Lab 2

Víctor MONTESDEOCA FENOY

Efrén BOYARIZO GARGALLO

Table of Contents

Part I: The RDF Knowledge Base

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

> The prefix name is "humans"
>
> The expanded uri is "http://www.inria.fr/2007/09/11/humans.rdfs"

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

> The prefix name is "xsd"
>
> The expanded uri is "http://www.w3.org/2001/XMLSchema#"

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

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

<http://www.w3.org/TR/rdf-syntax-grammar>
humans:John
  humans:name "John";
  humans:age: "37";
  humans:shoesize "14";
  humans:shirtsize "12";
  humans:trouserssize "44";
  humans:hasFriend humans:Alice;
  humans:hasParent humans:Harry;
  humans:hasParent humans:Sophie;
  humans:hasSpouse humans:Jennifer;
  humans:hasChild humans:Mark .
```

Part II: SPARQL 101

2. Write down in one sentence what this query means? Run this query, how many answers do you get? What is/are the type(s) of John?

> We are selecting all elements in the RDF (referenced by x) and we are also selecting the type for each one (referenced by t)
>
> We get 69 answers
>
> John is referenced three times, and as such has three different types:
> http://www.inria.fr/2007/09/11/humans.rdfs#Person http://www.inria.fr/2007/09/11/humans.rdfs#Animal
> http://www.inria.fr/2007/09/11/humans.rdfs#Male
>
> Note that a Male is a Person; and a Person in an Animal

3. Write down in one sentence what this query means? Run this query, how many answers do you get?

> Selects all attributes from instances that have the relationship humans:hasSpouse
>
> We get 6 rows

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

> The property is "shoesize"

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

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

> We get 7 results:

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

6. 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?

```
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 }
}
```

> We get 17 answers:

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

7. 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)).

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

num	?x	?y
1	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	14

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

8. 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?

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

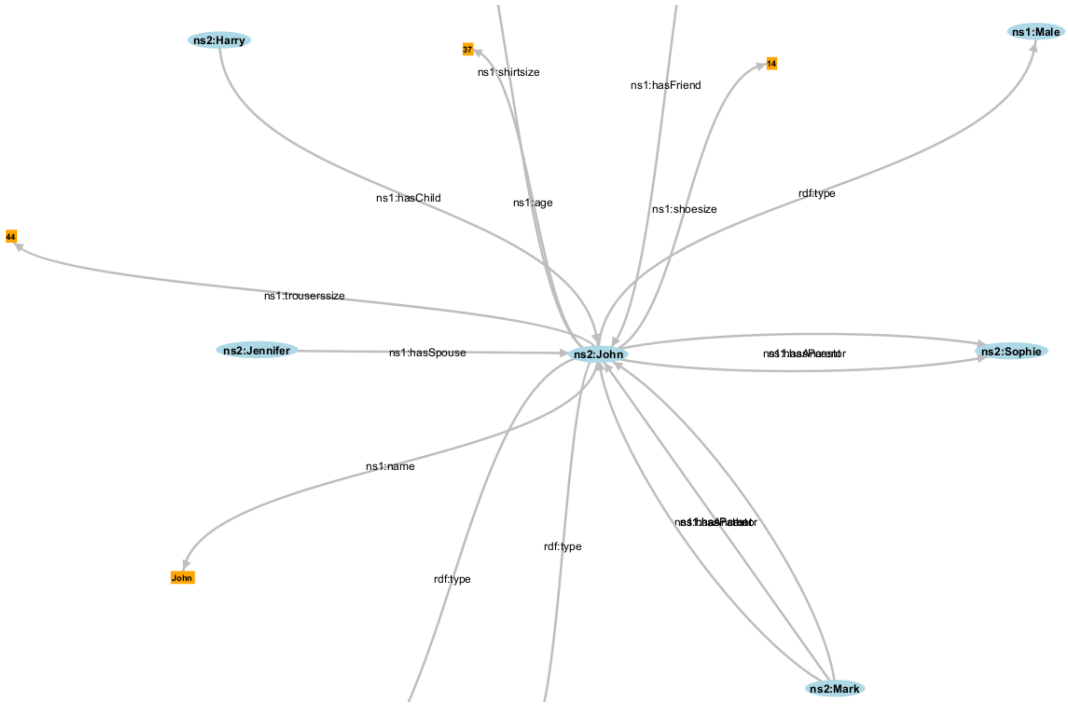
num	?x	?y	?z
1	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	14	12
2	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	11	12
3	http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	10	13

9. 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 URI that identifies John is http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
>
> We can ask the engine with the following query:

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

> And the resulting graph:



10. 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?

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

> We get 5 answers, where Gaston appears twice as he has two children: Jack and Pierre

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#John
2	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
3	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
4	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
5	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre

> To remove the duplicates we can use SELECT DISTINCT instead of SELECT (but only for x as we want parents to appear only once):

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

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

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

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?x
WHERE {
    ?x a humans:Man
    OPTIONAL { ?x humans:hasChild ?y}
    FILTER (!bound(?y))
}
```

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

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

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

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

13. 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 short size than x.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y ?z
WHERE {
    ?x humans:shirtsize ?z
    ?y humans:shirtsize ?z
    FILTER (?x != ?y)
}
```

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

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

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

> William, John, Mark, David and Karl are not defined as Man, only as Person. As such they are not excluded, giving us a total of 12 people who are not men:

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#Sophie
3	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark
4	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve
5	http://www.inria.fr/2007/09/11/humans.rdfs-instances#David

num	?x
6	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice
7	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora
8	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura
9	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer
10	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine
11	http://www.inria.fr/2007/09/11/humans.rdfs-instances#William
12	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl

Part III: The RDFS Ontology

2. Write down a SPARQL query that provides all the classes defined in this ontology?

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?class
WHERE {
    ?class a rdfs:Class
}
```

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

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

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?subClass ?superClass
WHERE {
    ?subClass rdfs:subClassOf ?superClass .
}
```

num	?subClass	?superClass
1	http://www.inria.fr/2007/09/11/humans.rdfs#Male	http://www.inria.fr/2007/09/11/humans.rdfs#Animal
2	http://www.inria.fr/2007/09/11/humans.rdfs#Female	http://www.inria.fr/2007/09/11/humans.rdfs#Animal
3	http://www.inria.fr/2007/09/11/humans.rdfs#Man	http://www.inria.fr/2007/09/11/humans.rdfs#Male

num	?subClass	?superClass
4	http://www.inria.fr/2007/09/11/humans.rdfs#Man	http://www.inria.fr/2007/09/11/humans.rdfs#Person
5	http://www.inria.fr/2007/09/11/humans.rdfs#Person	http://www.inria.fr/2007/09/11/humans.rdfs#Animal
6	http://www.inria.fr/2007/09/11/humans.rdfs#Lecturer	http://www.inria.fr/2007/09/11/humans.rdfs#Person
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

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

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?itemLabel ?item ?itemDescription
WHERE {
    ?item rdfs:label ?itemLabel
    ?item rdfs:comment ?itemDescription
    FILTER contains(?itemLabel, "shoe size")
}
```

num	?itemLabel	?item	?itemDescription
1	"shoe size"@en	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	"shoe size"@en	http://www.inria.fr/2007/09/11/humans.rdfs#shoesize	"taille, exprimée en points, des chaussures d'une personne."@fr

5. 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.

```
PREFIX humans: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?item  ?itemLabel
WHERE {

?item rdfs:label ?itemLabel

FILTER (lang(?itemLabel) = 'fr')
FILTER (?item = humans:Person)

}
```

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

Part IV: Querying the Open Web

1. 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.

```
SELECT (COUNT(DISTINCT ?class) AS ?classCount)
WHERE {
    ?class a owl:Class .
}
```

classCount
1568

```
SELECT (COUNT(DISTINCT ?property) AS ?objectPropertyCount)
WHERE {
    ?property a rdf:Property ;
        rdf:type owl:ObjectProperty .
}
```

objectPropertyCount
1194

```
SELECT (COUNT(DISTINCT ?property) AS ?datatypePropertyCount)
WHERE {
    ?property a rdf:Property ;
        rdf:type owl:DatatypeProperty .
}
```

datatypePropertyCount
1777

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

> /resource: represent the metadata in a RESTful or linked data context. This includes the resources, its properties, relationships, and other metadata
>
> /data: offers access the raw or structured data associated with a resource
>
> /page: is used to access human-readable web page with information about the resource

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

```
SELECT ?person ?name ?birthdate
WHERE {
    ?person rdf:type dbo:Person ;
        dbo:birthDate ?birthdate ;
        foaf:name ?name ;
        dbo:award dbr:Nobel_Prize_in_Physics .
}
ORDER BY ?birthdate
```

4. 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

```
SELECT ?university (COUNT(?person) AS ?count)
WHERE {
    ?person rdf:type dbo:Person ;
        dbo:almaMater ?university;
        foaf:name ?name ;
        dbo:award dbr:Nobel_Prize_in_Physics .
}
GROUP BY ?university
ORDER BY DESC(?count)
LIMIT 10
```

5. 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)

```
SELECT COUNT(*)
WHERE {
    ?person rdf:type dbo:Person ;
        dbo:almaMater ?university;
        dbo:award dbr:Nobel_Prize_in_Physics ;
        dbo:birthPlace ?birthPlace .
    ?birthPlace dbo:country ?birthCountry .
    ?university dbo:country ?uniCountry .

    FILTER (?birthCountry != ?uniCountry)
}
```

Part V: Querying Wikidata

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

```
SELECT ?itemLabel ?birthdate
WHERE {
  ?item p:P166 ?awardStat .
  ?awardStat ps:P166 wd:Q38104 .
  ?item wdt:P569 ?birthdate

SERVICE wikibase:label { bd:serviceParam wikibase:language "en" . }
}
ORDER BY ?birthdate
```

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

```
SELECT ?universityLabel (COUNT(?item) AS ?count)
WHERE {
  ?item p:P166 ?awardStat .
  ?awardStat ps:P166 wd:Q38104 .
  ?item wdt:P69 ?university .

SERVICE wikibase:label { bd:serviceParam wikibase:language "en" . }
}
GROUP BY ?universityLabel
ORDER BY DESC(?count)
LIMIT 10
```

3. 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)

```
SELECT (COUNT(*) AS ?count)
WHERE {
  ?item p:P166 ?awardStat ; wdt:P69 ?university ; wdt:P19 ?birthPlace .
  ?birthPlace wdt:P17 ?birthCountry .
  ?awardStat ps:P166 wd:Q38104 .
  ?university wdt:P17 ?uniCountry .

  FILTER (?birthCountry != ?uniCountry)
SERVICE wikibase:label { bd:serviceParam wikibase:language "en" . }
}
```

Part VI: SPARQL Puzzles

1. Find all even numbers.

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?evenNumber
WHERE {
    ?evenNumber number:primeFactor ?two
    FILTER (?two = number:Two)
}
```

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

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?succ
WHERE {
    ?succ number:primeFactor ?factor
    ?succ number:previous ?factor
}
```

3. Find all odd numbers.

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?number
WHERE {
    ?number number:primeFactor ?factor .
    FILTER (!(?factor = number:Two))
}
```

4. Find all prime numbers.

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?primeNumber
WHERE {
    ?primeNumber number:primefactor ?factor
    FILTER NOT EXISTS {
        ?otherFactor number:lessThan ?primeNumber
        ?primeNumber number:primefactor ?otherFactor
        FILTER (?otherFactor != ?factor)
    }
}
```

5. Find all non-prime numbers.

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?primeNumber
WHERE {
    ?primeNumber number:primefactor ?factor
    FILTER EXISTS {
        ?otherFactor number:lessThan ?primeNumber
        ?primeNumber number:primefactor ?otherFactor
        FILTER (?otherFactor != ?factor)
    }
}
```

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

```
PREFIX number: <http://km.aifb.kit.edu/projects/numbers/#>
SELECT ?twinPrime1 ?twinPrime2
WHERE {
    ?twinPrime1 number:primefactor ?factor1
    ?twinPrime2 number:primefactor ?factor2

    FILTER NOT EXISTS {
        ?otherFactor1 number:lessThan ?twinPrime1
        ?twinPrime1 number:primefactor ?otherFactor1
        FILTER (?otherFactor1 != ?factor1)
    }

    FILTER NOT EXISTS {
        ?otherFactor2 number:lessThan ?twinPrime2 .
        ?twinPrime2 number:primefactor ?otherFactor2 .
        FILTER (?otherFactor2 != ?factor2)
    }

    FILTER (?twinPrime2 - ?twinPrime1 = 2)
}
```