Group 38
Haoqi Yu, Bjørn Magnus Hoddevik

Approach to the problem

We began by looking into ways to convert CSV files into RDF triples given our RDFS produced in Assignment 2. We quickly found a python library called “rdfpandas”, which showed promise. However, it was abandoned for the more powerful, and therefore complex, “rdflib” library. After a quick prototyping session, with focus on the student csv file, we committed to this approach. Perhaps some more research into prebuilt tools would’ve saved some frustration.

By working iteratively, with one csv file at a time, we quickly got through the data but not without hiccups. We had to revise our work from Assignment 2 several times, as some attributes were simply missing or our ER-diagram from Assignment 2 had failed to satisfy the data. Moreover, there were some problems that were revealed when we began working on the queries. These problems were related to the way we were producing our RDF triples. We will discuss this in the next section.

We approached each query one at a time, starting with the first. We started small, and built up each query line by line, verifying that nothing went wrong for each step.

Decisions and assumptions

The most challenging part was how we would deal with weak entities and relations. We decided to treat both as a RDFS bag, although we are not certain that is the right term. We treated each ER-diagram entity as the main subject, and then every attribute, weak entity and relation was a predicate with their value as the object. While attribute objects were simple, the weak entities and relations were more tricky. We used BNodes, added them as an object connected to the entity subject. And then used the BNode as the main subject with the attributes of the weak entity or relation as objects.

We then decided to only add the relations to one side, meaning the relation “Teaches” was only added to teachers. There is no reverse of this. Which side the relation was added to was determined by which csv file had the information, meaning that if the csv had information about teachers, then the relation was added to teachers.

I am not aware of any assumptions we made, outside of assuming our RDFS definitions from Assignment 2 (after the aforementioned changes) captured all the relevant data without losing information.

Scripts used

We have attached the file “rdfs_and_csv_to_triples.py” which takes the input file “assignment_2_rdfs_revised.txt” and the given csv files, and outputs the finished graph in the file “graph.txt”. To run the scripts, we have also attached the “requirements.txt” file.

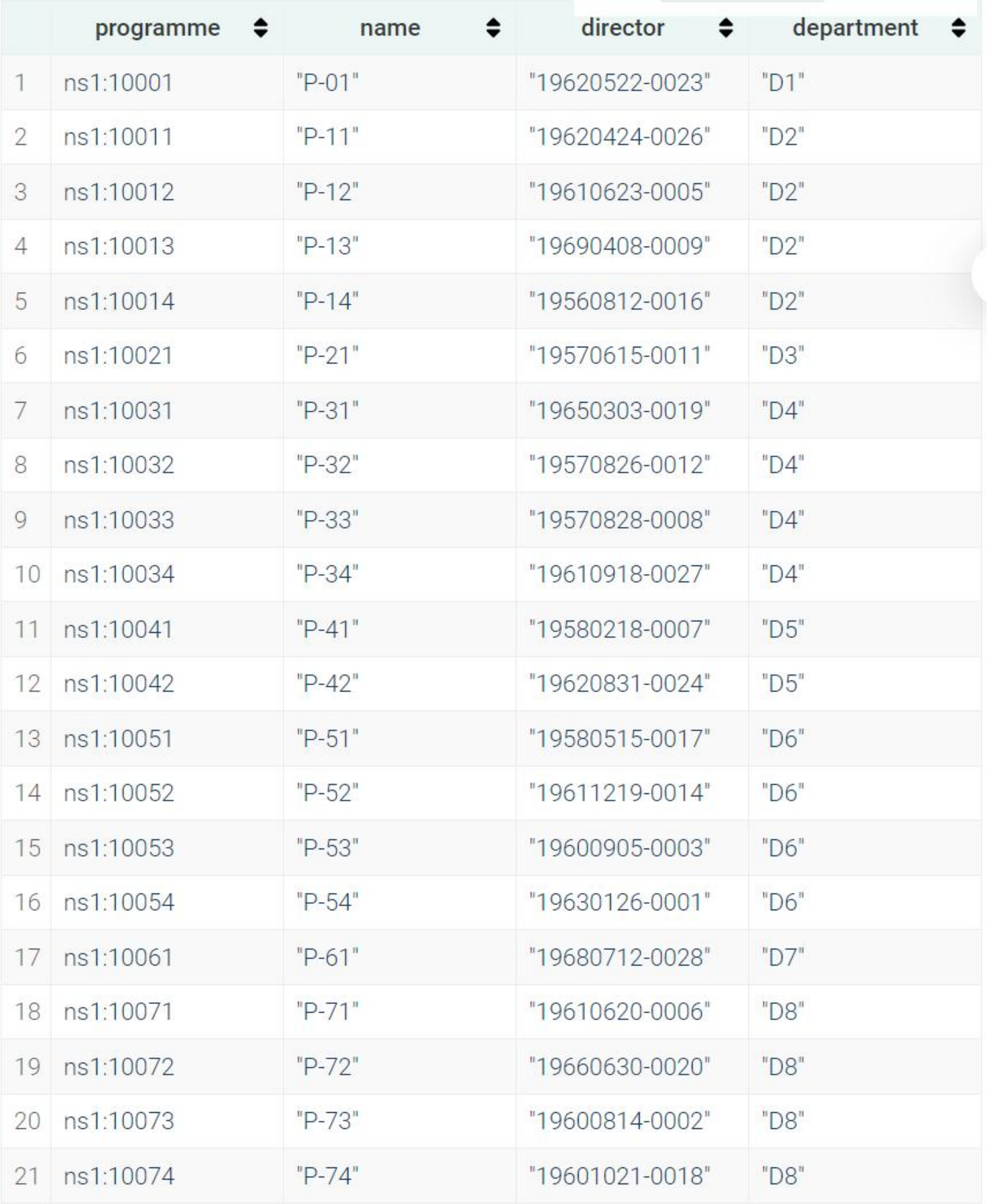
Queries and result

See next pages

1. Find the name, director and department of all programmes.

```
PREFIX ns1: <http://example.org/>
SELECT ?programme ?name ?director ?department WHERE {
        ?programme a ns1:Programme .
        ?programme ns1:programmeName ?name .
        ?programme ns1:programmeDirector ?director .
        ?programme ns1:OwnedByDepartment ?department
}
```

Result：



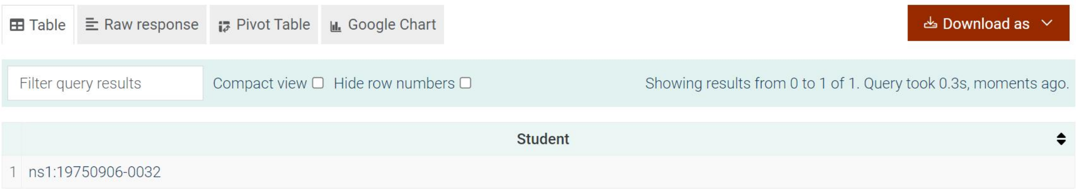
	programme	name	director	department
1	ns1:10001	"P-01"	"19620522-0023"	"D1"
2	ns1:10011	"P-11"	"19620424-0026"	"D2"
3	ns1:10012	"P-12"	"19610623-0005"	"D2"
4	ns1:10013	"P-13"	"19690408-0009"	"D2"
5	ns1:10014	"P-14"	"19560812-0016"	"D2"
6	ns1:10021	"P-21"	"19570615-0011"	"D3"
7	ns1:10031	"P-31"	"19650303-0019"	"D4"
8	ns1:10032	"P-32"	"19570826-0012"	"D4"
9	ns1:10033	"P-33"	"19570828-0008"	"D4"
10	ns1:10034	"P-34"	"19610918-0027"	"D4"
11	ns1:10041	"P-41"	"19580218-0007"	"D5"
12	ns1:10042	"P-42"	"19620831-0024"	"D5"
13	ns1:10051	"P-51"	"19580515-0017"	"D6"
14	ns1:10052	"P-52"	"19611219-0014"	"D6"
15	ns1:10053	"P-53"	"19600905-0003"	"D6"
16	ns1:10054	"P-54"	"19630126-0001"	"D6"
17	ns1:10061	"P-61"	"19680712-0028"	"D7"
18	ns1:10071	"P-71"	"19610620-0006"	"D8"
19	ns1:10072	"P-72"	"19660630-0020"	"D8"
20	ns1:10073	"P-73"	"19600814-0002"	"D8"
21	ns1:10074	"P-74"	"19601021-0018"	"D8"

2. Find the names of all students who worked as teaching assistants in courses given by the D3-2 division in study period 2 in academic year 2023/2024.

```
PREFIX ns1: <http://example.org/>
SELECT ?Student ?CourseOffering ?studyperiod ?academicyear ?division WHERE {
?Student a ns1:TeachingAssistant .
?Student ns1:Teaches [ns1:teachesCourseOfferingInstanceID    ?CourseOffering].

?courseOffering ns1:courseOfferingStudyPeriod ?studyperiod .
?Student ns1:teacherDivisionName ?division .
?courseOffering ns1:courseOfferingAcademicYear ?academicyear .
FILTER regex(?academicyear,"2023-2024")
FILTER (?studyperiod=2.0)
FILTER regex(?division,"D3-2")
}
```

Result: TA 32



Showing results from 0 to 1 of 1. Query took 0.3s, moments ago.

	Student
1	ns1:19750906-0032

3. Find all teachers who are assigned more than 120 hours in course 1015 in study period 1 in academic year 2018/2019.

```
PREFIX ns1: <http://example.org/>
SELECT ?Teacher ?CourseOffering ?assignedHours ?period ?courseCode ?academicyear WHERE {

?Teacher a ns1:TeachingAssistant .
?Teacher ns1:Teaches ?teachingRecord .
?teachingRecord ns1:teachesCourseOfferingInstanceID ?CourseOffering ;
                    ns1:teachesAssignedHours ?assignedHours .

?courseOffering ns1:courseOfferingCourseCode ?courseCode.
?courseOffering ns1:courseOfferingStudyPeriod ?period .
?courseOffering ns1:courseOfferingAcademicYear ?academicyear .
FILTER (?assignedHours>120)
FILTER (?period=1.0)
FILTER (?courseCode="1015")
FILTER regex(?academicyear,"2018-2019")

}
```

4. Find all students registered for course instance I-910 that were not registered for course instance I-911.

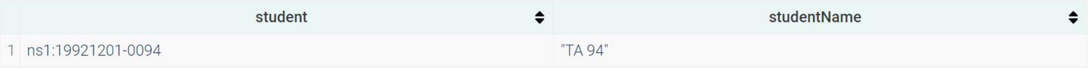
```
PREFIX ns1: <http://example.org/>

SELECT DISTINCT ?student ?studentName WHERE {
        ?student a ns1:Student ;
                        ns1:studentName ?studentName ;
                        ns1:Registrations ?register .

        ?register ns1:registrationsCourseOfferingInstanceID "I-910" .

        FILTER NOT EXISTS {
                ?student ns1:Registrations ?r2 .
                ?r2 ns1:registrationsCourseOfferingInstanceID "I-911" .
        }
}
```

Result: TA 94



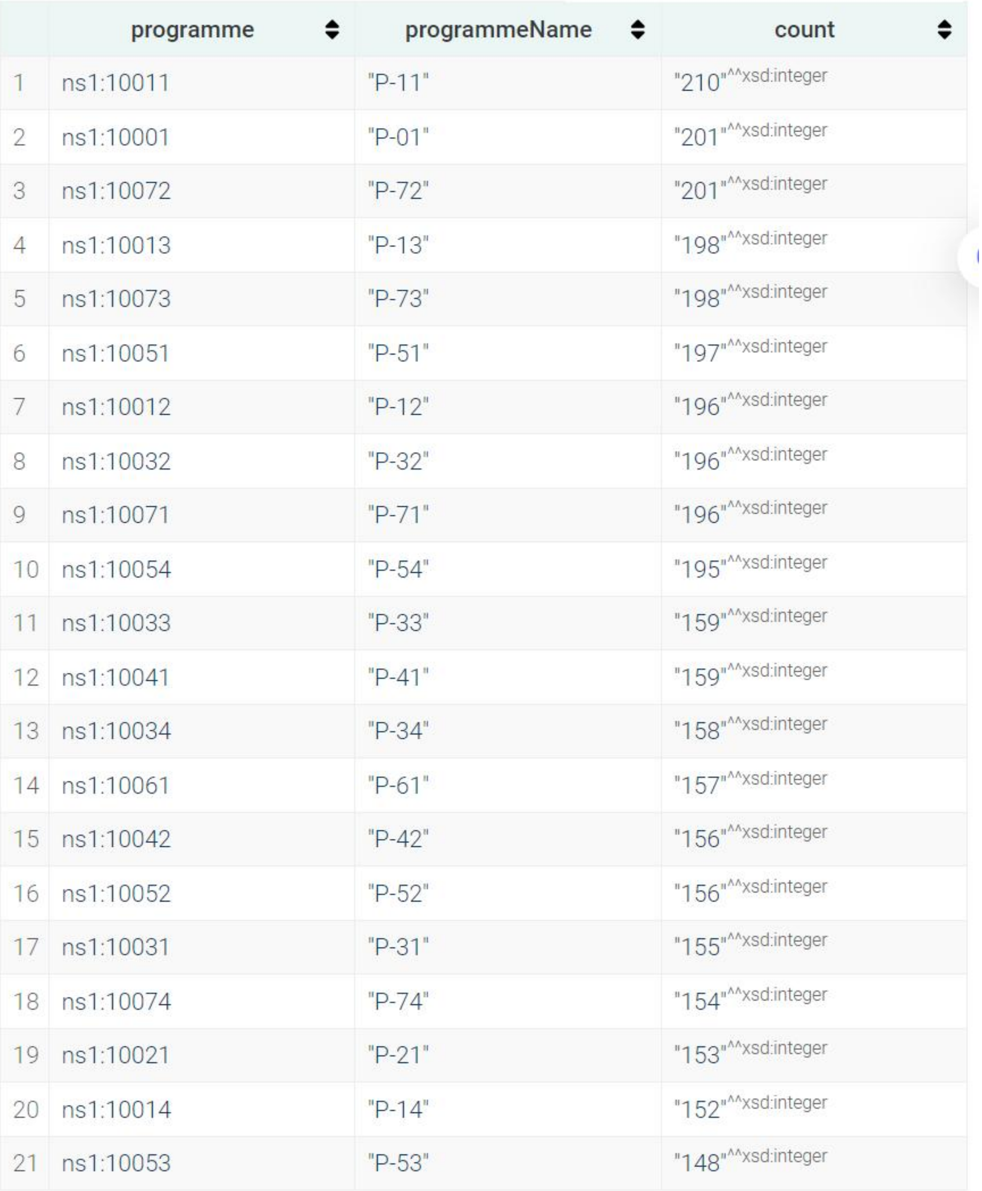
	student	studentName
1	ns1:19921201-0094	"TA 94"

5. Find all programmes along with the total number of owned courses. List the results in descending order of number of owned courses.

```
PREFIX ns1: <http://example.org/>
SELECT ?programme ?programmeName (COUNT(DISTINCT ?course) AS ?count)
WHERE {
        ?programme a ns1:Programme ;
              ns1:programmeName ?programmeName ;
              ns1:ProgrammeCourses [ns1:programmeCoursesCourseCode ?course] .

}
GROUP BY ?programme ?programmeName
ORDER BY DESC(?count)
```

Result:



	programme ⬍	programmeName ⬍	count ⬍
1	ns1:10011	"P-11"	"210"^^xsd:integer
2	ns1:10001	"P-01"	"201"^^xsd:integer
3	ns1:10072	"P-72"	"201"^^xsd:integer
4	ns1:10013	"P-13"	"198"^^xsd:integer
5	ns1:10073	"P-73"	"198"^^xsd:integer
6	ns1:10051	"P-51"	"197"^^xsd:integer
7	ns1:10012	"P-12"	"196"^^xsd:integer
8	ns1:10032	"P-32"	"196"^^xsd:integer
9	ns1:10071	"P-71"	"196"^^xsd:integer
10	ns1:10054	"P-54"	"195"^^xsd:integer
11	ns1:10033	"P-33"	"159"^^xsd:integer
12	ns1:10041	"P-41"	"159"^^xsd:integer
13	ns1:10034	"P-34"	"158"^^xsd:integer
14	ns1:10061	"P-61"	"157"^^xsd:integer
15	ns1:10042	"P-42"	"156"^^xsd:integer
16	ns1:10052	"P-52"	"156"^^xsd:integer
17	ns1:10031	"P-31"	"155"^^xsd:integer
18	ns1:10074	"P-74"	"154"^^xsd:integer
19	ns1:10021	"P-21"	"153"^^xsd:integer
20	ns1:10014	"P-14"	"152"^^xsd:integer
21	ns1:10053	"P-53"	"148"^^xsd:integer

6.Find the number of:

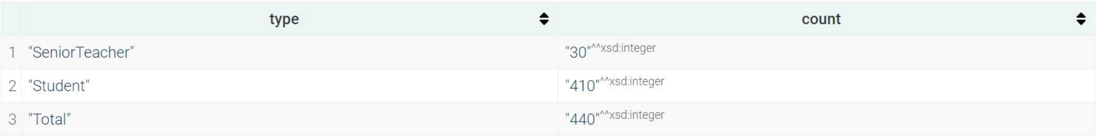
A.senior teachers

B.all people

```
PREFIX ns1: <http://example.org/>

SELECT ?type (COUNT(DISTINCT ?person) AS ?count) WHERE {
  {
    ?person a ns1:SeniorTeacher .
    BIND("SeniorTeacher" AS ?type)
  }
  UNION
  {
    ?person a ns1:Student .
    BIND("Student" AS ?type)
  }
  UNION
  {
    {
      ?person a ns1:SeniorTeacher .
    }
    UNION
    {
      ?person a ns1:Student .
    }
    BIND("Total" AS ?type)
  }
}
GROUP BY ?type
```

Result:



	type	count
1	"SeniorTeacher"	"30"^^xsd:integer
2	"Student"	"410"^^xsd:integer
3	"Total"	"440"^^xsd:integer