.fr/humans/schema#>
2 SELECT ?x WHERE {
3   ?x rdf:type :Person, :Male .
4 }
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://ns.inria.fr/humans/data#Pierre>
2	<http://ns.inria.fr/humans/data#Gaston>
3	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Karl>
5	<http://ns.inria.fr/humans/data#Lucas>
6	<http://ns.inria.fr/humans/data#Harry>
7	<http://ns.inria.fr/humans/data#Jack>

Question 4

Write a query to find the hasAncestor relations. Explain the result after checking where this property is used in the data.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {
  ?x :hasAncestor ?y . }
```

Your explanation of the result:

```
1 PREFIX : <http://ns.inria.fr/humans/schema#>
2 SELECT * WHERE {
3   ?x :hasAncestor ?y .
4 }
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
2	<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
3	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Karl>
4	<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
5	<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>

We got all the hasFather, hasMother and hasParent relations because according to the schema file:hasFather and :hasMother are subPropertyOf :hasParent; and :hasParent is a subPropertyOf :hasAncestor .

Question 5

1. Write a query to find the family cores (couples and their children) using a SELECT

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {
  ?x :hasSpouse ?y .
  ?z :hasMother|:hasFather ?x .
}
```

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {
  ?x :hasSpouse ?y .
  ?z :hasMother|:hasFather ?x .
}
```

Graph | XML/RDF | Table | Validate

num	?x	?y	?z
1	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Lucas>

2. Modify it to display the result with a CONSTRUCT query

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
CONSTRUCT {
?x :hasSpouse ?y .
?z (:hasMother|:hasFather) ?x .
}
WHERE {
?x :hasSpouse ?y .
?z (:hasMother|:hasFather) ?x .
}
```

Question 6

1. Declare the olderThan relationship in the schema to indicate between two persons which is eldest and construct the arcs between persons with a SPARQL query

Addition to schema:

```
:olderThan a rdf:Property ;
    rdfs:label "is older than"@en, "est plus vieux que"@fr ;
    rdfs:comment "relation between an animal and another animal which is older than it."@en,
                 "relation entre un animal et un autre animal qui est plus vieux que lui."@fr ;
    rdfs:subPropertyOf :age ;
    rdfs:domain :Person ;
    rdfs:range :Person .
```

Query:

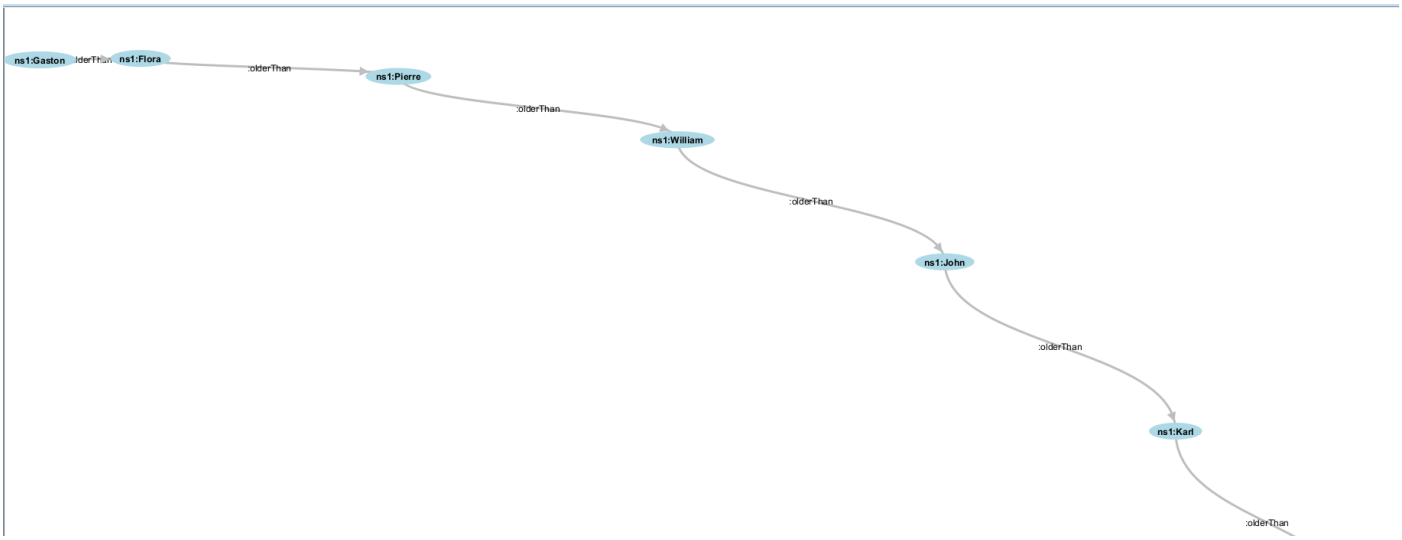
```
PREFIX : <http://ns.inria.fr/humans/schema#>
CONSTRUCT {
 ?person1 :olderThan ?person2
}
WHERE {
 ?person1 :age ?age1 .
 ?person2 :age ?age2 .
 FILTER(?age1 > ?age2)
}
```

2. Find a query that generates only the minimum number of links without redundancy with olderThan transitivity.

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
CONSTRUCT { ?person1 :olderThan ?person2 }
WHERE {
 ?person1 :age ?age1 .
 ?person2 :age ?age2 .
 FILTER(?age1 > ?age2)
```

MINUS {
?person1 :age ?age1 .
?person2 :age ?age2 .
?person3 :age ?age3
FILTER (?age1 >?age3 && ?age3> ?age2)}
}



Question 7

Write a query to find for John the properties which label contains the string "size" and the value of these properties.

Query:

SELECT ?label ?value
WHERE { <http://ns.inria.fr/humans/data#John> ?p ?value .
?p rdfs:label ?label
}

```
1  SELECT ?label ?value
2  WHERE { <http://ns.inria.fr/humans/data#John> ?p ?value .
3  ?p rdfs:label ?label
4  }
```

Graph | XML/RDF | Table | Validate

num	?label	?value
1	"age"@en	37
2	"âge"@fr	37
3	"has for ancestor"@en	<http://ns.inria.fr/humans/data#Sophie>
4	"a pour ancêtre"@fr	<http://ns.inria.fr/humans/data#Sophie>
5	"has for parent"@en	<http://ns.inria.fr/humans/data#Sophie>
6	"a pour parent"@fr	<http://ns.inria.fr/humans/data#Sophie>
7	"name"@en	John
8	"nom"@fr	John
9	"shirt size"@en	12
10	"size"@en	12
11	"taille"@fr	12
12	"taille de chemise"@fr	12
13	"size"@en	14
14	"shoe size"@en	14
15	"pointure"@fr	14
16	"size"@en	44
17	"taille"@fr	44
18	"trousers size"@en	44
19	"taille de pantalon"@fr	44

Question 8

Use the ontology to document your answers in natural language: write a query to find the types and properties of Laura in French.

Query:

```
SELECT ?class ?propertylabel ?value
WHERE {
{ <http://ns.inria.fr/humans/data#Laura> a ?t}
UNION
{<http://ns.inria.fr/humans/data#Laura> ?t ?value }

?t rdfs:label ?propertylabel

FILTER(LANGMATCHES(LANG(?propertylabel), "fr"))
}
```

```
1  SELECT ?class ?propertylabel ?value
2  WHERE {
3  { <http://ns.inria.fr/humans/data#Laura> a ?t}
4  UNION
5  {<http://ns.inria.fr/humans/data#Laura> ?t ?value }
6
7  ?t rdfs:label ?propertylabel
8
9  FILTER(LANGMATCHES(LANG(?propertylabel), "fr"))
10 }
```

Graph | XML/RDF | Table | Validate

num	?class	?propertylabel	?value
1		"professeur"@fr	
2		"homme"@fr	
3		"humain"@fr	
4		"personne"@fr	
5		"être humain"@fr	
6		"chercheur"@fr	
7		"scientifique"@fr	
8		"animal"@fr	
9		"femelle"@fr	
10		"a pour ami"@fr	<http://ns.inria.fr/humans/data#Alice>
11		"nom"@fr	Laura

Create your own schema Family schema (can be done after the OWL practical session too if you are running out of time)

- Write the RDF schema that you used in the description of Jen in a RDFS Turtle (or in RDF/XML and then translate it) and save the RDFS Turtle in a file called “Family_schema.ttl” (or “Family_schema.xml”). Of course, this assumes that the URIs for the classes and properties declared/used must match in both files. You may have to update the files Jen.rdf and Jen.ttl to use your ontology.

Your schema:

<ANSWER HERE/>

- Check that your RDF schema and RDF files are valid using the W3C's RDF validation service or other converter/ translators services.
- Launch the standalone interface of Corese and load your files Family_schema.ttl and Jen.ttl
- The interface contains a default SPARQL query:
  Select ?x ?t where {?x rdf:type ?t}
  Launch the query and look at the results.

Screenshot:

<ANSWER HERE/>

- Modify your ontology to declare the classes of Man and Woman as sub classes of Human (don't change the data), reload the schemas and data and search for the humans to see the results

Screenshot:

<ANSWER HERE/>

Explanation:

<ANSWER HERE/>

- Modify your ontology to declare the properties hasChild and hasSpouse as sub properties of familyLink (don't change the data), reload the schemas and data and search for the family links to see the results.

Screenshot:

<ANSWER HERE/>

Explanation:

<ANSWER HERE/>

- Modify your ontology to declare the class FamilyMember and use it to specify the signature of the property familyLink (don't change the data) then reload the schemas and data and search for the family members.

Screenshot:

<ANSWER HERE/>

Explanation:

<ANSWER HERE/>

Lab session on OWL.

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 (Corese-GUI jar file): https://project.inria.fr/corese/

A, Query data augmented by an OWL schema

Make a copy of the humans_schema.ttl file, name it humans_owl_schema.ttl and use it for the rest of the session. For each of the following statements, specify a SPARQL query that shows that the difference before and after running the OWL inferences: you will find that answers to these queries are different depending on whether you load the ontology humans_schema.ttl or the humans_owl_schema.ttl you modified.

Important: The "Engine Menu" allows you to control and witness the result of the inferences. If nothing is selected and if you run no rules you will just have the graph you loaded. If you start applying RDFS or OWL you will see new inferred results being added. For this practical session make sure to apply (unselect and reselect) "OWL RL" in the engine menu of Corese to run the rules to see the addition of results. Depending on the version of the CORESE-GUI you use you may have to repeat this operation several times to see all results.

1. Declare that hasSpouse is a symmetrical property and do the same for hasFriend .

Code added to the schema:

```
<owl:SymmetricProperty rdf:ID="hasSpouse">
</owl:SymmetricProperty>
<owl:SymmetricProperty rdf:ID="hasFriend">
</owl:SymmetricProperty>
```

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#>
SELECT * WHERE {?x :hasSpouse ?y}
```

Result before addition to the schema:

```
1 PREFIX : <http://ns.inria.fr/humans/schema#>
2 SELECT * WHERE {?x :hasSpouse ?y}
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#David>
2	<http://ns.inria.fr/humans/data#Flora>	<http://ns.inria.fr/humans/data#Gaston>
3	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
5	<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>
6	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

Result after addition to the schema (reload then unselect and reselect "OWL RL"):

<ANSWER HERE/>

Explanation:

<ANSWER HERE/>

2. Declare that hasChild is the inverse property of the hasParent property.

Code added to the schema:

<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

3. Declare hasAncestor as transitive property.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

4. Declare and define the chain property hasSibling has a super-property of the existing properties hasBrother and hasSister.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

5. Declare and define the chain properties: hasUncle and hasAunt and in the data declare that Jack and Pierre are brothers and vice-versa.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Result after addition to the schema:
<ANSWER HERE/>

6. Declare the disjunction between Male and Female. Violate the constraint in the data, check the results and then remove the violation you created.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

7. Declare that the class Professor is the intersection of the class Lecturer and Researcher class.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

8. Declare that the Academic class is the union of classes Lecturer and Researcher.

Code added to the schema:

<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

9. Create a class Organization and its sub class University. Create a new property mainEmployer, with domain Person and range Organization. Use a restriction to declare that any Professor has for main employer a University.

```
:Professor rdf:type owl:Class;
    owl:intersectionOf (:Lecturer :Researcher);
    rdfs:subClassOf
            [a owl:Restriction ;
            owl:onProperty :mainEmployer;
            owl:hasValue :University ]
```

Code added to the schema (new property, new classes and new restriction):
<ANSWER HERE/>

Code added to the data (just declare the main employer of a Professor):
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

10. Use a restriction to declare that any person must have a parent who is a woman. For this last statement, you need to run the rule engine after loading the ontology and data.

Code added to the schema:
<ANSWER HERE/>

Query:
<ANSWER HERE/>

Result before addition to the schema:
<ANSWER HERE/>

Result after addition to the schema (reload then unselect and reselect “OWL RL”):
<ANSWER HERE/>

Explanation:
<ANSWER HERE/>

B, Make your own OWL models:

For each one of the following OWL primitives imagine a definition that could use it and provide that definition in OWL using your preferred syntax (RDF/XML or N3/Turtle). For instance a possible definition using owl:TransitiveProperty would be a definition of the Ancestor property. For each primitive in the following list you imagine the definition of a class or property that was not given in the course and you give that definition in English and in OWL.

1. owl:oneOf <YOUR EXAMPLE HERE/>
2. owl:unionOf <YOUR EXAMPLE HERE/>
3. owl:intersectionOf <YOUR EXAMPLE HERE/>
4. owl:complementOf <YOUR EXAMPLE HERE/>
5. owl:disjointWith <YOUR EXAMPLE HERE/>
   or owl:AllDisjointClasses
   or owl:disjointUnionOf
6. owl:ObjectProperty <YOUR EXAMPLE HERE/>
7. owl:DatatypeProperty <YOUR EXAMPLE HERE/>
8. owl:SymmetricProperty <YOUR EXAMPLE HERE/>
   or owl:AsymmetricProperty
9. owl:inverseOf <YOUR EXAMPLE HERE/>
10. owl:TransitiveProperty <YOUR EXAMPLE HERE/>
11. owl:propertyDisjointWith <YOUR EXAMPLE HERE/>
12. owl:ReflexiveProperty <YOUR EXAMPLE HERE/>
    or owl:IrreflexiveProperty
13. owl:propertyChainAxiom <YOUR EXAMPLE HERE/>
14. owl:FunctionalProperty <YOUR EXAMPLE HERE/>
15. owl:InverseFunctionalProperty <YOUR EXAMPLE HERE/>
16. owl:hasKey <YOUR EXAMPLE HERE/>
17. owl:allValuesFrom <YOUR EXAMPLE HERE/>
18. owl:someValuesFrom <YOUR EXAMPLE HERE/>
19. owl:hasValue <YOUR EXAMPLE HERE/>
20. owl:maxCardinality <YOUR EXAMPLE HERE/>
    or owl:minCardinality
21. owl:qualifiedCardinality <YOUR EXAMPLE HERE/>