COMP 6741

Intelligent Systems

Project-1 Report

Roboprof

Submitted To: Prof. Dr. René Witte

Date: 22 March 2024

Team: AK_G_04

Aryan Saxena (**40233170**)

Benjamin Douglas (**40264251**)

GitHub Link

We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality

Table of Contents

1. Vocabulary

A. Classes

a. **University, Course, Lecture, Topic, Student:** These are the primary classes, each representing fundamental concepts such as educational institutions, units of teaching, individual instructional sessions, subjects covered, and learners.

b. **LectureContent (with subclasses Slides, Worksheets, Readings, OtherMaterial):** This hierarchy models different types of materials provided in lectures, offering a structured way to represent educational resources.

B. Properties

a. **General Properties like hasName, hasLink**: These are used across different classes to denote names and links respectively, applicable to various resources.

b. **Relationships between Universities and Courses (offersCourse), Courses and their attributes (courseSubject, courseNumber, etc.), Lectures and their specifics (lectureNumber, lectureName, etc.):** These properties define how different entities are interconnected and describe specific attributes of courses and lectures.

c. **Properties for Student interactions (studentID, completedCourse, courseGrade, hasCompetency):** These capture student-related data, including their identity, courses they've completed, their grades, and competencies in specific topics.

d. **Linking Topics with Courses and Lectures (isTopicOfLecture, isTopicOfCourse):** These properties explicitly connect topics to the lectures and courses where they are covered.

2. Data (CSVs)

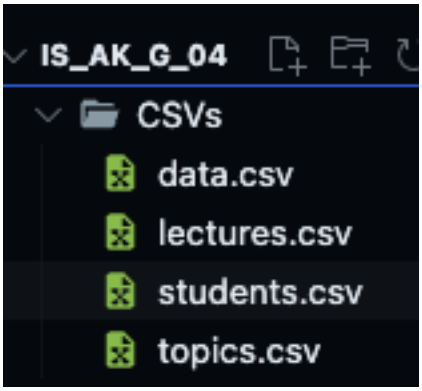


Fig 1: Structure

Secondly, Focusing on the datasets we're using as the foundation for creating our .**ttl (Turtle)** files, which are essential for our RDF framework.

The datasets are structured in CSV format, coming from various educational facets:

A. Data.csv: Harvested from Concordia's open data portal, this extensive file includes course details such as IDs, titles, units, descriptions, prerequisites, and equivalent courses. This will allow us to define and connect courses within our RDF schema accurately.

B. Lectures.csv: Contains information about individual lectures linked to their respective courses, including lecture IDs, numbers, names, content, and additional resources.

C. Students.csv**:** Provides a snapshot of student information, including their names, ID numbers, email addresses, the courses they've taken, their grades, and areas of competency.

D. Topics.csv: Lists out topics covered in various lectures and courses, along with IDs, names, and links to external resources for more information.

These CSV files are not just rows and columns of data; they represent the backbone of our semantic data model, allowing for an intricate web of interlinked information reflecting the complex reality of university academic structures.

3. Fetching the Data and Creating Turtle Files

The fourth point in our report delves into the process of **transforming CSV data into Turtle (TTL)** files, which are pivotal for populating our RDF framework.

Fetching the Data and Creating Turtle Files:

A. Setting Up the Environment:

- Import necessary libraries in Python: pandas for data manipulation, rdflib for RDF operations, SPARQLWrapper for querying.
- Define paths to CSV source files and destination TTL files.

B. Initiating RDF Graphs:

- Create RDF Graph instances for each TTL file.
- Define and bind the namespaces required for RDF triples such as 'ex', 'foaf', 'rdf', 'rdfs', 'xsd', 'dbo', and 'dbr'.

C. Populating the University TTL:

- Run a SPARQL query to fetch data about universities from DBpedia.
- Manually add Concordia University if it's not present.
- Serialize and save the university data into 'universities.ttl'.

D. Populating the Courses TTL:

- Read 'data.csv' containing course information from Concordia's open data.
- Iterate through each row, creating URIs for courses and adding relevant triples (such as course subject, number, description, etc.).
- Serialize and save the course data into 'courses.ttl'.

E. Populating the Students TTL:

- Read 'students.csv' for student information.
- For each student, create a URI and add triples denoting their name, email, student ID.
- For courses completed by the student, create blank nodes to represent the relationship between the student, the course, and their grade.
- Serialize and save the student data into 'students.ttl'.

F. Populating the Lectures TTL:

- Read 'lectures.csv' for lecture details.
- For each lecture, create a URI and add triples detailing the lecture's number, name,

content link, and additional resources.
- Serialize and save the lecture data into 'lectures.ttl'.

G. Populating the Topics TTL:

- Read 'topics.csv' for academic topics discussed in lectures.
- For each topic, create a URI and add triples detailing the topic's name, provenance, and link.
- Link each topic to its corresponding lecture and course.
- Serialize and save the topic data into 'topics.ttl'.

H. Error Handling and Validation:

- Ensure that the code handles missing data gracefully, such as checking for null values.
- Validate the URIs created and ensure proper encoding to avoid issues in RDF triples.

I. Serialization:

- Convert the populated graphs to Turtle format.
- Serialize and save the graphs to the defined paths, creating the TTL files for use in RDF stores or other applications.

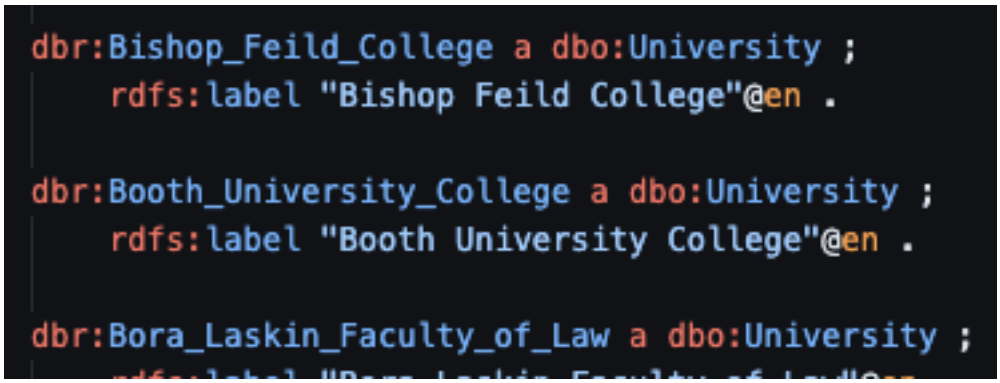
Each step ensures the data is **correctly structured and relationships** are defined following the RDF standards, allowing the semantic data to be queried and understood in a machine-readable format.

4. Turtle (.ttl) Files

For the fourth part of our report, we'll present visual documentation of the Turtle (TTL) files generated. Here's how it will be structured for inclusion in the Word document:

A. Universities.ttl

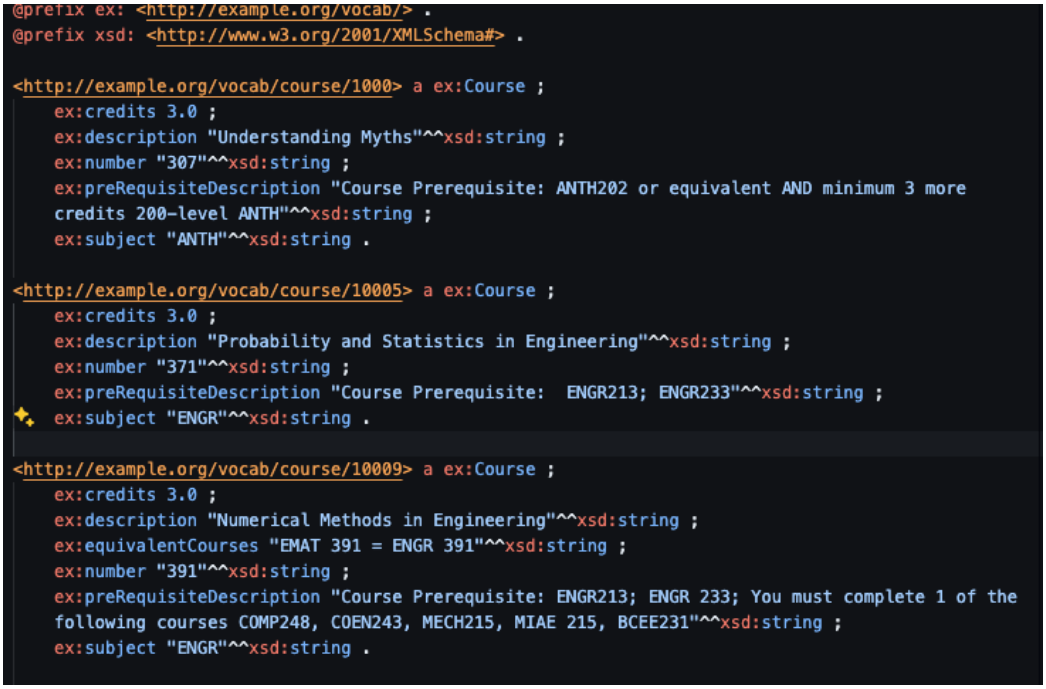
- *Screenshot*:



- **Description**: Captures RDF data of various universities, including Concordia, each with its own URI and a label in English.

B. Courses.ttl

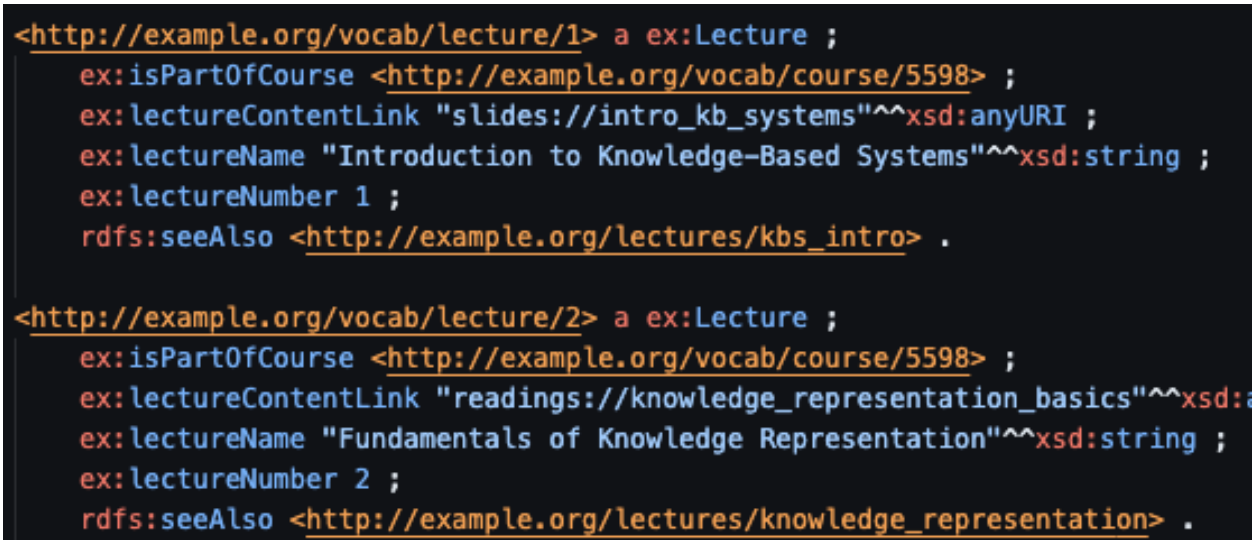
- *Screenshot*:



- **Description**: Outlines course offerings with details like subject and prerequisites, structured into RDF format.

C. Lectures.ttl

- *Screenshot*:



- **Description**: Each lecture is detailed with a unique URI, showing its connection to courses and the topics discussed.

D. Students.ttl

- *Screenshot*:

```
<http://example.org/vocab/student/S1001> a ex:Student ;
    ex:completedCourse [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5477> ;
            ex:courseGrade "C+"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5605> ;
            ex:courseGrade "A"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5527> ;
            ex:courseGrade "A-"^^xsd:string ],
        [ a ex:CompletedCourse ;
            ex:course <http://example.org/vocab/course/5570> ;
            ex:courseGrade "A"^^xsd:string ] ;
    ex:hasCompetency <http://example.org/vocab/course/5409>,
        <http://example.org/vocab/course/5411> ;
    ex:studentID "S1001"^^xsd:string ;
    foaf:mbox "aryan.saxena@concordia.ca"^^xsd:string ;
    foaf:name "Aryan Saxena"^^xsd:string .
```

- **Description**: Student profiles are represented here, linking them to the courses they've taken and their performance.

E. Topics.ttl

- *Screenshot*:

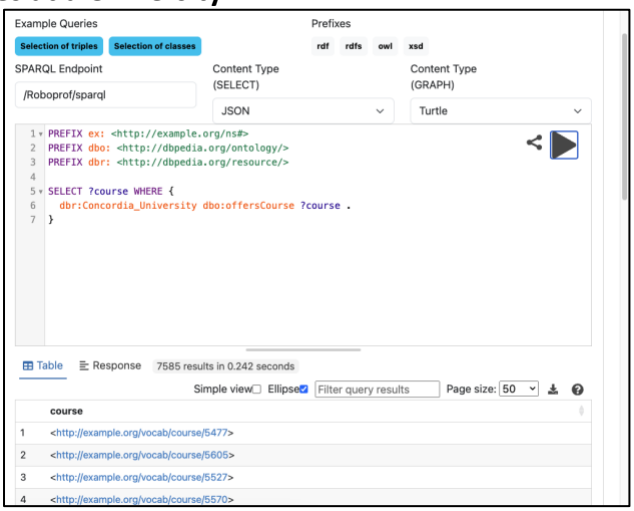
```
<http://example.org/vocab/topic/2> a ex:Topic ;
    ex:isTopicOfCourse <http://example.org/vocab/course/5598> ;
    ex:isTopicOfLecture <http://example.org/vocab/lecture/2> ;
    ex:topicLink <https://dbpedia.org/page/Knowledge_representatio
    ex:topicName "Knowledge Representation"^^xsd:string ;
    ex:topicProvenance "Lecture 2"^^xsd:string .

<http://example.org/vocab/topic/3> a ex:Topic ;
    ex:isTopicOfCourse <http://example.org/vocab/course/5603> ;
    ex:isTopicOfLecture <http://example.org/vocab/lecture/3> ;
    ex:topicLink <https://dbpedia.org/page/Automata_theory> ;
    ex:topicName "Automata Theory"^^xsd:string ;
    ex:topicProvenance "Lecture 3"^^xsd:string .
```

- **Description**: Defines the academic topics taught, linking them to the relevant lectures and courses.

5. Queries

Query 1: Finding Courses at a University



This query searches for all the courses offered by a specific university, showcasing the educational opportunities available there.

Query 2: Discovering Courses Covering a Topic

It identifies which courses discuss a particular topic, highlighting the academic exploration of specific subjects.

Query 3: Exploring Topics in a Lecture

This query reveals what topics are covered in a specific lecture of a course, offering a peek into the lecture's focus.

Query 4: Listing Subject-Specific Courses at a University

It lists all courses under certain subjects offered by a university, pointing out specialized educational paths.

Query 5: Materials Recommended for a Topic in a Course

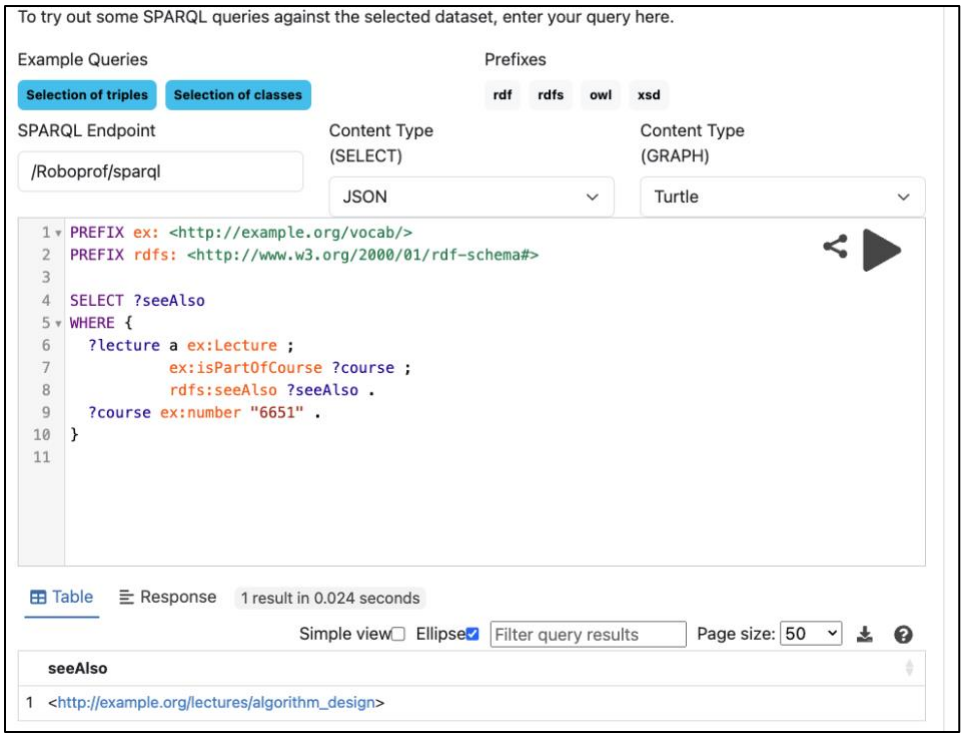
*No result because no matches.*This query fetches slides and readings recommended for studying

a particular topic within a course, guiding resourceful learning.

Query 6: Credit Value of a Course

It determines the credit value of a specific course, informing on its academic weight.

Query 7: Additional Resources for a Course

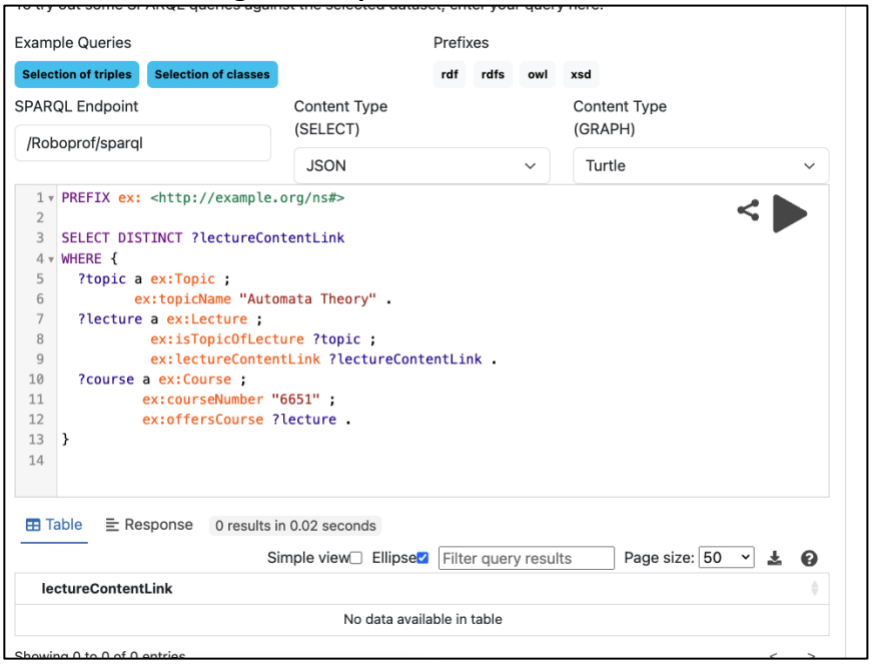


This query finds extra resources, like web links, for a course, providing avenues for extended learning.

Query 8: Content Available for a Lecture

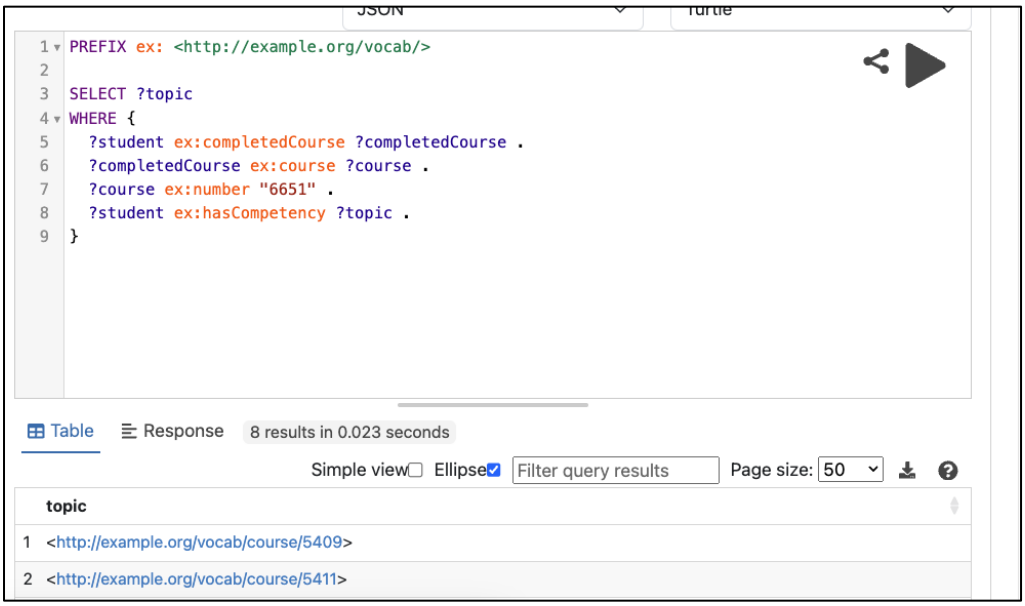
No result because no matches. It identifies all the content types available for a specific lecture, giving insights into the lecture's materials.

Query 9: Recommended Reading for a Topic



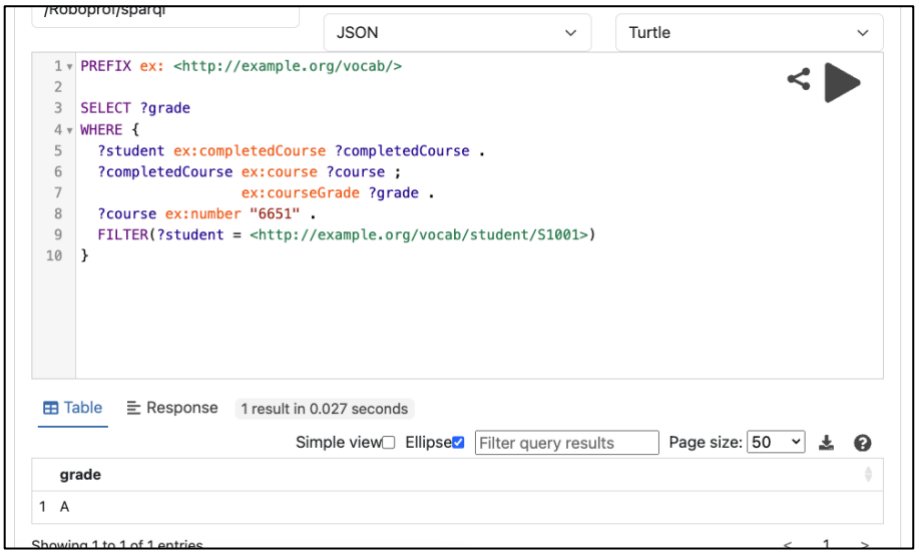
No result because no matches. This query lists reading materials recommended for a topic within a course, aiding in-depth study.

Query 10: Competencies Gained from a Course



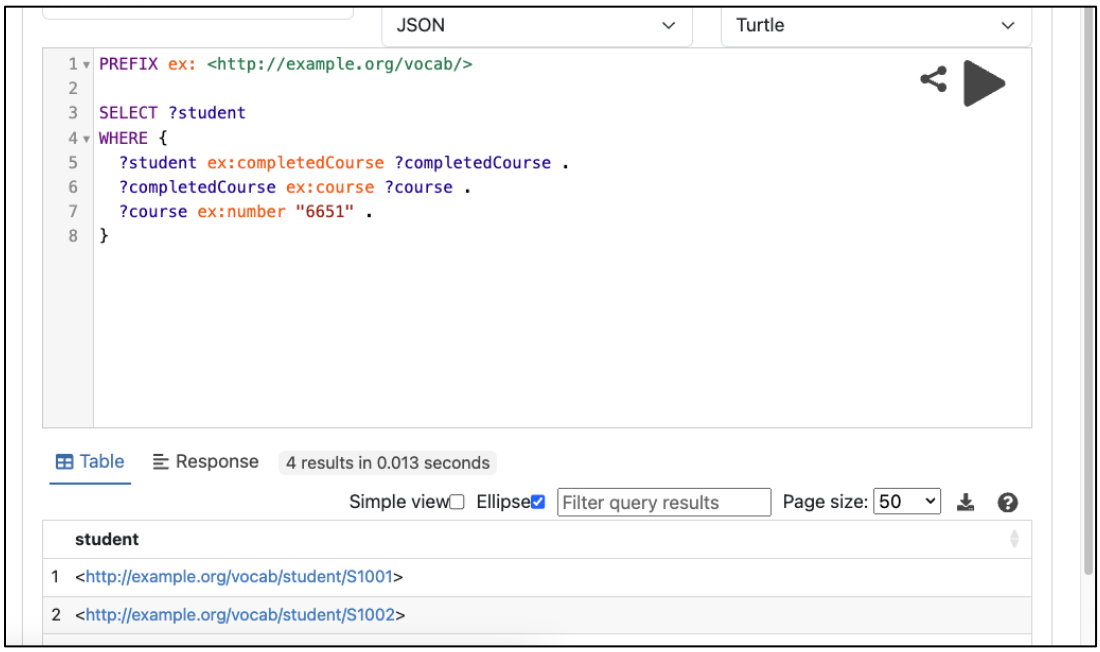
It explores the topics a student becomes proficient in after completing a course, reflecting on the learning outcome.

Query 11: Student's Grades in a Course



This query fetches the grades a student achieved in a course, reflecting their academic performance.

Query 12: Students Who Completed a Course



It identifies students who have completed a specific course, showing who has advanced through that academic challenge.

Query 13: Printing a Student's Transcript

This query compiles a student's courses and grades into a transcript, summarizing their academic journey.

6. Running Procedure

Fig: File Structure

The queries executed using our local Fuseki server, which provides a SPARQL endpoint for interacting with RDF data. Here's the breakdown of our process:

A. Setting up the SPARQL Endpoint:

- Initialized a SPARQL Wrapper with the endpoint URL pointing to our local Fuseki server.

B. Crafting Queries:

- Wrote SPARQL queries tailored to extract specific information from the RDF data.
- Each query was designed to target a distinct aspect of our dataset, ranging from course details offered by universities to lecture materials and student grades.

C. Executing Queries:

- Set each query to the SPARQL endpoint through the SPARQL Wrapper.
- Configured the return format to CSV for easy data manipulation and review.

D. Saving Results:

- Executed the queries and saved the results directly to corresponding CSV files, named after the specific query number, such as "q13.csv" for the 13th query.
- This structured approach allowed us to keep our results organized and accessible for further analysis.

E. Iterative Approach:

- Repeated this process for a series of queries, incrementally building up our collection of CSV files.
- This iterative method allowed for flexibility and the ability to refine queries as needed.

F. Verification:

- After running each query, we verified the output by checking the CSV files, ensuring the results matched our expectations.

The queries served as a powerful tool to dissect and understand our RDF data, bringing insights and clarity to the academic structure we've modeled. The systematic approach ensured a consistent, clear, and documented process, reflecting the depth and interconnectivity of our semantic dataset.