Jeanne Barthélemy

Semantic Web and Information Extraction Technologies
Practice SPARQL with CORESE and the Open Web

- **Time spent for this assignment :** I am not sure, but it surely have been more than 10 hours
- **Questions I have not been able to do :** None, I tried to do all of the questions, and I proposed a query at each time. However, I got some weird results for some of them, therefore there are surely some mistakes in my queries (for instance, I got different different universities for "lists the top 10 Universities with most winners of the Nobel Prize in Physics" with wikidata and dbpedia

----part I----

- 1. For instances, the namespace is :
  <http://www.inria.fr/2007/09/11/humans.rdfs-instances#ID>
  with for example ID = Jack (the id of the instance)

- 2. For the humans schema (the object/properties), the namespace is :
  <http://www.inria.fr/2007/09/11/humans.rdfs#ID>
  with for example rdf:ID="Animal" (ID is the ID of a specific object/property)

- 3. Using the turtle syntax, we can tell everything about John with :
  @prefix humans: <http://www.inria.fr/2007/09/11/humans.rdfs#> .
  <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
  a humans:Person ;
  humans:age "37" ;
  humans:hasParent <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie> ;
  humans:name "John" ;
  humans:shirtsize "12" ;
  humans:shoesize "14" ;
  humans:trouserssize "44" .
  It was validated by this website: http://ttl.summerofcode.be/

----part II----

- 1. _

- 2. The query :
  ```
  select ?x ?t where
  {
  ?x rdf:type ?t
  }
  ```
  returns all kinds of tuples (x,t) where x if of type t
  I get 69 answers. John if of the types "Animal", "Male" and "Person"

- 3. It returns all of the tuples (x,y) where y is indicated to be the spouse of x, from humans.rdfs
  I got 6 answers.

- 4. The RDF property used is the one with the id **shoesize** (the namespace is <http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>)

- 5. The query is :
  ```
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT ?x ?y
  WHERE
  {
  ?x humans:shoesize ?y
  }
  ```
  I got 7 answers

- 6. The query is :
  ```
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT ?x ?y
  WHERE
  {
  ?x rdf:type humans:Person
  optional{?x humans:shoesize ?y}
  }
  ```
  I got 17 answers

- 7. The query is :
  ```
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT ?x
  WHERE
  {
  ?x humans:shoesize ?y
  ```

FILTER (xsd:integer(?y) >= 8)
}
I got 5 results

- 8. The query is :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT distinct ?x
  WHERE{
  {
  ?x humans:shoesize ?y
  FILTER (xsd:integer(?y) >= 8)
  }
  UNION
  {
  ?x humans:shirtsize ?z
  FILTER (xsd:integer(?z) >= 12)
  }
  }
  I got 5 results

- 9. The query is :
  DESCRIBE <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>

- 10. The query is :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT ?x
  WHERE
  {
  ?x humans:hasChild ?y
  }
  5 results, two duplicates (as Gaston has 2 children).
  Revised query without duplicates :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT distinct ?x
  WHERE
  {
  ?x humans:hasChild ?y
  }
  4 results.

- 11. The query is :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT distinct ?x

```
WHERE
{
?x rdf:type humans:Male
filter (! EXISTS {
?x humans:hasChild ?y
})
}
```
I got 3 results with this query, but only two with the one (more accurate) below:
```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT distinct ?x
WHERE
{
?x rdf:type humans:Male
filter (! EXISTS {{
?x humans:hasChild ?y
}union{?z humans:hasFather ?x}})
}
```
Indeed, there often is missing data, that we have to look for somewhere else. (it is not written that John has a child, but there is someone that has a father which is John.

- 12. The query is :
```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE
{
?x humans:age ?y
FILTER (xsd:integer(?y) >= 100)
}
```
I got 1 result (Gaston)

- 13. The query is :
```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?z ?y
WHERE
{
?x humans:shirtsize ?y
?z humans:shirtsize ?y
FILTER (xsd:string(?x) != xsd:string(?z))
}
```
I got 8 results. If we remove the filter, everyone with a shirtsize will be displayed (as equal to her/himself).

- 14. The query is :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT ?x
  WHERE
  {
  ?x rdf:type humans:Animal
  filter (! exists{?x rdf:type humans:Male})
  }
  I got 11 results. Every instance is a subclass of Animal.
  It can indeed be seen from the description that none of those individuals are linked to the type "Male".

----part III----

- 1. _

- 2. The query is :
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT DISTINCT ?s ?type
  WHERE {
  ?s a ?type
  FILTER (?type IN (rdfs:Class))
  }
  This query gives all of the classes (but not the properties).

- 3. This query below lists all of the s,d tuples such as s is a subClass of d (with d a class, not a **property).**
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  SELECT DISTINCT ?s ?d
  WHERE {
  ?s rdfs:subClassOf ?d
  ?d a ?type
  FILTER (?type IN (rdfs:Class))
  }

- 4. This gives back the translation and definition for shoesize:
  PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
  select *
  where {
  humans:shoesize rdfs:label ?label
  humans:shoesize rdfs:comment ?comment

```
 FILTER (langMatches( lang(?label), "FR" ) )
}
```

- 5. This query gives the 4 synonyms:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
select  *
where {
humans:Person rdfs:label ?label .
 FILTER (langMatches( lang(?label), "FR" ) )
}
```

----part IV----

- 1. The query is :

```
PREFIX db: <http://dbpedia.org/ontology>
SELECT (count(?x) AS ?Num_class)  (count(?y) AS ?Num_dataProperties)(count(?z) AS
?Num_objProperties) WHERE {
{?x rdf:type owl:Class }
UNION {?y rdf:type owl:DatatypeProperty}
UNION {?z rdf:type owl:ObjectProperty }
}
```

  This query returns me:
  Num_class: 760;   Num_dataProperties: 1760; Num_objProperties: 1105

- 2. The query is :

```
SELECT DISTINCT ?x ?y ?a  WHERE {
?x rdf:type dbo:Person.
?y rdf:type skos:Concept.
FILTER (?y = dbc:Nobel_laureates_in_Physics).
Optional{
?x dbo:birthDate ?a.
FILTER(strlen(str(?a)) = 10).}
  ?x dct:subject ?y.
}
ORDER BY DESC(xsd:date(?a))
```

  I had to filter the size of the date, as I had weird results with different types of dates (even though considered as the same datatype)

- 3. The query is :

```
SELECT DISTINCT ?u (COUNT(?p) AS ?number)
WHERE {
?u rdf:type dbo:University .
?p dbp:workInstitutions ?u .
?y rdf:type skos:Concept.
FILTER (?y = dbc:Nobel_laureates_in_Physics).
?p dct:subject ?y.
}
GROUP BY (?u)
ORDER BY DESC(?number)
LIMIT 10
```

- 4. The query is :

```
SELECT  count(?x)
WHERE {
?x rdf:type dbo:Person.
?y rdf:type skos:Concept.
FILTER (?y = dbc:Nobel_laureates_in_Physics).
?x dct:subject ?y.
?x dbo:birthPlace ?z.
?u rdf:type dbo:University .
?x dbp:workInstitutions ?u .
?u dbo:country ?i
FILTER (?i != ?z)
}
```

I got a result of 54

- 5. The query is :

```
SELECT ?x  WHERE {
   ?s rdf:type skos:Concept .
   FILTER (?s = dbc:Bruce_Springsteen_songs) .
   ?x dct:subject ?s .
   ?x dbo:releaseDate ?y .
   BIND (year(xsd:date(?y)) as ?z ) .
   FILTER (?z >= 1980 && ?z <= 1990)
}
```

----part V----

- 1. The query is :

```
SELECT DISTINCT  ?winner ?date WHERE {
  ?winner wdt:P31 wd:Q5 ;
          wdt:P166 wd:Q38104;
```

```
        wdt:P569 ?date .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
ORDER BY DESC(?date)
```

We have: wd:Q5 for human, wdt:P31 for instance of, wd:Q38104 is the nobel prizes of physics, wdt:P166 means that the human owns it.
wdt:P569 is for the birth date

- 2. The query is :

  ```
  SELECT DISTINCT ?u (COUNT(?p) AS ?number)
  WHERE {
  ?u wdt:P31 wd:Q3918.
  ?p wdt:P31 wd:Q5 ;
    wdt:P166 wd:Q38104.
  ?p wdt:P108 ?u .
  }
  GROUP BY (?u)
  ORDER BY DESC(?number)
  LIMIT 10
  ```

  Here, wd:Q3918 is the identifier for a university. wdt:P108 is the identifier for "employer" (as a property)
  But I got very strange results.

- 3.

  ```
  SELECT  (count( DISTINCT ?p) as ?total)
  WHERE {
  ?p wdt:P31 wd:Q5 ;
    wdt:P166 wd:Q38104.
  ?u wdt:P31 wd:Q3918.
  ?p wdt:P108 ?u .

  ?p wdt:P27 ?z.

  ?u wdt:P17 ?i.
  FILTER (?i != ?z).
  }
  ```

  Here wdt:P27 is for the birthplace and wdt:P17 the country. I got a result of 70 , therefore it seems that there is a mistake (as I got also different results from the same queries with the dbpedia).

----part VI----

Just below are my real answers, but I also added in the end my failed attempts to make queries work with wikidata :

- 1. The query is :
  ```
  SELECT ?number
  WHERE
  {
    ?number primefactor 2.
  }
  ```
  Or, another query could be:
  ```
  SELECT ?number
  WHERE
  {
    ?number primefactor http://km.aifb.kit.edu/projects/numbers/n2.
  }
  ```
  (As http://km.aifb.kit.edu/projects/numbers/n2 is the identifier of the number 2 (URI))

- 2. The query is :
  ```
  SELECT ?number ?prime
  WHERE
  {
    ?number primefactor ?prime.
    ?number previous ?prime.
  }
  ```

- 3. The query is :
  ```
  SELECT ?odd
  WHERE
  {
    ?odd next ?even.
    ?even primefactor http://km.aifb.kit.edu/projects/numbers/n2.
  }
  ```

- 4. The query is :
  ```
  SELECT distinct ?number
  WHERE
  {
   ?number primefactor ?prime.
   FILTER(?number = ?prime)
  }
  ```

By definition, if a number has a prime number that is itself, it is a prime number.

- 5. The query is :
  SELECT distinct ?number
  WHERE
  {
  ?number primefactor ?prime.
  FILTER (?prime != 1).
  FILTER(?number != ?prime)
  }
  Or, another query could be:
  SELECT ?number
  WHERE
  {
  ?number primefactor ?prime.
  FILTER (?prime != http://km.aifb.kit.edu/projects/numbers/n1).
  FILTER(?number != ?prime)
  }
  By definition, if a number has a prime factor that is not itself (letting the number 1 aside), then the number is not prime. We have http://km.aifb.kit.edu/projects/numbers/n1 the identifier of 1.

- 6. The query is :
  SELECT ?number1 ?number2
  WHERE
  {
  ?number1 primefactor ?prime1.
  FILTER(?number1 = ?prime1).

  ?number2 primefactor ?prime2.
  FILTER(?number2 = ?prime2).

  ?number1 next ?y.
  ?y next ?number2.
  }

}

[Below are attempts on wikidata (I never expected to get all results, as there is an infinite number of natural numbers, but at least I expected to get more results for some queries.) It is not my official answer for problem 6.

1.SELECT ?number
WHERE
{
  ?number wdt:P31 wd:Q21199.
  ?number wdt:P5236 wd:Q200.
}
or just
SELECT ?number
WHERE
{
  ?number wdt:P5236 wd:Q200.
}
or, if we want directly the number values, as wdt:P1181 is the value (for the number):
SELECT ?num
WHERE
{?number wdt:P1181 ?num.
  ?number wdt:P5236 wd:Q200.
}
Indeed, wdt:P5236 = primefactor, wd:Q200= number 2, wd:Q21199 = natural number. We therefore have the
numbers divisible by 2, which are the even numbers.

2.
SELECT ?number ?prime
WHERE
{
  ?number wdt:P31 wd:Q21199.
  ?prime wdt:P31 wd:Q49008 .
  ?number wdt:P5236 ?prime.
  ?number wdt:P155 ?prime.
}
Or just
SELECT ?number ?prime
WHERE
{
  ?number wdt:P5236 ?prime.
  ?number wdt:P155 ?prime.
}

Here, wd:Q49008 is the class prime number, wdt:P155 the property "follows" (as it seems that "previous" and "next" are instead "follows" "followed by").

3./!\ not working

```
SELECT ?odd
WHERE
{
  ?odd wdt:P156 ?even.
  ?even wdt:P5236 wd:Q200.
}
```

It should work, as "wdt:P156" means followed by, and "?even wdt:P5236 wd:Q200" means that ?even is even.
But I do not get the expected results with this...

4. This query gives all of the prime numbers available in the database

```
SELECT ?num
WHERE
{ ?number wdt:P1181 ?num.
 ?prime wdt:P1181 ?num_.
  ?number wdt:P5236 ?prime.
 FILTER(?num = ?num_)
}
```

5.

```
SELECT ?num
WHERE
{ ?number wdt:P1181 ?num.
 ?prime wdt:P1181 ?num_.
  ?number wdt:P5236 ?prime.
 FILTER(?num != ?num_)
}
```

6. /!\
This query does not give results.

```
SELECT ?num ?num_
WHERE
{ ?number wdt:P1181 ?num.
 ?prime wdt:P1181 ?pnum.
  ?number wdt:P5236 ?prime.
 FILTER(?num = ?pnum)
 ?number_ wdt:P1181 ?num_.
 ?prime_ wdt:P1181 ?pnum_.
  ?number_ wdt:P5236 ?prime_.
 FILTER(?num_ = ?pnum_)

 ?number wdt:P155 ?y.
 ?y wdt:P155 ?number_
}
LIMIT 100
```

Even though this query

```
SELECT ?num ?num_
WHERE
{ ?number wdt:P1181 ?num.
 ?prime wdt:P1181 ?pnum.
  ?number wdt:P5236 ?prime.
 FILTER(?num = ?pnum)

 ?number_ wdt:P1181 ?num_.
 ?prime_ wdt:P1181 ?pnum_.
  ?number_ wdt:P5236 ?prime_.
 FILTER(?num_ = ?pnum_)
 }
LIMIT 100
```

Gives all of the 100 first tuples of 2 prime numbers.
The issue seems to be with the   two following lines:

```
 ?number wdt:P155 ?y.
 ?y wdt:P155 ?number_
```

Where I use the follow property wdt:P155
I already had an issue at the question 3.
Indeed, even this type of query does not work:

```
 SELECT ?num  ?num_ ?t
WHERE {
?odd wdt:P1181 ?num.
?even wdt:P1181 ?num_.
?odd wdt:P31 wd:Q21199.
?even wdt:P31 wd:Q21199.
?odd ?t ?even.
FILTER(?num - ?num_ =-1)
}
limit 100
```

(If we just want to display the couples of numbers following each other)]