SPARQL

Statement:

Solve the following queries with SPARQL. You are suggested to validate your answers with the SPARQL Explorer (http://dbpedia.org/snorql/). If you create any PREFIX include it in your answer. You can also check the DBPedia ontology schema at: http://mappings.dbpedia.org/server/ontology/classes/.

1. *Write the SPARQL query to list all the DBPedia classes (i.e., instances of the owl:Class).*

```
Select ?class {
   ?class a owl:Class .
}
```

2. *Write the SPARQL query to list all the DBPedia properties (i.e., either instances of the owl:DatatypeProperty or owl:ObjectProperty properties).*

```
Select ?property WHERE {
   {?property a owl:DatatypeProperty}
     UNION
   {?property a owl:ObjectProperty}
}
```

3. *Write the SPARQL query to list all the triples in DBPedia using the populationEstimate (i.e., http://dbpedia.org/property/populationEstimate) property:*

```
SELECT ?d ?r
WHERE {
  ?d <http://dbpedia.org/property/populationEstimate> ?r
}
```

4. *Write the SPARQL query (i.e., a single query) to list all the pairs <SUBJECT, PROPERTY> where <http://dbpedia.org/ontology/CyclingTeam> is the OBJECT **and** all the pairs <PROPERTY, OBJECT> where <http://dbpedia.org/ontology/CyclingTeam> is the SUBJECT.*

```
SELECT ?property ?range ?domain
WHERE {
```

```
  { <http://dbpedia.org/ontology/CyclingTeam> ?property ?range }
  UNION
  { ?domain ?property <http://dbpedia.org/ontology/CyclingTeam> }
}
```

5. *Now check the results you obtained from the previous query. Briefly explain the SPARQL query results. What do these pairs mean (you can group the results per similarity and comment on each group)? You can ignore the following properties (for internal usage of DBPedia): wasDerivedFrom and isDefinedBy as well as describedby, defines and describes (mainly used for metadata purposes).*

PROPERTY	RANGE	DOMAIN
Type tell us CyclingTeam is a class, and subClassOf that it's subclass of SportsTeam		
rdf:type	owl:Class	-
rdfs:subClassOf	dbpedia:ontology/SportsTeam	-
The rdfs:label provides a human-readable interpretation. The @ determines the language tag. Thus, it is provided in four languages: German, English, Italian and Greek		
rdfs:label	"Radsportteam"@de	-
rdfs:label	"cycling team"@en	-
rdfs:label	"squadra di ciclismo"@it	-
rdfs:label	"ομάδα ποδηλασίας"@el	-
<http://www.w3.org/ns/prov#wasDerivedFrom>	<http://mappings.dbpedia.org/index.php/OntologyClass:CyclingTeam>	-
rdfs:isDefinedBy	dbpedia:ontology/	-
<http://www.w3.org/2007/05/powder-s#describedby>	:classes#	-
The next ones (rdf:type where CyclingTeam is at the domain) refer to instances of this class (below, just one example, but there are plenty).		
rdf:type	-	:Miche-Guerciotti

Since CyclingTeam is at the range of the rdfs:domain property it means that the uciCode property must have as domain a CyclingTeam

rdfs:domain	-	dbpedia:ontology/uciCode
<http://open.vocab.org/terms/defines>	-	dbpedia:ontology/
<http://open.vocab.org/terms/describes>	-	:classes#

6. *Now, let us explore the data. Write the SPARQL query to list the name (i.e., rdfs:label) of all persons (i.e., of type <http://dbpedia.org/ontology/Person>) that were born (i.e., <http://dbpedia.org/property/birthPlace>) in Barcelona. Assume that the range of the birthPlace property is a literal. Also, order the result in ascending order.*

```
SELECT ?name
WHERE {
       ?p  a <http://dbpedia.org/ontology/Person> ;
           rdfs:label ?name   ;
           dbpedia2:birthPlace ?c .
         FILTER (regex(?c, "^barcelona"@en, "i") &&
langMatches(lang(?name), "EN")) .
} ORDER BY ASC (?p)
```