# Lab: Query Linked Data

**Reading Materials:**

* Lecture slides
* SPARQL 1.1 Query Language: <https://www.w3.org/TR/sparql11-query/>

**Submissions:**

You should hand in a document containing the queries for the exercises 2, and the program for the task 3.

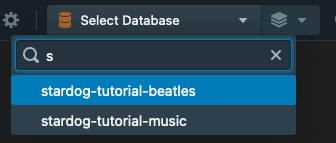
## 1. Get Familiar with SPARQL

In this part you will get familiar with SPARQL using [Stardog Express.](https://www.stardog.com/stardogexpress/) Stardog Express is a Cloud Starter Kit, a free, hosted environment that includes Stardog Studio, pre-built datasets, and sample SPARQL queries, etc.

* Go to Stardog Studio [(https://stardog.studio/#/.](https://stardog.studio/#/))
* Click the button on the left bottom corner  , choose the Stardog Express server and connect to the server.



* Select the database to query.



* Follow the tutorial <https://docs.stardog.com/tutorials/learn-sparql>and practice the SPARQL queries in the tutorial.

## 2. Query Open Linked Data

In this task you will query the DBpedia via its SPARQL endpoint. Go to the [DBpedia SPARQL endpoint](https://dbpedia.org/sparql) and solve the exercises below. In the menu ”Tables” you can find "[Namespace Prefixes"](https://dbpedia.org/sparql/?help=nsdecl) which provides the list of namespace prefixes encoded in the endpoint.

2.1) Find 20 example concepts in the DBpedia dataset.

select distinct ?Concept where {[] a ?Concept} LIMIT 20

2.2) Find the resource of Jönköping. Tip: don’t forget to use the language tag in your query: “Jönköping”@en

SELECT DISTINCT ?city

WHERE {

?city a dbo:City;

dbp:officialName "Jönköping"@en.

} LIMIT 1

2.3) Describe Jönköping using the URI retrieved by the previous query (use a DESCRIBE query).

DESCRIBE <https://dbpedia.org/resource/Jönköping/>

2.4) The behavior of a DESCRIBE query is up to the server. Create a query to retrieve all properties around Jönköping, i.e. all predicates and objects where Jönköping is the subject, and all predicated and subjects where Jönköping is the object.

SELECT DISTINCT ?s, ?p, ?o

WHERE {{ <http://dbpedia.org/resource/Jönköping> ?p ?o.}

UNION

{?s ?p <http://dbpedia.org/resource/Jönköping> .}}

2.5) List the people and their names who are born in Jönköping before 1900.

SELECT ?person, ?name, ?date

WHERE {

?person dbo:birthPlace dbr:Jönköping;

dbp:name ?name;

dbp:birthDate ?date.

FILTER (?date < "1900-01-01"^^xsd:date).

}

2.6) Are there musical artists who were born in Jönköping? (use an ASK query).

true

ASK {

?person rdf:type dbo:MusicalArtist;

dbo:birthPlace dbr:Jönköping.

}

2.7) Find 10 people (URI and place of death) who were born in Jönköping, but died elsewhere.

SELECT ?person, ?name, ?deathPlace

WHERE {

?person dbo:birthPlace dbr:Jönköping;

dbp:name ?name;

dbo:deathPlace ?deathPlace.

FILTER (?deathPlace != dbr:Jönköping).

} LIMIT 10

2.8) Now make some (at least two) fun queries over DBpedia.

***Note***: There are a few different ways to explore the DBpedia ontology at [https://www.dbpedia.org/resources/ontology/.](https://www.dbpedia.org/resources/ontology/)

***Motto and mascot for private universities***

SELECT DISTINCT ?university, ?mascot, ?motto

WHERE {

?university a dbo:University;

dbp:type dbr:Private\_university;

dbo:mascot ?mascot;

dbo:motto ?motto.

} LIMIT 100

**Swedish writers**

SELECT DISTINCT ?writer, ?name, ?nationality

WHERE {

?writer a dbo:Writer;

rdfs:label ?name;

dbp:nationality ?nationality

FILTER(LANG(?name)="en").

FILTER(?nationality = "Swedish"@en).

} LIMIT 100

## 3. Query Linked Data from program

[SPARQLWrapper](https://sparqlwrapper.readthedocs.io/en/latest/main.html) is a simple Python wrapper around a SPARQL service to remotely execute queries. Read its documentation and understand how to use it. The SPARQL query result is serialized in a JSON format. The format details can be found at <https://www.w3.org/TR/sparql11-results-json/>

Make some (at least two) queries over DBpedia from your program. If you wish to visualize the query result (not required), <https://github.com/google/google-visualization-python> might be a good tool. It is a nice python library built on Google Visualization API.

The same goes for this program as the issue with queries is prevalent.