PhUSE CSS 2019

“Let’s Make a Knowledge Graph!”

An Interactive Hands-on Workshop

EXERCISES

Version 4.0

2019-06-09

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Contents

[Introduction 2](#_Toc10547740)

[Server Login and Preparation 2](#_Toc10547741)

[Exercises 3](#_Toc10547742)

[1. Create Your Study Graph 3](#_Toc10547743)

[1.1 Open the Graph Editor 3](#_Toc10547744)

[1.2 Add a Study Information 4](#_Toc10547745)

[1.3 Add Persons and Treatments 5](#_Toc10547746)

[1.4 Add Demographics 6](#_Toc10547747)

[1.5 Add another Person 7](#_Toc10547748)

[1.6 Review your graph 8](#_Toc10547749)

[1.7 Export to TTL 8](#_Toc10547750)

[1.8 Validate the Data 9](#_Toc10547751)

[1.9 Upload to Database 11](#_Toc10547752)

[2. Query Your Graph 12](#_Toc10547753)

[2.1 Show all triples 12](#_Toc10547754)

[2.2 How many people *randomizedTo* treatment arms inthe study? 12](#_Toc10547755)

[2.3 Find the names of people who are randomized to treatment arms. 12](#_Toc10547756)

[2.4 List the names of the people randomized to treatment arm type and the drug they received. 13](#_Toc10547757)

[3. Extend to Other Graphs (Federated Query) 13](#_Toc10547758)

[3.1 Your study in ClinicalTrials.gov 13](#_Toc10547759)

[3.2 Study Title and Phase 14](#_Toc10547760)

[3.3 Information about the Study Drug on DBpedia 14](#_Toc10547761)

[3.4 Drug Abstract 14](#_Toc10547762)

[4. Ontology and Inference 14](#_Toc10547763)

[4.1 Review the Ontology 14](#_Toc10547764)

[4.2 Add the Ontology to the LDWStudy database 15](#_Toc10547765)

[4.3 Find the names of HumanStudySubjects in your study. 15](#_Toc10547766)

[4.4 Write a reasoner-based query to find all types (classes) of Persons 16](#_Toc10547767)

[5. Merge Studies 17](#_Toc10547768)

[5.1 Create the Data Pool 17](#_Toc10547769)

[5.2 How many HumanStudySubjects participated in all Studies? 18](#_Toc10547770)

[5.3 How many women received active treatment (non placebo) across all studies combined? 18](#_Toc10547771)

[5.4 List the Types of Persons in the studies. 18](#_Toc10547772)

[Appendix 1: Course Resources 20](#_Toc10547773)

[Appendix 2: Update Files using TortoiseGit 20](#_Toc10547774)

# Introduction

Instructions in this document are specific to the workshop server environment, Graph Editor application, and workshop methods. The exercises represent one of many possible approaches to the material and make no claim to be best or recommended method. An instructor will complete the exercises at the same time as the class.

Your feedback is welcomed and encouraged. Please send your comments to: **tim.williams@PhUSE.eu**

**Materials**

Printed copies of the following will be supplied by the instructor:

* Exercises (this document)
* Info Sheet (unique to each person)
* Graph Editor Reference
* SPARQL Reference Sheet

All course content is available on GitHub at:

https://github.com/phuse-org/LinkedDataWorkshop/tree/master/**CSS2019**

**Symbols**

|  |  |
| --- | --- |
|  | Stop and wait. Do not proceed to the next section until instructed. |
| **!** | Cautions and warnings. Failure to follow instructions may lead to unanticipated problems. |
|  | Presentation by the instructor. |
| IdeaIcon_clean_20mm | Helpful tips and advice. |
| Image result for information icon | Additional information. |

# Server Login and Preparation

Instructions assume MS Windows operating system. *Mac users please ask the instructor for OSX login steps if you need assistance.*

1. On your laptop, ensure your screen resolution is set to 1600 x 900 or higher.
2. In the search box on the taskbar on your laptop, type **remote desk,** then select **Remote Desktop Connection** from the Programs section.﻿
3. If needed, select **Show Options** to see the tabs in the next step.
4. Select the **Display** tab. Ensure the Display Configuration slider is all the way to the right for **Large,** **Full Screen**:



1. On the **General** tab, enter values in the **Computer:** and **User name:** fields as supplied on your **Info Sheet**.
2. Click **Connect**.
3. When prompted, enter the password value from the **Info Sheet** and click **OK**.

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Exercises

# Create Your Study Graph

The workshop cloud server is used for all exercises.

# Open the Graph Editor

1. Open **Google Chrome** using the shortcut on the server's desktop.
2. Click on  in the shortcuts bar.
3. The Graph Editor opens and shows six nodes (see the Instructors’ display).

***Ask for assistance if your graph does not match the graph displayed by the instructor.***

# Add a Study Information

|  |  |
| --- | --- |
| **!** | Values unique to your graph are found on your **Info Sheet** and referenced in the exercises as <Info Sheet:[field]> . |

1. Create a **Study*n*** node. Follow along with the instructor to create a study node using the value from your Info Sheet. Ensure you always use the correct Prefix: value when creating nodes.

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Study>* | Image result for information icon*See Info Sheet for value* |

1. Create your study's NCT identifier number node, using the ***ct*** prefix.

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | ct |  |
| **Label:** | *<Info Sheet: NCTID>* | Image result for information icon*See Info Sheet for value* |

1. Create a link ***from*** the **Study*n*** node ***to*** the NCT identifier node. Edit the new link by double-clicking it and selecting the correct value from the list of available predicates, then click **Update/Hide**.

|  |  |
| --- | --- |
| **Predicate:** | ct:nct-id |

1. Create two Treatment Arm nodes for **Study*n*** using values from your **Info Sheet**. Match the upper and lowercase for the labels exactly. Ensure you use the ***eg*** prefix for these nodes.

Node 1 values:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Treat Arm One>* | Image result for information icon*See Info Sheet for value.* |

Node 2 values:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Treat Arm Two>* | Image result for information icon*See Info Sheet for value.* |

Confirm you now have two Treatment Arm (**TrtArm***n-n*) nodes in your graph.

1. Create a link ***from*** **Study*n*** ***to*** each of the two Treatment Arm nodes using the predicate **eg:trtArm** .

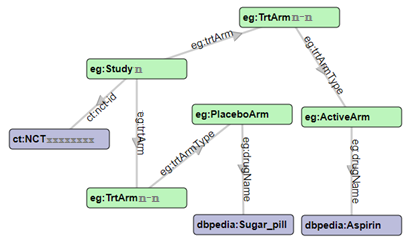
|  |  |
| --- | --- |
| **Predicate:** | eg:trtArm |

There should now be one **eg:trtArm** link from the Study to each of the Treatment Arm nodes.

1. Now join ***from*** each Treatment Arm node ***to*** one of the Treatment Type nodes (eg:ActiveArm, eg:PlaceboArm) using the predicate **eg:trtArmType**. It does not matter which arm is linked to the **PlaceboArm** or the **ActiveArm**.

|  |  |
| --- | --- |
| **Predicate:** | eg:trtArmType |

The Study and Treatment Arm portion of your graph should look similar to this diagram:



|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | Click  to save your graph in case anything goes wrong in the next steps. |
| Image result for information icon | You now have two Treatment Arms that are unique to your study, assigned to one of two Treatment Arm Types: PlaceboArm, ActiveArm. These "arm type" nodes are identical across all studies in the workshop and will allow identification of all persons who received active treatment. In real life, the Treatment Arm nodes would be connected to nodes for dosage and administration information unique to that study, resulting in a graph that would be too complex for this workshop. |

# Add Persons and Treatments

1. Add two Person identifier nodes using values from your **Info Sheet**.

Person 1 node:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Person One>* | Image result for information icon*See Info Sheet for values.* |

Person 2 node:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Person Two>* | Image result for information icon*See Info Sheet for values.* |

1. Link ***from*** one of the Person nodes ***to*** one of the Treatment Arm (**TrtArm*n-n***) nodes using the **eg:randomizedTo** predicate.

|  |  |
| --- | --- |
| **Predicate:** | eg:randomizedTo |

1. Link ***from*** the second Person node ***to*** the other Treatment Arm (**TrtArm*n-n***) node using another **eg:randomizedTo** predicate.

|  |  |
| --- | --- |
| **Predicate:** | eg:randomizedTo |

Now each Person is **randomizedTo** a Treatment Arm.

# Add Demographics

1. Link ***from***each of the two Person nodes ***to*** a Gender node using the **ncit:gender** predicate.

|  |  |
| --- | --- |
| **Predicate:** | ncit:gender |

1. Create one Given Name(First name) for eachPersonnode in the graph. Names have **Type: STRING** because they are string literals.

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | STRING |  |
| **Prefix:** | *NOT APPLICABLE* | Image result for information iconPrefix is not applicable to Type:STRING and will be ignored. |
| **Label:** | *<Any name you want>* |  |

1. Link ***from*** each Person node ***to*** their Given Name using the **schema:givenName** predicate.

|  |  |
| --- | --- |
| **Predicate:** | schema:givenName |

1. Create **Age** nodes for each Person as Type:INT (integer) values:

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | INT |  |
| **Prefix:** | *NOT APPLICABLE* | Image result for information iconPrefix is not applicable to Type:INT and will be ignored. |
| **Label:** | *<Any integer age value you want>* |  |

1. Link ***from***each Person ***to*** their age value using the **eg:age** predicate.

|  |  |  |
| --- | --- | --- |
| **Predicate:** | eg:age |  |

# Add another Person

1. Add another Person node to the graph:

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | IRI |  |
| **Prefix:** | eg |  |
| **Label:** | *<Info Sheet: Person Three>* |  |

*This node will represent you and your role in this study.*

1. Create a new node for your Given Nameas a Type:STRING.

Node values:

|  |  |  |
| --- | --- | --- |
| **Type:** | STRING |  |
| **Prefix:** | *NOT APPLICABLE* | Image result for information iconPrefix is not applicable to Type:STRING and will be ignored. |
| **Label:** | *<Your given (first) name>* |  |

1. Link ***from*** your Person node ***to*** your Given Name using the **schema:givenName** predicate.

|  |  |  |
| --- | --- | --- |
| **Predicate:** | schema:givenName |  |

1. Link ***from*** your Person node ***to*** your gender using the **ncit:gender** predicate.

|  |  |
| --- | --- |
| **Predicate:** | ncit:gender |

*The polite instructor will not ask you to create an age node and link. :)*

1. Your role in this study is Linked Data Expert. Link ***from*** your Person node ***to*** the **Study*n*** node using the **eg:LDExpert** predicate.

|  |  |  |
| --- | --- | --- |
| **Predicate:** | eg:LDExpert |  |

1. Congratulate yourself for becoming a Linked Data Expert, then Click  to save your graph.

# Review your graph

1. Use this checklist to confirm proper spelling and upper/lower case for node names in your graph. Make any necessary changes.

|  |  |  |
| --- | --- | --- |
| **Nodes** |  | √ |
| **eg:Study*<n>*** | ***<n>*** *is the same number for Study, TrtArm, and Person* |  |
| **eg:TrtArm*<n>*-*<x>*** | ***<x>*** = Number 1, or 2 |  |
| **eg:Person*<n>* <y>** | **<y>** = Number 1, 2 or 3 |  |
| **Node Colors** | ***18 Nodes total***  ***3*** *violet (purple)*  ***8*** *green*  ***3*** *gray*  ***2*** *orange*  *2 blue* |  |

1. Review your **predicates** by following the relationship ***from*** the Subject ***to*** the Object as indicated by the arrows in the graph. For example: "**Person21** **randomizedTo** **TrtArm1-1**" is correct. " **TrtArm1-1** **randomizedTo** **Person21**" is wrong and the direction of the link must be reversed.
2. Ask someone else (instructor, assistant, or other attendee) to take a look at your graph. A “fresh set of eyes” will find errors you may have missed. 😊
3. Save your graph if you make any changes to the values.

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor. |  |

# Export to TTL

1. Click on  to export the graph to TTL format.
2. Click "OK" to confirm.
3. If a "Download multiple files" message appears, click **Allow**.
4. Two files will appear at the bottom of the Chrome window: **WhiteBoardTriples.TTL** and **whiteboard.json.**
5. Click the dropdown arrow beside **WhiteBoardTriples.TTL** and select **Open.**
6. Follow along with the instructor's explanation of the TTL file content.
7. Find the various ***Subjects, Predicates,*** and ***Objects*** you created in your graph.

|  |  |
| --- | --- |
| Image result for information icon | If you create the TTL more than once, you will see multiple files that include a number in their name:  WhiteBoardTriples(1).TTL  WhiteBoardTriples(2).TTL  etc.  Always use the highest numbered (most recent) file. |

1. If you find any errors, correct them using the Graph Editor and then  again using the steps in this section.

|  |  |
| --- | --- |
| **!** | Leave the graph open in Chrome. You will refer to it later when creating queries. Ask for assistance if you accidentally close or refresh the Graph Editor window. |

# Validate the Data

1. Open **Windows Explorer** and click on the **R** folder under **Quick access**.
2. Double-click the file **TTLValidation.R** to open it into RStudio.
3. **Run** the code by clicking the Run App icon 
4. Click **Browse...** under **.TTL File**
5. **** Scroll up in the left pane to find the **Quick access** links and select **Downloads**.
6. Double-click on **WhiteBoardTriples.TTL** to load the file into the app.

|  |  |
| --- | --- |
| **!** | Remember to load the most recent version of your TTL file, which may have a number in the name : WhiteBoardTriples(1).TTL, WhiteBoardTriples(2).TTL , etc. Ask for assistance if needed. |

1. A default query appears in the app. Click **Run query** to run the query and view the result in the **Query Result:** area.
2. Review the values in the **Query Result** and compare them to your graph.
3. Click on the  tab. If you see the message "All QC Checks Passed", proceed to **Step 10**. ***If not all checks pass:***
   1. Review the values listed and compare them to your graph. Do the values or their prefixes need to change to match the values given in the exercises?
   2. Ask for assistance if the problem is not obvious.
   3. Return the Graph Editor to make corrections as needed.
   4. Export to TTL again using the steps in section **1.7 Export to TTL**, then re-run the validation steps in this section until you receive the message "All QC Checks Passed" or the instructor approves any items displayed on the QC Check page.

|  |  |
| --- | --- |
| **!** | Unless advised by the instructor, do not proceed past this step until the QC Check page displays "All QC Checks Passed". Failure to validate your graph may cause errors in later exercises. |

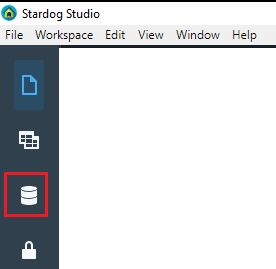
1. Click on the  tab at the top of the app to view a network graph of the query result.
2. Explore the visualization:

|  |  |  |
| --- | --- | --- |
|  | **Mouse** | **Touchpad** |
| Reposition graph | Hold down the left mouse button and drag | Hold finger on touchpad, drag |
| Zoom | Scroll wheel up/down | Move two fingers on touchpad together to zoom in, apart to zoom out |
| View labels | Position cursor over node or link | Position cursor over node or link |
| Node Connections | Click on node | Tap on node |

1. Close both the **RShiny app** and **RStudio** when you are done browsing your graph. If prompted to "Save workspace image..." Click "Don't Save".

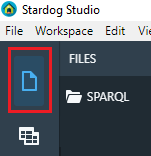
|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor. |  |

# Upload to Database

1. Double-click on the Stardog Studio icon on your desktop.
2. Click 
3. Click **Connect** (make no changes)
4. Click the database icon on the far-left menu.
5. Select **LDWStudy**
6. Click **LOAD DATA**.
7. In the Load Data window, scroll to top in the left part of the window to find the Quick access icon.
8. Select the **Quick access | Downloads** folder and double-click **WhiteBoardTriples.ttl** to select the file. Remember to use the most recent file if you have multiple versions.
9. Double click the TTL file or highlight it and click Open.
10. The message: ***Data successfully added to LDWStudy***  appears briefly in the lower right of the screen and the LDWStudy database should show: **Triple Count: 18**

# Query Your Graph

# Show all triples

Find all the Subjects, Predicates, and Objects in your graph.

1. Click on the Workspace icon in the upper left.
2. The pane will then display a list of query files.
3. Click on **210-AllTriples.rq** to display it in the query window.
4. Click the **Select Database** drop-down and select **LDWStudy**.
5. Click **Run** icon.
6. Scroll through the results in the Results pane.
7. Compare this view to what you expected after viewing the TTL file in an earlier exercise.

# How many people *randomizedTo* treatment arms inthe study?

Find the number persons who are randomized to treatments arms (including both Active and Placebo arms) by counting the number of Person nodes that have **randomizedTo**predicates.

1. Load the SPARQL query **220-CountRandomized.rq**.
2. Examine the query and refer back to your graph in the Graph Editor while the instructor explains the query.
3. **Run** the query.
4. Is this this result you expected, based on your graph? Why are you (the Linked Data Expert) not included in the count?

# Find the names of people who are randomized to treatment arms.

In this exercise you want to obtain the givenName values of the Persons who participate in the study. Refer back to your graph in the Graph Editor to create the query using the steps below, or skip down to the SOLUTION: to load the query file.

Try modifying the query from **Exercise 2.2** using the next steps as a guide.

1. Return the person's name (givenName) in the SELECT statement:

SELECT ?givenName

1. In the WHERE statement, identify Person nodes that have the **randomizedTo** predicate:

?person eg:randomizedTo ?treatArm ;

1. Then follow from that same Person node using the **givenName** predicate to find their name.

?person eg:randomizedTo ?treatArm ;

schema:givenName ?givenName .

1. **Run** your query and view the results. If you have trouble writing the query, load and run the file:

SOLUTION: **230-NamesOfRandomized.rq**

# List the names of the people randomized to treatment arm type and the drug they received.

Extend your query used in **Exercise 2.3** to obtain the treatment arm type and treatment drug in that arm.

1. Modify the SELECT statement from **Exercise 2.3** to select both treatment arm type (?trtArmType) and drug name (?drugName)n addition to person's ?givenName variable.

SELECT ?givenName ?trtArmType ?drugName

1. Refer to your graph in the Graph Editor. You want to follow the graph from the **?trtArm** nodes along the **eg:trtArmType** path (predicate) to the type of treatment ( **?trtType** ) .

?person eg:randomizedTo ?trtArm ;

schema:givenName ?givenName .

?trtArm eg:trtArmType ?trtArmType .

1. Next, your query must follow the graph from the trtType to the drugName assigned to that treatment arm:

?trtArmType eg:drugName ?drugName .

1. **Run** the query.

SOLUTION: **240-NamesAndTreatments.rq**

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Extend to Other Graphs (Federated Query)

These exercises demonstrate how easy it is to supplement your local graph using information from other Knowledge Graphs.

# Your study in ClinicalTrials.gov

What information is available for your study using an endpoint that has information uploaded from ClinicalTrials.gov?

1. Load the SPARQL query **310-StudyInfoCTGov.rq** into Stardog Studio.
2. Follow along with the instructor while they explain the query.
3. **Run** the query.
4. Examine the information returned from the query. What can you learn about the study from this information?

SOLUTION: **310-StudyInfoCTGov.rq**

# Study Title and Phase

The exercise Modify the query used in Exercise **3.1** to obtain just the Study Title and Study Phase.

1. Review the query results from Exercise **3.1** . Locate the predicates (p) for Study Title and Study Phase. Consider how would you modify the query in Exercise **3.1** to obtain just this subset of information for your study?
2. Load the SPARQL query **320-StudyTitlePhaseCTGov.rq** into Stardog Studio.
3. Follow along with the instructor while they explain the query.
4. **Run** the query.
5. Examine the information returned from the query. What Phase is your study?

SOLUTION: **320-StudyTitlePhaseCTGov.rq**

# Information about the Study Drug on DBpedia

What information is available for the study drug on DBPedia?

1. Load the SPARQL query **330-StudyDrugInfo.rq** into Stardog Studio.
2. Follow along with the instructor while they explain the query.
3. **Run** the query.
4. Examine the information returned from the query. What can you learn about the study from this information?

SOLUTION: **330-StudyDrugInfo.rq**

# Drug Abstract

Obtain the Drug Abstract from DBPedia in English.

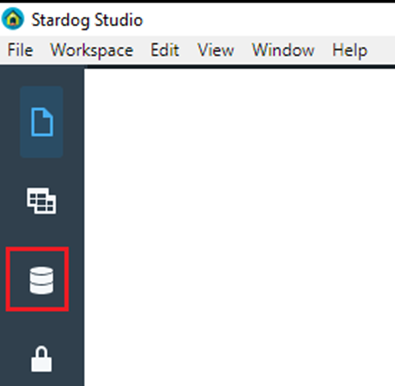
1. Load the SPARQL query **340-EnglishAbstract.rq** into Stardog Studio.
2. Follow along with the instructor while they explain the query.
3. **Run** the query.

SOLUTION: **340-EnglishAbstract.rq**

# Ontology and Inference

In this section you will add an ontology to your local graph data and employ a reasoner.

# Review the Ontology

1. Open **Windows Explorer** and click on the **data** folder under **Quick access**
2. Double-click on the file **StudyOntology.TTL** to open it into a text editor.
3. Follow along with the instructor's explanation of the file.

# Add the Ontology to the LDWStudy database

1. Return to the Stardog Studio.
2. Click the database icon on the far-left menu.
3. Select **LDWStudy**.
4. Select **LOAD DATA**.
5. Select the **Quick access | data** folder and double-click **StudyOntology.TTL** to select the file.
6. Double click the TTL file or highlight it and click Open.
7. The message: ***Data successfully added to LDWStudy***  appears briefly in the lower right of the screen.

# Find the names of HumanStudySubjects in your study.

*Your original data contains no definition of a HumanStudySubject. For the purpose of this exercise, a HumanStudySubject is someone who is randomizedTo a treatment arm which includes both active arms and placebo arms.*

|  |  |
| --- | --- |
|  | Follow along with the instructor as they describe how HumanStudySubject can be inferred by the ontology using the links and entities in your original data. |

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | In SPARQL queries, the **rdf:type** link can be shorted to **a** and interpreted as "*is a*".  Example:  You read: eg:Person11 rdf:type eg:HumanStudySubject  as: "Person11 ***is a*** HumanStudySubject."  Instead of querying using **randomizedTo** , as in:  ?person eg:randomizedTo ?treatArm ;  With a Reasoner you can query using the *inferred* entity **eg:HumanStudySubject**.  ?person a eg:HumanStudySubject; |

1. Click on the workspace icon  to return to the query editor window.
2. Open the query file **230-NamesofRandomized.rq** that you used in **Exercise 2.3 .** Use the information in the advice box IdeaIcon_clean_20mmabove this step to modify the queryto find ?person "is a" HumanStudySubject.
3. Turn on **Reasoning** button. 
4. **Run** the query.
5. Refer back to your graph in the Graph Editor and be amazed! There is no "Human Study Subject" defined in your original graph. Using an ontology and reasoner *inferred* new information.
6. What happens if you turn OFF the reasoner and run the query? Don't forgot to turn the reasoner back ON for the next exercise.

SOLUTION: **430-NameHumanStudySubject.rq**

# Write a reasoner-based query to find all types (classes) of Persons

Query to find all persons associated with the study, not just those randomized to treatment arms.

1. Write a query that uses the ***superclass*** of the **HumanStudySubject** and **DataExpert** classes.

CAUTION: The superclass prefix is not "eg" because it is defined in an external ontology!

1. Return the following values: Person IRI, Given Name of the Person, and the Person Type

HINT: Use the "is a" predicate: ?person a ?person\_type

SOLUTION: **440-NamesPersonTypes.rq**

1. The query result returns all the inferred "types of things" for Person nodes, not just their role in the study. You can return just the roles in the study by filtering on the http name space used for the Study graph.

FILTER ( strstarts(str(?person\_type), "http://example.org/LDWorkshop#") )

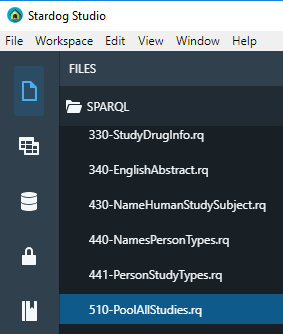
SOLUTION: **441-PersonStudyTypes.rq**

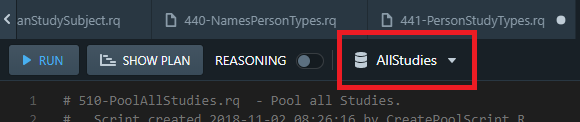
|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Merge Studies

# Create the Data Pool

|  |  |
| --- | --- |
| **!** | The instructor will perform these steps first and provide instruction on how the class should proceed to create the pool. |

1. Click on query **510-PoolAllStudies.rq** to load it into the Stardog studio editor.
2. **Select the AllStudies database** in the dropdown at the top of the SPARQL editor.



1. Confirm the 510-AllStudies.rq script is selected to run on the AllStudies Graph. Click **Run**.
2. The query may take some time to execute while it pulls data from all of the other graphs.
3. After the query has completed, **run** the query **515-PoolTripleCount.rq** on the **AllStudies** graph. Compare your triple count to the number provided by the instructor to ensure you have the correct data.

SOLUTION: **510-PoolAllStudies.rq , 515-PoolTripleCount.rq**.

# How many HumanStudySubjects participated in all Studies?

1. Ensure the **Reasoner** is turned **On**.
2. Load the query **520-AllStudiesCountHumanSubjects.rq** and examine the query.
3. **Run** the query.

|  |  |
| --- | --- |
| IdeaIcon_clean_20mm | If the count is 0, turn the Reasoner ON. |

BONUS Question: Alter the query to count **all people** involved in all studies (not just HumanStudySubjects).

HINT: a) What is the superclass of **HumanStudySubject**?

b) Ensure you are using the correct prefix!

SOLUTION to BONUS: **525-AllStudiesCountPeople.rq**

# How many women received active treatment (non placebo) across all studies combined?

1. Use the graph of your study in the Graph Editor to determine the paths that need to be followed to obtain the solution.

HINT : follow the pattern: ?person -- gender -- **Female** ;

-- randomizedTo --- ?trt\_arm .

?trt\_arm -- treatmentArmType -- **ActiveArm**.

SOLUTION: **530-AllStudiesWomenActiveTreat.rq**

# List the Types of Persons in the studies.

*You may be getting tired so consider skipping directly to the Solution to see one way of solving this question.*

1. List all Person IRIs, their givenName, their assigned Study, and their role in that study (as a **HumanStudySubject** or **DataExpert**). Ensure the Reasoner is turned ON.

HINT: a) Find **Person**s using the logical OR operator for their role in the study:

?person eg:participatesIn|eg:LDExpert ?study.

b) To return only "person types", you need to use the ontology's Person subclasses:

?personType rdfs:subClassOf schema:Person ;

1. Review the results. **Do you find anything surprising in the pooled results?**

SOLUTION: **540-AllStudiesPersonTypes.rq**

Congratulations- You've made to the End of the Exercises!

Please Sign out from the server using **Windows | Person Icon | Sign out**.

|  |  |  |
| --- | --- | --- |
|  | Stop here and wait for the instructor.  Presentation follows |  |

# Appendix 1: Course Resources

* [Linked Data Introduction](https://www.youtube.com/watch?v=4x_xzT5eF5Q)

https://www.youtube.com/watch?v=4x\_xzT5eF5Q

* [SPARQL in 11 minutes](https://www.youtube.com/watch?v=FvGndkpa4K0)

https://www.youtube.com/watch?v=FvGndkpa4K0

* [Learning SPARQL](http://www.learningsparql.com/)

http://www.learningsparql.com/

-------------------------------------------------------------------------------------------------------

[Hands-on Workshop Github Repository](https://github.com/phuse-org/LinkedDataWorkshop/tree/master/EUConnect18/presentation/images/rdf_logo.png)

Course content is available at:

https://github.com/phuse-org/LinkedDataWorkshop/tree/master/EUConnect18

# Appendix 2: Update Files using TortoiseGit

Follow these steps if the instructor asks you to update the exercise files.

1. Click on the **LDWorkshop** shortcut  on the desktop.
2. Right-click on the **EUConnect18** folder and select **Git Sync** from the menu.



1. Click **Pull**
2. Click **Close** when the command completes. You have updated all content to the latest version as needed for the exercises.