In this PSO, you will learn how to issue SPARQL queries over an endpoint that serves the DBpedia RDF dataset. An endpoint is essentially a server that exposes a dataset through a public interface/website that can be queried using the SPARQL query language. The endpoint we will be using is for the DBpedia dataset, a rich RDF dataset that represents Wikipedia as RDF entries.

The following query is used as a guide to help you derive other SPARQL queries. The query consists of a set of triple patterns that correspond to information we are interested about regarding the famous figure, Muhammad Ali.

A triple pattern consists of three main parts, a subject, predicate, and an object. Each part can be either bound (has a value such as :Muhammad_Ali) or unbound (a variable such as ?name). As a first exercise, write the following query in the editor found at the following address to get a feel of SPARQL queries and how the result looks like:

Endpoint Address: http://dbpedia.org/snorql/

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX : <http://dbpedia.org/resource/>

SELECT ?name ?education ?deathplace ?birthplace ?birthyear ?deathyear
WHERE {
  :Muhammad_Ali rdfs:label  ?name .
  :Muhammad_Ali dbo:education ?education .
  :Muhammad_Ali dbo:birthYear ?birthyear .
  :Muhammad_Ali dbo:birthPlace ?birthplace .
  :Muhammad_Ali dbo:deathYear ?deathyear .
  :Muhammad_Ali dbo:deathPlace ?deathplace .
FILTER(lang(?name) = 'en')
}
```

LISTING 1. Sample SPARQL query

The query retrieves the name, education, birth year, birth place, death year, and death place of the resource :Muhammad_Ali. The FILTER operation is also used to present only strings with the English tag (i.e. with the @en tag).

1. TIPS

(1) Visit **http://dbpedia.org/page/Muhammad_Ali** to get a feel of what a DBpedia entry looks like
(2) Don't forget to end every triple pattern in your SPARQL query with a dot '.' to avoid syntax errors
(3) Use prefixes (e.g. dbo:) to shorten URLS (dbo:eduction instead of <http://dbpedia.org/ontology/edu
(4) If needed, you can show all unbound variables (e.g, ?name) using asterisk symbol '*' after the SELECT keyword

2. Queries

2.1. **Query 1.** Find the English name (i.e., en), total number of decision wins (decWins), decision losses (decLosses), knock-out wins (koWins), and knock-out losses (koLosses) for the resource **:Muhammad_Ali**. Make sure the variable names are representative of the result, e.g., ?name for a name.

2.1.1. *Resources.*

(1) <http://dbpedia.org/property/decWins>
(2) <http://dbpedia.org/property/decLosses>
(3) <http://dbpedia.org/property/koWins>
(4) <http://dbpedia.org/property/koLosses>

2.2. **Query 2.** Find the English name (i.e., en), and education of all resources of type Boxers.

2.2.1. *Resources.*

(1) <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> (**Tip**: rdf:type)
(2) <http://dbpedia.org/ontology/Boxer>

2.3. **Query 3.** Find the English name of boxers and the number of knock-out wins (koWins) where the knock-out wins are more than 60.
(Tip: FILTER(?var > 60))

2.3.1. *Resources.*

(1) <http://dbpedia.org/property/koWins>

2.4. **Query 4.** Find the sorted names of all boxers that have the WBA Super Featherweight champion title (WBA_Super_Featherweight_Champion).
(TIP: ORDER BY(?var))

2.4.1. *Resources.*

(1) <http://dbpedia.org/ontology/Boxer>
(2) <http://dbpedia.org/resource/WBA_Super_Featherweight_Champion>

2.5. **Query 5.** Come up with a new query about boxing. Use the DBpedia entry of Muhammad Ali as a guide to determine the type of predicates that can be used to infer information about boxers.