KNOWLEDGE TECHNOLOGIES
PROJECT 3

ΜΑΛΩΝΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
ID: 7115112200020
EMAIL: cs22200020@di.uoa.gr

Important Note
A lot of queries do not terminate even after providing tomcat with as much memory as I was able to provide given my setup. For most of these queries I am providing more than one solution and an explanation of my process of thought.

PREFIXES for all the following queries

```
PREFIX lgd:<http://linkedgeodata.org/triplify/>
PREFIX lgdgeo:<http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX lgdont:<http://linkedgeodata.org/ontology/>
PREFIX geonames:<http://www.geonames.org/ontology#>
PREFIX clc: <http://geo.linkedopendata.gr/corine/ontology#>
PREFIX gag: <http://geo.linkedopendata.gr/greekadministrativeregion/ontology#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX geor: <http://www.opengis.net/def/rule/geosparql/>
PREFIX strdf: <http://strdf.di.uoa.gr/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
PREFIX class: <https://ai.di.uoa.gr/ontology/osm#>
PREFIX uoa: <https://ai.di.uoa.gr/ontology#>
```

Exercise 1

(a) Find name and geometries of the administrative units having United Kingdom as an upper level unit.

Query

```
SELECT  ?country ?division ?polygon WHERE{
?s uoa:hasNAME_1 ?country .
?s uoa:hasTYPE_2 ?division .
?s uoa:hasCOUNTRY "United
Kingdom"^^<http://www.w3.org/2001/XMLSchema#string> .
?s geo:hasGeometry [geo:asWKT ?polygon]
}
```

Result

country	division	polygon
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-2.419980146999933 51.296355314... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Administrative County"^^<http://www.w3.org /2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.5112028179999584 52.09660150... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-2.4554310339999574 53.77399189... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"NA"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-3.039889229999858 53.779584426... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.9053770369998801 50.72491080... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.8411642089999418 50.71963534... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.7487840709998181 51.35166165... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Metropolitan Borough (City)"^^<http://www.w3.org /2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.710999976999915 53.79156115... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.081835769999941 50.818744278... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Metropolitan Borough"^^<http://www.w3.org /2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.8537026389998346 53.67251454... more
"England"^^<http://www.w3.org/2001/XMLSchema#string>	"Unitary Authority"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.6995010809998234 54.53432515... more

(b) Find geometries and names of all the administrative counties of England.

Query

```
SELECT  ?name ?polygon WHERE{
?s uoa:hasNAME_1 "England"^^<http://www.w3.org/2001/XMLSchema#string> .
?s uoa:hasTYPE_2 "Administrative
County"^^<http://www.w3.org/2001/XMLSchema#string> .
?s uoa:hasNAME_2 ?name .
?s geo:hasGeometry [geo:asWKT ?polygon]
}
```

Result

name	polygon
"Bedford"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.5112028179999584 52.09660150... more
"Cumbria"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-3.2451449999999227 54.10698600... more
"Kent"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON ((((0.8653688420001231 50.930087969... more
"North Yorkshire"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-1.3038309999998319 53.74189496... more
"Somerset"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-3.110446250999928 50.902576723... more
"Suffolk"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((1.1642007680001143 51.950546826... more
"Surrey"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.2553974659999767 51.14361970... more
"West Sussex"^^<http://www.w3.org/2001/XMLSchema#string>	"<http://www.opengis.net/def/crs/EPSG/0/4326> MULTIPOLYGON (((-0.7814238959999216 50.72642024... more

(c) Find names of all the administrative units that are neighbours of Southampton.

Query

```
SELECT ?name ?division WHERE
{
  ?s uoa:hasENGTYPE_3 ?division .
  {
  SELECT ?s ?name WHERE{
        ?s uoa:hasNAME_3 ?name .
        ?s geo:hasGeometry ?geom .
        ?geom geo:asWKT ?geom_poly .
        FILTER(?name !=
"Southampton"^^<http://www.w3.org/2001/XMLSchema#string>)
        ?s1 uoa:hasNAME_3
"Southampton"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s1 geo:hasGeometry ?south_geom .
        ?south_geom geo:asWKT ?south_geom_poly .
        FILTER(strdf:touches(?geom_poly,?south_geom_poly)) .

  }
 }
}GROUP BY ?name ?division
```

Result

name	division
"Eastleigh"^^<http://www.w3.org/2001/XMLSchema#string>	"Administrative county district"^^<http://www.w3.org/2001/XMLSchema#string>
"New Forest"^^<http://www.w3.org/2001/XMLSchema#string>	"Administrative county district"^^<http://www.w3.org/2001/XMLSchema#string>
"Test Valley"^^<http://www.w3.org/2001/XMLSchema#string>	"Administrative county district"^^<http://www.w3.org/2001/XMLSchema#string>

(d) Compute the total area of all counties of Ireland.

Query

```
SELECT (strdf:area(strdf:union(?geom_poly)) as ?area) WHERE {
  ?s geo:hasGeometry ?geom .
  ?s uoa:hasCOUNTRY "Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?geom geo:asWKT ?geom_poly .
}
```

Result

area
"9.443830482725483"^^<http://www.w3.org/2001/XMLSchema#double>

(e) Compute the minimum bounding box of the geometry of Ireland.

Query

```
SELECT (strdf:extent(?geom_poly) as ?min_bounding_box) WHERE{
        ?s uoa:hasCOUNTRY
"Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s geo:hasGeometry ?geom .
        ?geom geo:asWKT ?geom_poly .
}
```

Result

min_bounding_box
"POLYGON ((-10.662837999999965 51.419913000000065, -10.662837999999965 55.43513600000006, -5.994503... more

(f) Find administrative units that are on the border of England.

For that question I retrieve the type of divisions of United Kingdom (?division_outside) and their geometries and I also retrieve the divisions and the geometries of England (?division_inside, ?geom_eng). Then I check inside the **FILTER** which geometries that don't belong to England touch England's geometries with the strdf:touch function.

Query

attempt 1

```
SELECT ?name ?division_outside ?division_inside WHERE
{
   ?s uoa:hasNAME_1 ?name .
   ?s uoa:hasCOUNTRY "United
Kingdom"^^<http://www.w3.org/2001/XMLSchema#string> .
   ?s geo:hasGeometry ?geom .
   ?s uoa:hasTYPE_2 ?division_outside .
   ?geom geo:asWKT ?geom_poly .
   ?s1 uoa:hasNAME_1 "England"^^<http://www.w3.org/2001/XMLSchema#string> .
   ?s1 uoa:hasTYPE_2 ?division_inside .
   ?s1 geo:hasGeometry ?geom_eng .
   ?geom_eng geo:asWKT ?geom_eng_poly .
   FILTER(strdf:touches(?geom_poly,?geom_eng_poly) && ?name !=
"England"^^<http://www.w3.org/2001/XMLSchema#string>) .
} GROUP BY ?name ?division_outside ?division_inside LIMIT 2
```

attempt 2

```
SELECT ?name ?division WHERE
{
  ?s uoa:hasNAME_1 ?name .
  ?s uoa:hasTYPE_2 ?division .
  {
    SELECT ?s ?s1 ?geom_out ?geom_in WHERE
    {
      ?s geo:hasGeometry [geo:asWKT ?geom_out ] .
      ?s1 geo:hasGeometry [geo:asWKT ?geom_in ] .
      {
        SELECT ?s ?s1 WHERE
        {
          ?s uoa:hasCOUNTRY "United
Kingdom"^^<http://www.w3.org/2001/XMLSchema#string> .
```

```
        ?s uoa:hasNAME_1 ?name .
        ?s1 uoa:hasNAME_1
"England"^^<http://www.w3.org/2001/XMLSchema#string> .
        FILTER(?name != "England"^^<http://www.w3.org/2001/XMLSchema#string>)
      }
    }
  }
 }
 FILTER(geof:touches(?geom_out, ?geom_in)) .
} LIMIT 5
```

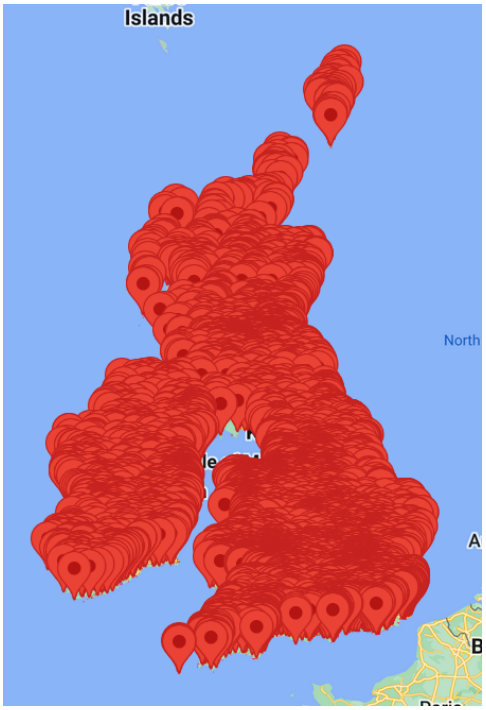
Exercise 2

(a) Find all hotels and show them on the map.

Query

```
SELECT ?hotel_geom_poly WHERE{
  ?s class:hasFclass "hotel"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s geo:hasGeometry ?hotel_geom .
  ?hotel_geom geo:asWKT ?hotel_geom_poly .
}
```

Result



(b) Find the longest five rivers and show them on a map.

Query

```
SELECT ?name (SUM(strdf:area(?poly)) as ?area) ?poly
WHERE
{
        ?s class:hasFclass ?obj .
        ?s class:hasName ?name .
        ?s geo:hasGeometry [ geo:asWKT ?poly ] .
        FILTER((?obj = "riverbank"^^xsd:string) && (REGEX(?name,
"River"^^xsd:string)))
} GROUP BY ?name ?poly ORDER BY DESC(?area) LIMIT 5
```

Result

name	area	poly
"River Dee"^^<http://www.w3.org/2001/XMLSchema#string>	"6.633268768601492E-4"^^<http://www.w3.org/2001/XMLSchema#double>	"MULTIPOLYGON (((-3.0837506 57.0550756, -3.0834921 57.0556379, -3.0833828 57.0558016, -3.0832889 57... more
"River Parrett"^^<http://www.w3.org/2001/XMLSchema#string>	"5.771880342650137E-4"^^<http://www.w3.org/2001/XMLSchema#double>	"MULTIPOLYGON (((-3.0600674 51.1743274, -3.0600532 51.1744155, -3.0598999 51.1746195, -3.0596203 51... more
"River Erne"^^<http://www.w3.org/2001/XMLSchema#string>	"5.282322015699903E-4"^^<http://www.w3.org/2001/XMLSchema#double>	"MULTIPOLYGON (((-7.6585834 54.3560094, -7.6582487 54.3561918, -7.6578575 54.356405, -7.6575167 54.... more
"River Wye"^^<http://www.w3.org/2001/XMLSchema#string>	"5.192078103900672E-4"^^<http://www.w3.org/2001/XMLSchema#double>	"MULTIPOLYGON (((-2.6872786 51.6635153, -2.687104 51.6640051, -2.6867545 51.6648031, -2.6865149 51.... more
"River Trent"^^<http://www.w3.org/2001/XMLSchema#string>	"3.9039371485500153E-4"^^<http://www.w3.org/2001/XMLSchema#double>	"MULTIPOLYGON (((-0.769208 53.4972394, -0.769101 53.4981224, -0.7689378 53.4986437, -0.768349 53.49... more

(c) Find all hospitals that are within 3 kilometers of a hotel.

Query

attempt 1

```
SELECT ?name WHERE
{
 {
  SELECT ?name (strdf:distance(strdf:envelope(?hosp_poly),
strdf:envelope(?hot_poly), uom:metre) as ?distance) WHERE
  {
        ?s class:hasFclass "hospital"^^<http://www.w3.org/2001/XMLSchema#string>
.
        ?s class:hasName ?name . FILTER(regex(?name, "Hospital")) .
        ?s geo:hasGeometry ?geo_hosp_var .
        ?geo_hosp_var geo:asWKT ?hosp_poly .
        ?s1 class:hasFclass "hotel"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s1 class:hasName ?hot_name . FILTER(regex(?hot_name, "Hotel")) .
        ?s1 geo:hasGeometry ?geo_hot_var .
        ?geo_hot_var geo:asWKT ?hot_poly .
  } LIMIT 5000
 }
 FILTER(?distance < 1000)
}
```

Result

name
"Bedford Hospital South Wing"^^<http://www.w3.org /2001/XMLSchema#string>
"Royal National Throat, Nose and Ear Hospital"^^<http://www.w3.org /2001/XMLSchema#string>
"St. Pancras Hospital"^^<http://www.w3.org /2001/XMLSchema#string>
"University College Hospital"^^<http://www.w3.org /2001/XMLSchema#string>
"Royal National ENT and Eastman Dental Hospitals"^^<http://www.w3.org/2001/XMLSchema#string> ... more
"Great Ormond Street Hospital for Children"^^<http://www.w3.org /2001/XMLSchema#string>
"Royal London Hospital for Integrated Medicine"^^<http://www.w3.org /2001/XMLSchema#string>
"National Hospital for Neurology and Neurosurgery"^^<http://www.w3.org /2001/XMLSchema#string>... more
"St Michael's Hospital"^^<http://www.w3.org /2001/XMLSchema#string>

I also made some extra attempts to eliminate the nesting of the above query

attempt 2
I retrieve the name and the geometry of the hospitals and the hotels and with the geof:distance function inside the FILTER function I keep only the ones that their distance is less than 3000 meters. The query does not terminate.

```
SELECT ?name ?hot_name WHERE
{
  ?s class:hasFclass "hospital"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s class:hasName ?name . FILTER(regex(?name, "Hospital")) .
  ?s geo:hasGeometry ?geo_hosp_var .
  ?geo_hosp_var geo:asWKT ?hosp_poly .
  ?s1 class:hasFclass "hotel"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 class:hasName ?hot_name . FILTER(regex(?hot_name, "Hotel")) .
  ?s1 geo:hasGeometry ?geo_hot_var .
  ?geo_hot_var geo:asWKT ?hot_poly .
  FILTER((geof:distance(?hosp_poly, ?hot_poly, uom:metre) < 3000))

} LIMIT 2
```

attempt 3
In the same logic I use the geof:buffer function to create a geometry of radius 3000 around each hotel geometry and with the geof:within function I find the hospitals that their geometry is inside that radius. The query does not terminate.

```
SELECT ?hosp_poly ?hot_poly WHERE
{
  ?s class:hasFclass "hospital"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s class:hasName ?name . FILTER(regex(?name, "Hospital")) .
  ?s geo:hasGeometry ?geo_hosp_var .
  ?geo_hosp_var geo:asWKT ?hosp_poly .
  ?s1 class:hasFclass "hotel"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 class:hasName ?hot_name . FILTER(regex(?hot_name, "Hotel")) .
  ?s1 geo:hasGeometry ?geo_hot_var .
  ?geo_hot_var geo:asWKT ?hot_poly .
  FILTER(geof:within(?hosp_poly, geof:buffer(?hot_poly, 3000, uom:metre)))
} LIMIT 5
```

(d) Find all the libraries that are in 2 kilometer radius of a university.

Query

attempt 1
In the nested select I retrieve the name of the libraries and the distance of that library from a university. At the outer select I keep only the ones that their distance from a university is less than 2000 meters. I use a LIMIT clause so my query terminates.

```
SELECT ?library_name WHERE
{
 {
  SELECT ?library_name  (strdf:distance(strdf:envelope(?lib_geo),
strdf:envelope(?uni_geo), uom:metre) as ?distance)
  {
        ?s class:hasFclass "library"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s class:hasName ?library_name .
        ?s geo:hasGeometry [ geo:asWKT ?lib_geo ] .
        ?s1 class:hasFclass
"university"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s1 geo:hasGeometry [ geo:asWKT ?uni_geo ] .

  } LIMIT 1000
 }
 FILTER(?distance < 2000)
}
```

Result

library_name
"Library"^^<http://www.w3.org/2001/XMLSchema#string>
"Bedford Central Library"^^<http://www.w3.org /2001/XMLSchema#string>
"Putnoe Library"^^<http://www.w3.org/2001/XMLSchema#string>

I also made some attempts to eliminate the nesting

attempt 2
I retrieve library and university information (name, geometry) and use FILTER with the distance function to find the libraries that their distance is less than 2000 meters from a library. The query did not terminate even after 24 hours of execution.

```
SELECT ?library_name WHERE
{
  ?s class:hasFclass "library"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s class:hasName ?library_name .
  ?s geo:hasGeometry [ geo:asWKT ?lib_geo ] .
  ?s1 class:hasFclass "university"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 geo:hasGeometry [ geo:asWKT ?uni_geo ] .
  FILTER(geof:distance(strdf:envelope(?lib_geo), strdf:envelope(?uni_geo),
uom:metre) < 2000) .

} LIMIT 5
```

attempt 3

Here I retrieve the library name and geometry and the university geometry and with the use of buffer I check if any university geometry is within the geometry created with the buffer function. The query does not terminate (again the query was executing for at least 24 hours).

```
SELECT ?library_name WHERE
{
  ?s class:hasFclass "library"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s class:hasName ?library_name .
  ?s geo:hasGeometry [ geo:asWKT ?lib_geo ] .
  ?s1 class:hasFclass "university"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 geo:hasGeometry [ geo:asWKT ?uni_geo ] .
  FILTER(strdf:within(?lib_geo, strdf:buffer(?uni_geo, 2000, uom:metre))) .

} LIMIT 5
```

Exercise 3

a) Which river crosses London?

Query

attempt 1
For that query I used the strdf:within inside FILTER clause instead of the strdf:crosses (in my opinion crosses function is more correct), That was the only query from all the attempts for that question that terminated.

```
SELECT ?river_name WHERE
{
  ?s uoa:hasNAME_2 "Greater
London"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s geo:hasGeometry [ geo:asWKT ?lond_geom ] .
  ?s1 class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 class:hasName ?river_name . FILTER(regex(?river_name, "River")) .
  ?s1 geo:hasGeometry [ geo:asWKT ?river_geom ] .
  FILTER(strdf:within(strdf:envelope(?river_geom), strdf:envelope(?lond_geom)))
} GROUP BY ?river_name LIMIT 10
```

Result

river_name
"River Crane"^^<http://www.w3.org/2001/XMLSchema#string>
"River Wandle"^^<http://www.w3.org/2001/XMLSchema#string>
"River Lea"^^<http://www.w3.org/2001/XMLSchema#string>

attempt 2

In the nested query I retrieve river names and geometries from england and I check if the river geometry crosses the london geometry. I used LIMIT to return fewer results for improving the execution time. I tried up to LIMIT 200000 but nothing gets returned and the query does not terminate.

```
SELECT ?river_name WHERE
{
 {
  SELECT ?river_name ((strdf:crosses(strdf:envelope(?river_geom),
strdf:envelope(?lond_geom))) AS ?crosses) WHERE
  {
        ?s uoa:hasNAME_2 "Greater
London"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s geo:hasGeometry [ geo:asWKT ?lond_geom ] .
        ?s1 class:hasFclass
"riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
        FILTER(REGEX(str(?s1), "england")) .
        ?s1 class:hasName ?river_name . FILTER(regex(?river_name, "River")) .
        ?s1 geo:hasGeometry [ geo:asWKT ?river_geom ] .

  } LIMIT 1000
 }
 FILTER(REGEX(str(?crosses), "true"))
}
```

attempt 3

I retrieve the Greater London geometries and the river geometries and I check if the river geometries cross the Greater London geometries. The query does not terminate.

```
SELECT ?river_name WHERE
{
  ?s uoa:hasNAME_2 "Greater
London"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s geo:hasGeometry [ geo:asWKT ?lond_geom ] .
  ?s1 class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 class:hasName ?river_name . FILTER(regex(?river_name, "River")) .
  ?s1 geo:hasGeometry [ geo:asWKT ?river_geom ] .
  FILTER(strdf:crosses(?river_geom, ?lond_geom))
}
```

(b) Find all counties of London that have a theatre.

Query

```
SELECT DISTINCT ?name WHERE
{
  ?s class:hasFclass "theatre"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s geo:hasGeometry [ geo:asWKT ?theatre_geom] .
  FILTER(REGEX(str(?s), "england")) .
  ?s1 uoa:hasNAME_2 "Greater
London"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 uoa:hasNAME_3 ?name .
  ?s1 geo:hasGeometry [ geo:asWKT ?lond_count_geom] .
  FILTER(strdf:within(?theatre_geom, ?lond_count_geom))

} LIMIT 10
```

Result

name
"NA"^^<http://www.w3.org/2001/XMLSchema#string>
"Barking and Dagenham"^^<http://www.w3.org /2001/XMLSchema#string>
"Barnet"^^<http://www.w3.org/2001/XMLSchema#string>
"Bexley"^^<http://www.w3.org/2001/XMLSchema#string>
"Brent"^^<http://www.w3.org/2001/XMLSchema#string>
"Camden"^^<http://www.w3.org/2001/XMLSchema#string>
"City of London"^^<http://www.w3.org/2001/XMLSchema#string>
"City of Westminster"^^<http://www.w3.org /2001/XMLSchema#string>
"Croydon"^^<http://www.w3.org/2001/XMLSchema#string>
"Ealing"^^<http://www.w3.org/2001/XMLSchema#string>

(c) Find all parks that have a museum inside.

Query

```
SELECT ?park_name ?museum_name WHERE
{
  ?s class:hasFclass "park"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s class:hasName ?park_name .
  ?s geo:hasGeometry [ geo:asWKT ?park_geom] .
  ?s1 class:hasFclass "museum"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 class:hasName ?museum_name .
  ?s1 geo:hasGeometry [ geo:asWKT ?museum_geom] .
  FILTER(strdf:within(?museum_geom, ?park_geom))
} LIMIT 10
```

Result

park_name	museum_name
"Heartlands"^^<http://www.w3.org/2001/XMLSchema#string>	"World Heritage Site Exhibitions"^^<http://www.w3.org /2001/XMLSchema#string>
"Heartlands"^^<http://www.w3.org/2001/XMLSchema#string>	"Heartlands"^^<http://www.w3.org/2001/XMLSchema#string>
"Queen's Park"^^<http://www.w3.org/2001/XMLSchema#string>	"Charnwood Museum"^^<http://www.w3.org /2001/XMLSchema#string>
"Queen's Park"^^<http://www.w3.org/2001/XMLSchema#string>	"Charnwood Museum"^^<http://www.w3.org /2001/XMLSchema#string>
"Queen's Park"^^<http://www.w3.org/2001/XMLSchema#string>	"Loughborough Carillon"^^<http://www.w3.org /2001/XMLSchema#string>
"Clapham Common"^^<http://www.w3.org/2001/XMLSchema#string>	"Clapham South Deep Level Shelter"^^<http://www.w3.org /2001/XMLSchema#string>
"West Park"^^<http://www.w3.org/2001/XMLSchema#string>	"West Park Conservatory"^^<http://www.w3.org /2001/XMLSchema#string>
"West Park"^^<http://www.w3.org/2001/XMLSchema#string>	"West Park Museum"^^<http://www.w3.org /2001/XMLSchema#string>
"Victoria Park"^^<http://www.w3.org/2001/XMLSchema#string>	"Burton Art Gallery & Museum"^^<http://www.w3.org /2001/XMLSchema#string>
"Victoria Park"^^<http://www.w3.org/2001/XMLSchema#string>	"The Museum in the Park"^^<http://www.w3.org /2001/XMLSchema#string>

(d) Find all rivers in the region of Scotland that are entirely contained in a single administrative unit of Scotland.

Query

```
SELECT ?river_name ?dist_name WHERE
{

  ?s class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
  FILTER(REGEX(str(?s), "scotland")) .
  ?s class:hasName ?river_name .
  FILTER(REGEX(str(?river_name), "River")) .
  ?s geo:hasGeometry [ geo:asWKT ?river_geom] .


  ?s1 uoa:hasNAME_1 "Scotland"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s1 geo:hasGeometry [ geo:asWKT ?dist_geom] .
  ?s1 uoa:hasTYPE_2 ?dist_type .
  ?s1 uoa:hasNAME_2 ?dist_name .


  ?s2 uoa:hasNAME_1 "Scotland"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s2 geo:hasGeometry [ geo:asWKT ?dist_geom_2] .
  ?s2 uoa:hasTYPE_2 ?dist_type_2 .
  ?s2 uoa:hasNAME_2 ?dist_name_2 .

 FILTER(strdf:within(?river_geom, ?dist_geom) && strdf:within(?river_geom,
?dist_geom_2) && ?dist_name = ?dist_name_2)

} GROUP BY ?river_name ?dist_name LIMIT 30
```

Result

river_name	dist_name
"River Don"^^<http://www.w3.org/2001/XMLSchema#string>	"Aberdeen City"^^<http://www.w3.org/2001/XMLSchema#string>
"River Dee"^^<http://www.w3.org/2001/XMLSchema#string>	"Aberdeenshire"^^<http://www.w3.org/2001/XMLSchema#string>
"River Deveron"^^<http://www.w3.org/2001/XMLSchema#string>	"Aberdeenshire"^^<http://www.w3.org/2001/XMLSchema#string>
"River Almond"^^<http://www.w3.org/2001/XMLSchema#string>	"City of Edinburgh"^^<http://www.w3.org/2001/XMLSchema#string>
"Bladnoch River"^^<http://www.w3.org/2001/XMLSchema#string>	"Dumfries and Galloway"^^<http://www.w3.org /2001/XMLSchema#string>
"River Cree"^^<http://www.w3.org/2001/XMLSchema#string>	"Dumfries and Galloway"^^<http://www.w3.org /2001/XMLSchema#string>
"River Kelvin"^^<http://www.w3.org/2001/XMLSchema#string>	"Glasgow City"^^<http://www.w3.org/2001/XMLSchema#string>
"River Clyde"^^<http://www.w3.org/2001/XMLSchema#string>	"Glasgow City"^^<http://www.w3.org/2001/XMLSchema#string>
"River Kiachnish"^^<http://www.w3.org/2001/XMLSchema#string>	"Highland"^^<http://www.w3.org/2001/XMLSchema#string>
"Glenmore River Estuary"^^<http://www.w3.org /2001/XMLSchema#string>	"Highland"^^<http://www.w3.org/2001/XMLSchema#string>
"River Glass"^^<http://www.w3.org/2001/XMLSchema#string>	"Highland"^^<http://www.w3.org/2001/XMLSchema#string>

(e) Find all rivers that cross at least two administrative units of England.
For that query I retrieve the name of the river (?river_name_1, ?river_name_2), their geometries (?river_geom_1, ?river_geom_2), the administration type and the administration types (?adm_type, ?adm_type_2) and the names of the administrations (?adm_name_1, ?adm_name_2). I keep only the rivers with the same name and the different geometries (so different parts of the same river). Each river should intersect one of the two geometries of administration that have been retrieved (last FILTER clause). The query does not terminate.

Query

```
SELECT ?river_name_1 ?river_name_2 WHERE
{
	?r1 class:hasFclass
"riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
	FILTER(REGEX(str(?r1), "england"))
	?r1 class:hasName ?river_name_1 .
	FILTER(REGEX(str(?river_name_1), "River")) .
	?r1 geo:hasGeometry [geo:asWKT ?river_geom_1] .

	?r2 class:hasFclass
"riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
	FILTER(REGEX(str(?r2), "england"))
	?r2 class:hasName ?river_name_2 .
	FILTER(REGEX(str(?river_name_2), "River")) .
	?r2 geo:hasGeometry [geo:asWKT ?river_geom_2] .

	?s1 uoa:hasNAME_1
"England"^^<http://www.w3.org/2001/XMLSchema#string> .
	?s1 uoa:hasTYPE_2 ?adm_type .
	?s1 uoa:hasNAME_2 ?adm_name_1 .
	?s1 geo:hasGeometry [ geo:asWKT ?adm_geom_1] .
	?s2 uoa:hasNAME_1
"England"^^<http://www.w3.org/2001/XMLSchema#string> .
	?s2 uoa:hasTYPE_2 ?adm_type_2 .
	?s2 uoa:hasNAME_2 ?adm_name_2 .
	?s2 geo:hasGeometry [ geo:asWKT ?adm_geom_2] .

	FILTER(?river_name_1 = ?river_name_2) .
	FILTER(?adm_name_1 != ?adm_name_2) .
	FILTER(!(strdf:equals(strdf:envelope(?river_geom_1),
strdf:envelope(?river_geom_2)))) .
```

```
        FILTER((strdf:intersects(?river_geom_1, ?adm_geom_1) ||
strdf:intersects(?river_geom_1, ?adm_geom_2)) && (strdf:intersects(?river_geom_2,
?adm_geom_1) || strdf:intersects(?river_geom_2, ?adm_geom_2)))

} LIMIT 1
```

(f) Is there any river that crosses at least two countries of the United Kingdom?

Query

```
SELECT ?river_name ?scot_river_name ?wales_river_name ?ireland_river_name WHERE
{

{

  SELECT ?river_name ?check WHERE
 {
  {
        SELECT ?river_name (strdf:intersects(strdf:envelope(?river_geom),
strdf:envelope(?eng_poly)) AS ?check) WHERE
        {
        {
        SELECT ?river_name ?river_geom WHERE
        {
       ?r class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
       ?r class:hasName ?river_name .
       ?r geo:hasGeometry [ geo:asWKT ?river_geom] .
       FILTER(REGEX(str(?river_name), "River")) .
        }
        }
        {
        SELECT ?eng_poly WHERE
        {
     ?s uoa:hasNAME_1 "England"^^<http://www.w3.org/2001/XMLSchema#string> .
     ?s geo:hasGeometry [ geo:asWKT ?eng_poly] .
        }
        }
        } LIMIT 2000
 }

 } LIMIT 2000

} # end of first nested query (england)

  # start of second nested query (scotland)
```

```
{
  SELECT ?scot_river_name ?check_2 WHERE
 {
  {
          SELECT ?scot_river_name (strdf:intersects(strdf:envelope(?scot_riv_geom),
strdf:envelope(?scotland)) AS ?check_2) WHERE
          {
          {
          SELECT ?scot_river_name ?scot_riv_geom WHERE
          {
       ?r class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
       ?r class:hasName ?scot_river_name .
       ?r geo:hasGeometry [ geo:asWKT ?scot_riv_geom] .
       FILTER(REGEX(str(?scot_river_name), "River")) .
          }
          }
          {
          SELECT ?scotland WHERE
          {
     ?s uoa:hasNAME_1 "Scotland"^^<http://www.w3.org/2001/XMLSchema#string> .
     ?s geo:hasGeometry [ geo:asWKT ?scotland] .
          }
          }
          } LIMIT 2000

  }

   } LIMIT 2000
  } # end of second nested query (scotland)


  # start of third nested query (wales)
{
  SELECT ?wales_river_name ?check_3 WHERE
 {
  {
          SELECT ?wales_river_name (strdf:intersects(strdf:envelope(?wales_riv_geom),
strdf:envelope(?wales)) AS ?check_3) WHERE
          {
          {
          SELECT ?wales_river_name ?wales_riv_geom WHERE
          {
       ?r class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
       ?r class:hasName ?wales_river_name .
       ?r geo:hasGeometry [ geo:asWKT ?wales_riv_geom] .
       FILTER(REGEX(str(?wales_river_name), "River")) .
          }
          }
```

```
          {
          SELECT ?wales WHERE
          {
     ?s uoa:hasNAME_1 "Wales"^^<http://www.w3.org/2001/XMLSchema#string> .
     ?s geo:hasGeometry [ geo:asWKT ?wales] .
          }
          }
          } LIMIT 2000

  }

   } LIMIT 2000
  } # end of third nested query (wales)

  # start of fourth nested query (ireland)
{
  SELECT ?ireland_river_name ?check_4 WHERE
 {
  {
          SELECT ?ireland_river_name (strdf:intersects(strdf:envelope(?ireland_riv_geom),
strdf:envelope(?ireland)) AS ?check_4) WHERE
          {
          {
          SELECT ?ireland_river_name ?ireland_riv_geom WHERE
          {
       ?r class:hasFclass "riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
       ?r class:hasName ?ireland_river_name .
       ?r geo:hasGeometry [ geo:asWKT ?ireland_riv_geom] .
       FILTER(REGEX(str(?ireland_river_name), "River")) .
          }
          }
          {
          SELECT ?ireland WHERE
          {
     ?s uoa:hasNAME_1 "Northern Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
     ?s geo:hasGeometry [ geo:asWKT ?ireland] .
          }
          }
          } LIMIT 2000

  }

   } LIMIT 2000
  } # end of fourth nested query (ireland)

 FILTER( (?river_name = ?scot_river_name) || (?river_name = ?wales_river_name) ||
(?wales_river_name = ?scot_river_name) || (?river_name = ?ireland_river_name) ||
```

(?ireland_river_name = ?wales_river_name) || (?ireland_river_name = ?scotland_river_name))

} LIMIT 1

Result

river_name	scot_river_name	wales_river_name	ireland_river_name
"River Weaver"^^<http://www.w3.org/2001/XMLSchema#string>	"River Ribble"^^<http://www.w3.org/2001/XMLSchema#string>	"River Weaver"^^<http://www.w3.org/2001/XMLSchema#string>	"River Ribble"^^<http://www.w3.org/2001/XMLSchema#string>

(g) Which city of Wales has most schools?

Query

attempt 1

Inside the nested query I retrieve the name of each town (?town), the type authority of that town (?authority) and it's geometry. I also retrieve the name of each school (?school_name) and it's geometry. I check all if any of the school geometries are within each towns geometry. Then at the outer SELECT I keep only those that for each town returned true inside the ?check variable (I use FILTER). I use the GROUP BY and the ORDER BY clauses to show the results in descending order.

```
SELECT DISTINCT ?town ((COUNT(?check)) AS ?number) WHERE
{

 {
  SELECT ?town ((strdf:within(strdf:envelope(?school_geom),
strdf:envelope(?town_geom))) AS ?check ) WHERE
  {


    ?s uoa:hasNAME_1 "Wales"^^<http://www.w3.org/2001/XMLSchema#string> .
    ?s uoa:hasNAME_2 ?town .
    ?s uoa:hasTYPE_2 ?authority .
    ?s geo:hasGeometry [ geo:asWKT ?town_geom] .

    ?r class:hasFclass "school"^^<http://www.w3.org/2001/XMLSchema#string> .
    ?r class:hasName ?school_name .
    FILTER(REGEX(str(?r), "wales")) .
    ?r geo:hasGeometry [ geo:asWKT ?school_geom] .
```

```
        FILTER(?town != "NA"^^<http://www.w3.org/2001/XMLSchema#string>)

  } LIMIT 10000
 }
 FILTER(REGEX(str(?check), "true"))
} GROUP BY ?town ORDER BY DESC(?number) LIMIT 5
```

Result

town	number
"Cardiff"^^<http://www.w3.org/2001/XMLSchema#string>	"149"^^<http://www.w3.org/2001/XMLSchema#integer>
"Carmarthenshire"^^<http://www.w3.org/2001/XMLSchema#string>	"148"^^<http://www.w3.org/2001/XMLSchema#integer>
"Caerphilly"^^<http://www.w3.org/2001/XMLSchema#string>	"139"^^<http://www.w3.org/2001/XMLSchema#integer>
"Conwy"^^<http://www.w3.org/2001/XMLSchema#string>	"94"^^<http://www.w3.org/2001/XMLSchema#integer>
"Flintshire"^^<http://www.w3.org/2001/XMLSchema#string>	"81"^^<http://www.w3.org/2001/XMLSchema#integer>

attempt 2

For that query at the outer SELECT clause I retrieve the name of the town, the type of authority that town is classified as (usually Unitary Authority) and the geometry of the town. At the inner query I retrieve the schools names and geometries. Then again at the outer query I use FILTER with the function strdf:within to check which returned school geometries are within the town's geometry and keep only those results. I use the GROUP BY clause to group the results by the name of each city and the COUNT clause to count how many school names got returned (?school_name) which are inside that town's geometry. The query does not terminate.

```
SELECT ?town ((COUNT(?school_name)) AS ?school_count) WHERE
{


        ?s uoa:hasNAME_1 "Wales"^^<http://www.w3.org/2001/XMLSchema#string>
.
        ?s uoa:hasNAME_2 ?town .
        ?s uoa:hasTYPE_2 ?authority .
        ?s geo:hasGeometry [ geo:asWKT ?town_geom] .

 {
 SELECT ?school_geom ?school_name WHERE
  {
        ?r class:hasFclass "school"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?r class:hasName ?school_name .
        FILTER(REGEX(str(?r), "wales")) .
        ?r geo:hasGeometry [ geo:asWKT ?school_geom] .
  }
```

```
  }
 FILTER(strdf:within(strdf:envelope(?school_geom), strdf:envelope(?town_geom)))
} GROUP BY ?town ORDER BY DESC(?school_count) LIMIT 1
```

(h) Which district of Northern Ireland has most neighbouring districts?

Note: the average execution time was two (2) hours.

Query

```
SELECT ?district_name (COUNT(?s2) AS ?neighbor_districts) WHERE
{
  ?s uoa:hasNAME_1 "Northern
Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s uoa:hasTYPE_2 "District"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s uoa:hasNAME_2 ?district_name .
  ?s geo:hasGeometry [ geo:asWKT ?dist_geom] .
  ?s2 uoa:hasNAME_1 "Northern
Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s2 uoa:hasTYPE_2 "District"^^<http://www.w3.org/2001/XMLSchema#string> .
  ?s2 uoa:hasNAME_2 ?district_name_2 .
  ?s2 geo:hasGeometry [ geo:asWKT ?dist_geom_2] .
  FILTER(strdf:touches(?dist_geom, ?dist_geom_2) && ?s != ?s2)
}  GROUP BY ?district_name ORDER BY DESC(?neighbor_districts) LIMIT 5
```

Result

district_name	neighbor_districts
"NA"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Mid Ulster"^^<http://www.w3.org/2001/XMLSchema#string>	"6"^^<http://www.w3.org/2001/XMLSchema#integer>
"Antrim and Newtownabbey"^^<http://www.w3.org /2001/XMLSchema#string>	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Armagh City, Banbridge and Craig"^^<http://www.w3.org /2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Belfast"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>

(i) Which country has largest area covered by rivers, United Kingdom or Republic of Ireland?

Query

```
SELECT ?country ?total_riv_area WHERE
{
{
SELECT ("Ireland" AS ?country) (SUM(strdf:area(?riv_ir_geom)) AS ?total_riv_area)
WHERE
{
  {
        SELECT ?riv_ir_geom ((strdf:within(?riv_ir_geom, ?ir_geom)) AS ?check)
WHERE
        {

        ?s uoa:hasCOUNTRY
"Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s geo:hasGeometry [ geo:asWKT ?ir_geom] .
        ?r_i class:hasFclass
"riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?r_i class:hasName ?ir_river .
        ?r_i geo:hasGeometry [ geo:asWKT ?riv_ir_geom] .
        FILTER(REGEX(str(?ir_river), "River")) .
        } LIMIT 10000
  }
} GROUP BY ?country
}
 UNION
{
SELECT ("GBT" AS ?country) (SUM(strdf:area(?riv_brit_geom)) AS ?total_riv_area)
WHERE
{
  {
        SELECT ?riv_brit_geom ((strdf:within(?riv_brit_geom, ?brit_geom)) AS
?check) WHERE
        {

        ?s uoa:hasCOUNTRY "United
Kingdom"^^<http://www.w3.org/2001/XMLSchema#string> .
        ?s geo:hasGeometry [ geo:asWKT ?brit_geom] .
        ?b_i class:hasFclass
"riverbank"^^<http://www.w3.org/2001/XMLSchema#string> .
```

```
        ?b_i class:hasName ?brit_river .
        ?b_i geo:hasGeometry [ geo:asWKT ?riv_brit_geom] .
        FILTER(REGEX(str(?brit_river), "River")) .
        } LIMIT 10000
  }
} GROUP BY ?country
}
} ORDER BY DESC(?total_riv_area) LIMIT 1
```

Result

Note: Because I use LIMIT clause to avoid the java.lang outofmemoryerror maybe the results do not respond to reality.

country	total_riv_area
"Ireland"	"0.17248080979348543"^^<http://www.w3.org/2001/XMLSchema#double>

(j) Find name and geometry for the neighbouring districts of the district of Northern Ireland which has the castle with the biggest area

At the first inner nested SELECT I retrieve all the districts of Northern Ireland. At the second inner nested SELECT I retrieve the biggest castle (castle name, polygon of the castle, name of the district of the castle, polygon of that district). At the outer inner SELECT I check the polygons that touch the polygon of the district which the castle is within and at the outer SELECT I keep the districts name and the names of the neighboring districts.

```
SELECT ?castles_dist_name ?neigh_dist_name WHERE
{

{
 SELECT ?castles_dist_name ?neigh_dist_name
((strdf:touches(strdf:envelope(?neigh_dist_poly), strdf:envelope(?castles_dist_poly)))
AS ?check ) WHERE
 {

  {
  SELECT ?neigh_dist_name ?neigh_dist_poly WHERE
  {
   ?j uoa:hasNAME_1 "Northern
Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
   ?j uoa:hasNAME_2 ?neigh_dist_name .
   ?j geo:hasGeometry [ geo:asWKT ?neigh_dist_poly] .
  } LIMIT 1000
```

```
  }

  {
   SELECT ?castle_name (strdf:area(?castle_poly) AS ?area) ?castle_poly
?castles_dist_name ?castles_dist_poly WHERE
   {
          ?i uoa:hasNAME_1 "Northern
Ireland"^^<http://www.w3.org/2001/XMLSchema#string> .
          ?i uoa:hasNAME_2 ?castles_dist_name .
          ?i geo:hasGeometry [ geo:asWKT ?castles_dist_poly] .
          ?c class:hasFclass "castle"^^<http://www.w3.org/2001/XMLSchema#string> .
          ?c class:hasName ?castle_name .
          ?c geo:hasGeometry [ geo:asWKT ?castle_poly] .
          FILTER(strdf:within(?castle_poly, ?castles_dist_poly)) .
          } ORDER BY DESC(?area) LIMIT 1
  }
  FILTER(REGEX(str(?check), "true"))
 }
}

} LIMIT 10
```