WEBSEM ASSIGNMENT-2

Shubhika Garg

1. Open the file human_2007_09_11.rdf in your favourite text editor.

What is the namespace (prefix name and expanded URI) used for the individuals (or instances) in this knowledge base?

Expanded URI: http://www.inria.fr/2007/09/11/humans.rdfs

Prefix: humans

2. What is the namespace (prefix name and expanded URI) used by the schema of this simple ontology?

Expanded URI: http://www.w3.org/2000/01/rdf-schema#

Prefix: rdfs

3. Write down all the information you know about John in the Turtle syntax.

I gave the Prefix *ent* for <http://www.inria.fr/2007/09/11/humans.rdfs-instances>

<ent#John> <humans#name> “John” ;

<humans#shirtsize> 12 ;

<humans#trouserssize> 44 ;

<humans#shoesize> 14 ;

<humans#age>37 ;

<humans#hasParent> <ent#Sophie> .

<ent#Mark> <humans#hasFather> <ent#John> .

<ent#Alice> <humans#hasFriend> <ent#John> .

<ent#Harry> <humans#hasChild> <ent#John> .

<ent#Jennifer> <humans#hasSpouse> <ent#John> .

4. The interface has a general pre-entered SPARQL query. Write instead the following query:

```
select ?x ?t where
{
?x rdf:type ?t
}
```

Graph | XML/RDF | Table | Validate

num	?x	?t
1	rdfs:subClassOf	rdf:Property
2	rdfs:label	rdf:Property
3	rdf:type	rdf:Property
4	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>	rdfs:Class
5	rdfs:comment	rdf:Property
6	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>	rdfs:Class
7	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>	rdfs:Class
8	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>	rdfs:Class
9	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	rdfs:Class
10	<http://www.inria.fr/2007/09/11/humans.rdfs#Lecturer>	rdfs:Class
11	<http://www.inria.fr/2007/09/11/humans.rdfs#Researcher>	rdfs:Class
12	<http://www.inria.fr/2007/09/11/humans.rdfs#Woman>	rdfs:Class
13	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>	rdf:Property
14	rdfs:domain	rdf:Property
15	rdfs:range	rdf:Property
16	<http://www.inria.fr/2007/09/11/humans.rdfs#hasParent>	rdf:Property
17	rdfs:subPropertyOf	rdf:Property
18	<http://www.inria.fr/2007/09/11/humans.rdfs#hasChild>	rdf:Property
19	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFather>	rdf:Property
20	<http://www.inria.fr/2007/09/11/humans.rdfs#hasMother>	rdf:Property
21	<http://www.inria.fr/2007/09/11/humans.rdfs#hasBrother>	rdf:Property
22	<http://www.inria.fr/2007/09/11/humans.rdfs#hasSister>	rdf:Property

5. Write down in one sentence what this query means?

This SPARQL query retrieves all the instances (?x) and their respective types (?t) from the dataset.

6. Run this query, how many answers do you get?

The query returned 69 rows.

Also, to count the number of rows returned, we can use the **COUNT** function:

```
select (count(?x) as ?number_of_results) where
{
?x rdf:type ?t
}
```

Graph | XML/RDF | Table | Validate

num	?number_of_results
1	69

7. What is/are the type(s) of John?

To retrieve the types of John, I used the **FILTER** Function:

```
select ?x ?t where
{
?x rdf:type ?t
FILTER (?x=<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>).
}
```

Graph | XML/RDF | Table | Validate

num	?x	?t
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>

The three types of John retrieved are: Animal, Male and Person.

8. Write down in one sentence what this query means?

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

SELECT *

WHERE

{

?x humans:hasSpouse ?y

}

The query uses a prefix declaration to define the namespace for the ontology, and then it selects all the resources (?x) that have a spouse (?y) using the predicate "hasSpouse" defined in that namespace.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT *
WHERE
{
?x humans:hasSpouse ?y
}
```

Graph | XML/RDF | Table | Validate

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

9. Run this query, how many answers do you get?

On running the above query, I got 6 answers. Also, on running **COUNT(*)**, it retrieved the count value as 6:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT (count(*) as ?count)
WHERE
{
?x humans:hasSpouse ?y
}
```

Graph | XML/RDF | Table | Validate

num	?count
1	6

10. Look at the knowledge base and write down which RDF property is used for indicating the shoe size of the people?

The RDF property is shoesize.

```
select ?x ?t where
{
?x rdf:type ?t
FILTER (?x=<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>).
}
```

Graph | XML/RDF | Table | Validate

num	?x	?t
1	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	rdf:Property

11. Write down the SPARQL query that provides for all the people their shoe size? How many answers do you get?

On running the query, 7 results were retrieved.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
select * where
{
?x humans:shoesize ?y
}
ORDER BY ?y
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	11
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	7
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	7
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	8
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	8

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
select (count(*) as ?count) where
{
?x humans:shoesize ?y
}
```

Graph | XML/RDF | Table | Validate

num	?count
1	7

12. Write down the SPARQL query that provides for all the people their shoe size if this information is available? How many answers do you get?

On running the below query, 17 answers were retrieved for me:

```
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
}
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>	
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>	
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>	
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	11
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14
11	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	7
12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	
13	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	7
14	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	8
15	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	8
16	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>	
17	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10

13. Write down the SPARQL query that provides the people who have a shoe size greater than 8? You can use the xsd:integer function to cast a variable into an integer (i.e. xsd:integer (?var))

The query returned 3 results.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT *
WHERE
{
?x rdf:type humans:Person
?x humans:shoesize ?y
FILTER(xsd:integer(?y)>8)
}
ORDER BY ?y
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	11
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14

14. Write down the SPARQL query that provides the people who have a shoe size greater than 8 or who have the shirt size greater than 12?

The query returned 3 rows.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT *
WHERE
{
?x humans:shoesize ?shoesize ;
humans:shirtsize ?shirtsize .
FILTER ( xsd:integer(?shoesize) > 8 || xsd:integer(?shirtsize)>12)
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x	?shoesize	?shirtsize
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	11	12
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14	12
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10	13

15. Look up the URI that identifies John and ask the engine what is the description of this person using the appropriate SPARQL keyword (see slide 50)?

The query retrieved 16 results.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
DESCRIBE ?x
WHERE
{
?x humans:name "John" .
}
```

Graph | XML/RDF | Table | Validate

num	?x	?_ast_v_0	?_ast_p_0	?_ast_v_1	?_ast_p_1
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	37	<http://www.inria.fr/2007/09/11/humans.rdfs#age>		
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>		
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasParent>		
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	John	<http://www.inria.fr/2007/09/11/humans.rdfs#name>		
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	12	<http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize>		
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>		
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	44	<http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize>		
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	rdf:type		
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>	rdf:type		
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>	rdf:type		
11	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inri
12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inri
13	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inri
14	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>	<http://www.inri
15	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inri
16	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>			<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	<http://www.inri

16. Write down the SPARQL query that provides the people that have at least one child? How many answers do you get? How many duplicates can you identify? Write down another SPARQL query that will remove the duplicates?

On running the query, 5 results were retrieved, with one duplicate pair: “Gaston”

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE
{
?x humans:hasChild ?children.
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

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

On running the query with **DISTINCT,** duplicates were removed, and 4 results were retrieved:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?x
WHERE
{
?x humans:hasChild ?children .
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

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

17. Write down the SPARQL query that provides all the men who do not have any children.

The query returned 3 results.

Query | Validate | to SPIN | to SPARQL | Prove | Trace | Search | Refresh stylesheet | Default stylesheet

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?x
WHERE
{
{?x rdf:type humans:Male} UNION {?x rdf:type humans:Man}
FILTER NOT EXISTS
{
?x humans:hasChild ?y
}
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>

18. Write down the query that provides all the people who have more than 100 years old?

The query returned 1 result:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT *
WHERE
{
?x rdf:type humans:Person
?x humans:age ?y
FILTER (xsd:integer(?y)>100)
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	102

19. Write down the SPARQL query that provides all the people pairs who have the same shirt size? It is ok to have rows like “x, y, size” and “y, x, size” as if x has the same shirt size than y, then y has the same shirt size than x.

The query returned 8 results:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?t ?size1
WHERE
{
?x humans:shirtsize ?size1 .
?t humans:shirtsize ?size2
FILTER (xsd:integer(?size1) = xsd:integer(?size2) && ?x!=?t)
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x	?t	?size1
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	12
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	12
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9

The below query retrieved all results showing both the values of the shirt sizes of two people and also being equal using **SELECT ***

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT *
WHERE
{
?x humans:shirtsize ?size1 .
?t humans:shirtsize ?size2
FILTER (xsd:integer(?size1) = xsd:integer(?size2) && ?x!=?t)
}
ORDER BY ?size1 ?size2
```

Graph | XML/RDF | Table | Validate

num	?x	?size1	?t	?size2
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	12
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	12
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	9
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	9

20. Write down the SPARQL query that provides all the people who are not men? How many answers do you get?

The query returned 11 results:

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE
{
  ?x rdf:type humans:Person .
   FILTER NOT EXISTS {?x rdf:type humans:Male}
   FILTER NOT EXISTS {?x rdf:type humans:Man}
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

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

21. Close and re-launch the corese application and load only the ontology human.rdfs. Write down a SPARQL query that provides all the classes defined in this ontology?

```
select DISTINCT ?x where
{
  ?x rdf:type ?y
FILTER (?y = rdfs:Class)
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
2	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#Lecturer>
4	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>
5	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>
6	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
7	<http://www.inria.fr/2007/09/11/humans.rdfs#Researcher>
8	<http://www.inria.fr/2007/09/11/humans.rdfs#Woman>

The above query retrieved 8 results.

22. Write down a SPARQL query that provides all subClassOf relationships defined in this ontology?

The query returned 9 results.

```
select * where
{
?x rdfs:subClassOf ?y
}
ORDER BY ?x
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
2	<http://www.inria.fr/2007/09/11/humans.rdfs#Lecturer>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
4	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>
5	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
6	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	<http://www.inria.fr/2007/09/11/humans.rdfs#Animal>
7	<http://www.inria.fr/2007/09/11/humans.rdfs#Researcher>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
8	<http://www.inria.fr/2007/09/11/humans.rdfs#Woman>	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>
9	<http://www.inria.fr/2007/09/11/humans.rdfs#Woman>	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>

23. Write down a SPARQL query that provides the definition and the translation of “shoe size”?

The query returned 2 results:

```
select * where
{
?x rdfs:comment ?y
FILTER (?x=<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>)
}
ORDER BY ?y
```

Graph | XML/RDF | Table | Validate

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	"express in some way the approximate length of the shoes for a person."@en
2	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	"taille, exprimée en points, des chaussures d'une personne."@fr

24. Write down a SPARQL query that provides all synonyms of the French term “personne”? You can make use of the lang(?var) function for this.

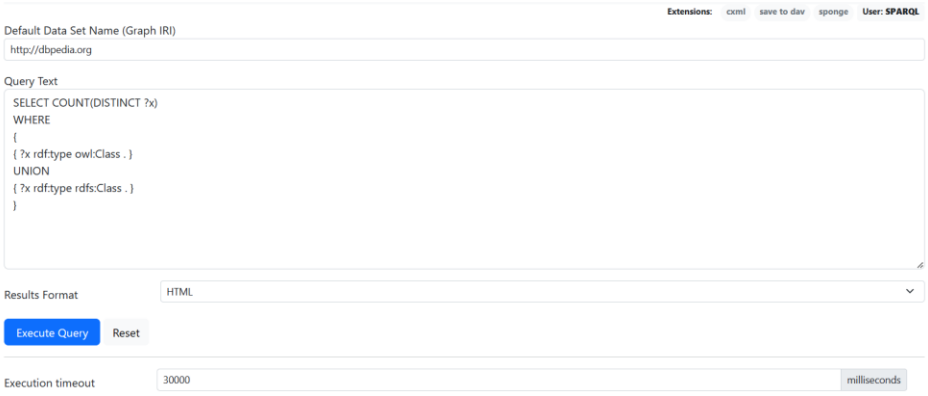
The query returned 4 results for the French synonyms:

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

Graph | XML/RDF | Table | Validate

num	?x	?personne
1	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	"homme"@fr
2	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	"humain"@fr
3	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	"personne"@fr
4	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>	"être humain"@fr

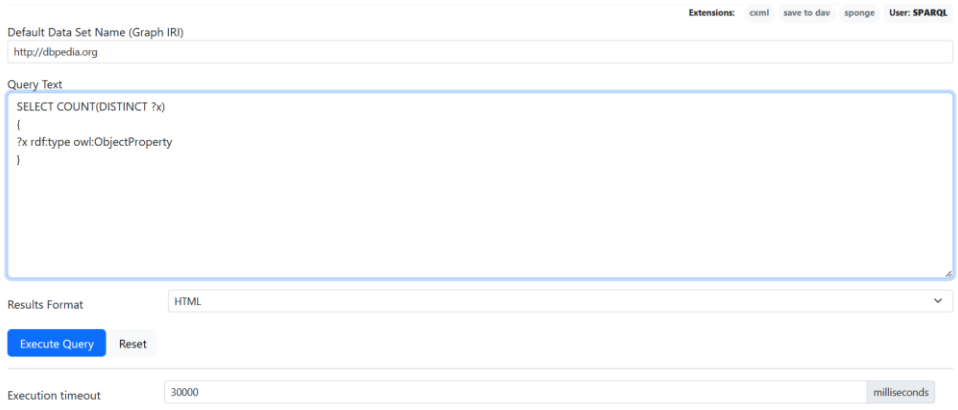
25. Use now the DBpedia SPARQL endpoint at "http://dbpedia.org/sparql". Write down 3 SPARQL queries that respectively counts the number of classes, object properties and datatype properties contained in the DBpedia ontology. Do not try to write a single query since it is likely to time out.



Output:

SPARQL | HTML5 table

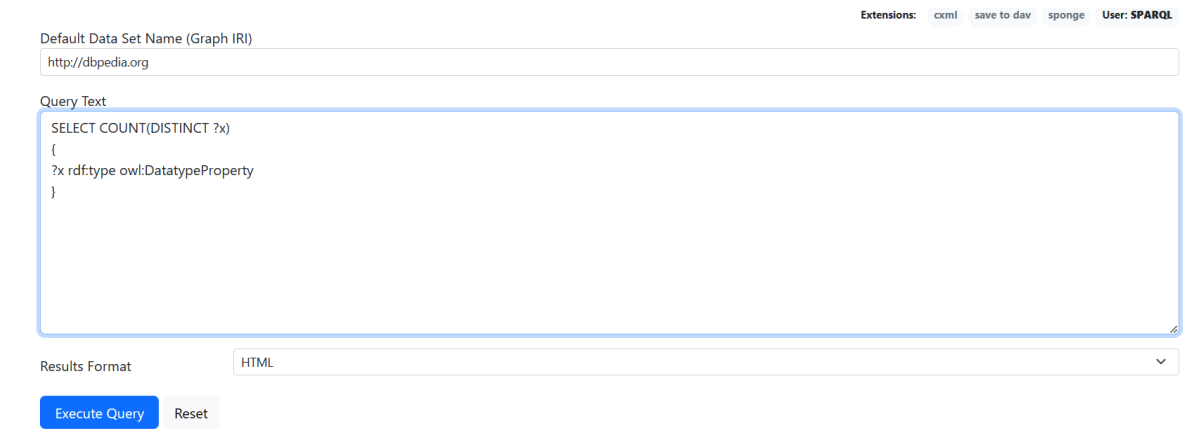
callret-0
1568



Output:

SPARQL | HTML5 table

callret-0
1192



Output:

SPARQL | HTML5 table

callret-0
2035

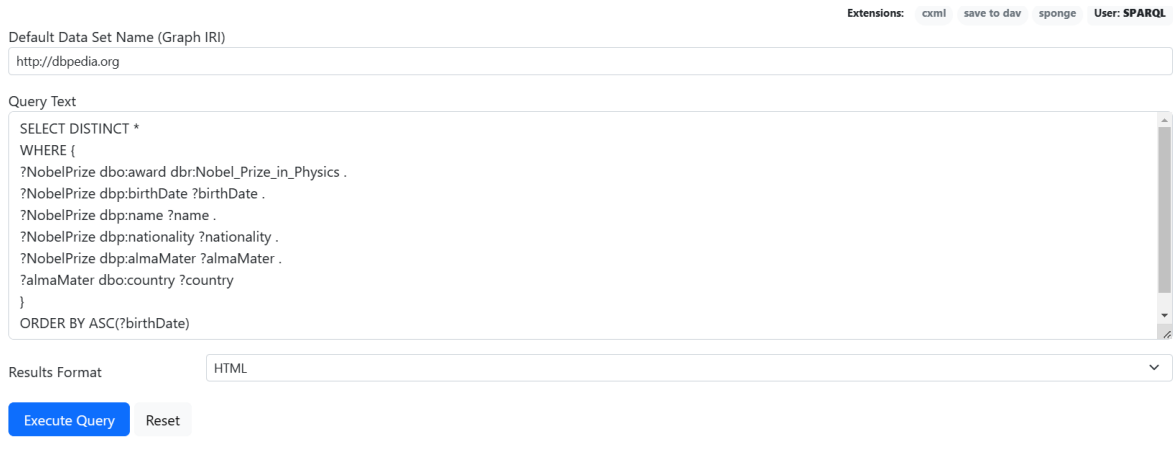
26. Explain the differences between the /resource, /data and /page URIs for a given resource.

/resource URI: It represents the resource itself and gives information about the resource, such as its properties, classes, and relationships with other resources.

/data URI: It is used to retrieve the data associated with a particular resource. The data includes the values of the resource's properties and links to other resources.

/page URI: It is used to retrieve a web page that provides more general information about the resource, such as its history or some notable achievements.

27. Write down a SPARQL query that lists all winners of the Nobel Prize in Physics sorted from oldest to youngest.



Output File:

sparql_2023-03-15_22-11-43Z.csv

28. Write down a SPARQL query that lists the top 10 Universities with most winners of the Nobel Prize in Physics. Hint: you may want to use the property dbo:almaMater

SPARQL Query Editor About Tables Conductor Facet Browser Permalink

http://dbpedia.org

Query Text

```
SELECT ?university (COUNT(?person) AS ?winners)
WHERE {
  ?award dbo:award dbr:Nobel_Prize_in_Physics .
  ?person dbo:award ?award .
  ?person dbo:almaMater ?university .
}
GROUP BY ?university
ORDER BY DESC(?winners)
LIMIT 10
```

Results Format HTML

Output:

SPARQL | HTML5 table

university	winners
http://dbpedia.org/resource/Massachusetts_Institute_of_Technology	4
http://dbpedia.org/resource/University_of_California,_Berkeley	2
http://dbpedia.org/resource/New_York_University	2
http://dbpedia.org/resource/California_Institute_of_Technology	2
http://dbpedia.org/resource/University_of_Cambridge	1
http://dbpedia.org/resource/University_at_Buffalo	1
http://dbpedia.org/resource/Technion_–_Israel_Institute_of_Technology	1
http://dbpedia.org/resource/Warsaw_University_of_Technology	1
http://dbpedia.org/resource/University_of_Sarajevo	1
http://dbpedia.org/resource/University_of_Warsaw	1

29. Write down a SPARQL query that provides the number of winners of the Nobel Prize in Physics who are immigrants (i.e., born in a country different from that of where is located the employer University)

Extensions: cxml save to dav sponge User: SPARQL

Default Data Set Name (Graph IRI)

http://dbpedia.org

Query Text

```
SELECT (COUNT(DISTINCT ?person) AS ?immigrantWinners)
WHERE {
  ?award dbo:award dbr:Nobel_Prize_in_Physics .
  ?person dbo:award ?award ;
          dbo:birthPlace ?nationality ;
          dbo:almaMater ?university .
  ?university dbo:country ?employerLocation .
  FILTER (?nationality != ?employerLocation)
}
```

Results Format HTML

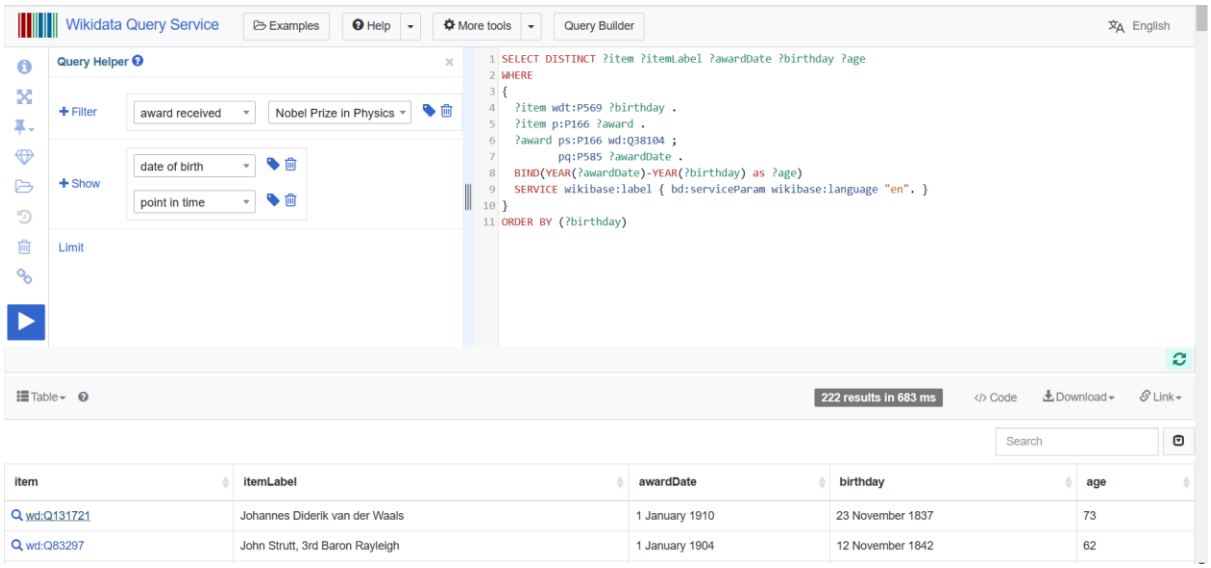
Execute Query Reset

Output:

SPARQL | HTML5 table

immigrantWinners
7

30. Let's now query Wikidata using the SPARQL endpoint at http://query.wikidata.org/. Write down a SPARQL query that lists all winners of the Nobel Prize in Physics sorted from oldest to youngest.

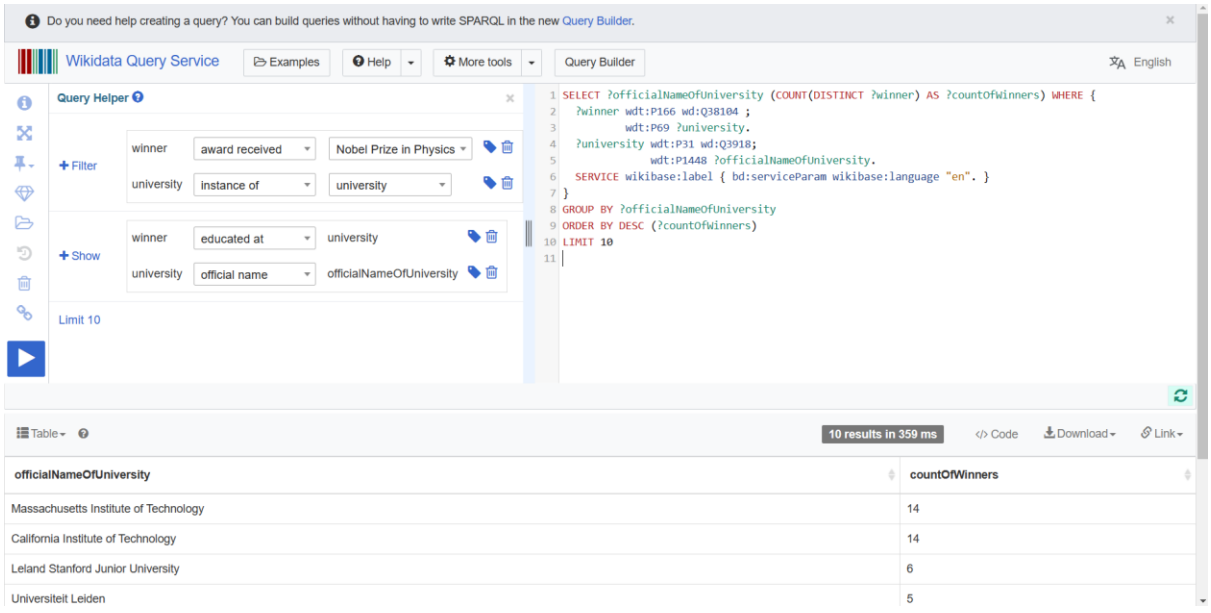


item	itemLabel	awardDate	birthday	age
wd:Q131721	Johannes Diderik van der Waals	1 January 1910	23 November 1837	73
wd:Q83297	John Strutt, 3rd Baron Rayleigh	1 January 1904	12 November 1842	62

Output:

query.csv

31. Write down a SPARQL query that lists the top 10 Universities with most winners of the Nobel Prize in Physics



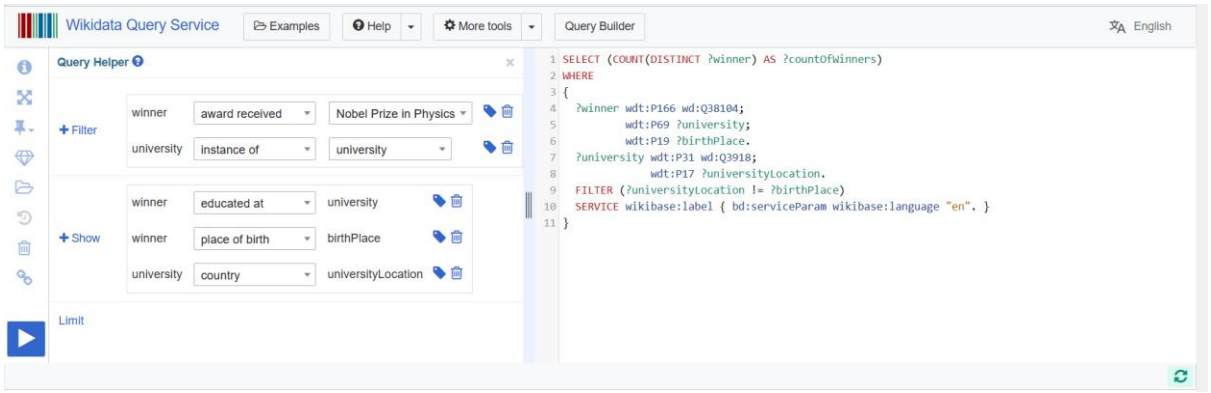
officialNameOfUniversity	countOfWinners
Massachusetts Institute of Technology	14
California Institute of Technology	14
Leland Stanford Junior University	6
Universiteit Leiden	5

Output:

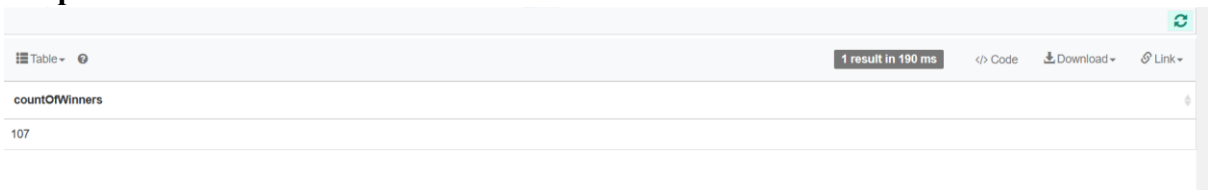
Table | 10 results in 359 ms | Code | Download | Link

officialNameOfUniversity	countOfWinners
Massachusetts Institute of Technology	14
California Institute of Technology	14
Leland Stanford Junior University	6
Universiteit Leiden	5
Universität Zürich	5
Universiteit Utrecht	4
Московский государственный университет имени М. В. Ломоносова	3
Uniwersytet Wrocławski	3
Owens College	3
Victoria University of Manchester	3

32. Write down a SPARQL query that provides the number of winners of the Nobel Prize in Physics who are immigrants (i.e., born in a country different from that of the University).



Output:



31. Using the Linked Open Numbers dataset (http://km.aifb.kit.edu/projects/numbers/) find all even numbers.

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?evenNumbers
{
?evenNumbers num:NaturalNumber ;
                    num:primefactor ?primeNumbers .
FILTER EXISTS (?primeNumbers = 2)
}
```

32. Find all numbers that are successors of one of their prime factors.

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?successors
{
?successors num:NaturalNumber .
?successors num:primefactor ?primeNumbers .
?primeNumbers num:next ?successors .
}
```

33. Find all odd numbers.

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?oddnumbers
{
?oddnumbers num:NaturalNumber .
?oddnumbers num:primefactor ?primenumbers .
FILTER NOT EXISTS (?primenumbers = 2)
}
```

34. Find all prime numbers.

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?primeNumber
WHERE {
  ?primeNumber num:NaturalNumber ;
                    num:primefactor ?primes .
  FILTER NOT EXISTS
  {
    ?primeNumber num:primefactor ?nonPrimes .
    FILTER (?nonPrimes != ?primes && ?primeNumber = ?primes * ?nonPrimes)
  }
```

35. Find all non-prime numbers.

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?nonPrimeNumber
WHERE {
  ?nonPrimeNumber num:NaturalNumber ;
                    num:primefactor ?primes .
  FILTER EXISTS {
    ?nonPrimeNumber num:primefactor ?nonPrimes .
    FILTER (?nonPrimes != ?primes && ?nonPrimeNumber = ?primes * ?nonPrimes)
  } FILTER (?nonPrimeNumber > 1)
}
```

36. Find all twin primes (i.e., two prime numbers at a distance of 2, e.g., 17 and 19)

```
PREFIX num: <http://km.aifb.kit.edu/projects/numbers#>
SELECT DISTINCT ?num1 ?num2
WHERE
{
  ?num1 num:NaturalNumber ;
          num:primefactor ?primes1 .
  ?num2 num:NaturalNumber ;
          num:primefactor ?primes2 .
  FILTER (?primes1 = ?primes2 && ?num2 = ?num1 + 2)
}
```