Final Project: Web of Data

Pablo Mollá Chárlez

December 23, 2024

Contents

1 Introduction and Motivation

I have a strong personal interest in economic and financial matters, as I am considering pursuing a Master's degree in **Quantitative Finance**, after finishing the current **M2 in Data Science**. Topics like taxation policies, economic freedom, and overall economic environments of different countries pique my curiosity. In this project, my goal is to leverage Knowledge Graphs and geospatial data to help answer a question related to where I might consider moving within Europe, balancing both proximity to my home country (Spain) and favorable economic conditions.

1.1 Initial Question and Adjustments

Originally, I aimed to determine which European country would be best to relocate to by considering several key economic indicators such as VAT, Income Tax Rate, Corporate Tax, and GDP.

- VAT: A **value-added tax** (VAT) is a **consumption tax** that is levied on the value added at **each stage of a product's production and distribution**. VAT is similar to, and is often compared with, a sales tax. VAT is an indirect tax, because the consumer who ultimately bears the burden of the tax is not the entity that pays it.
- Income Rate Tax: An **income tax** is a tax **imposed on individuals or entities (taxpayers)** in respect of the income or profits earned by them (commonly called taxable income).
- Corporate Tax: A **corporate tax**, also called corporation tax or company tax, is a type of **direct tax levied on the income or capital of corporations and other similar legal entities**. The tax is usually imposed at the national level, but it may also be imposed at state or local levels in some countries.
- GDP: **Gross domestic product (GDP)** is a **monetary measure of the market value of all the final goods and services produced and rendered in a specific time period by a country or countries**. GDP is often used to measure the economic health of a country or region.

The intention was to integrate these economic attributes with spatial data, specifically the distances of various European cities from Madrid, Spain. However, during data exploration on Wikidata, I encountered a significant challenge: data on income tax rates and corporate tax rates was either missing or not readily accessible through standard SPARQL queries. After spending many hours trying different queries and searching for relevant properties, I concluded that I could not extract reliable values for these two indicators directly from Wikidata.

1.2 Definitive Question

Given these data limitations, I decided to narrow the focus of my question. Instead of using all four attributes, I now primarily rely on the VAT (Value-Added Tax) rates and GDP (Gross Domestic Product) of each European country, as these metrics are more readily available. While this reduces the richness of my initial approach, it still allows me to derive insights into which countries—through their representative cities—might provide relatively more favorable economic conditions. This lead me to define the following question, which this project is meant to solve:

Which European cities are closest to Madrid and simultaneously offer relatively favorable economic conditions based on VAT and GDP?

1.3 Data and Sources

To answer this question, I need to combine geospatial data (the distance of various cities to Madrid) with economic indicators at the country level. I have a dataset of approximately 47,000 cities worldwide, each with attributes such as **latitude**, **longitude**, **country**, and **population**. Using these coordinates, I can compute distances. From Wikidata and related data sources, I retrieve the country's VAT and GDP. Ideally, lower VAT and higher GDP would signify more favorable economic conditions, assuming that lower VAT might correlate with a more consumption-friendly environment and higher GDP may signal a wealthier, more stable economy.

- **WorldCities Dataset:** The Worldcities.csv dataset contains attributes like **city** name, **latitude**, **longitude**, **country**, **ISO** codes, **administrative names**, and **population**.

	A	B	C	D	E	F	G	H	I	J	K
1	city	city_ascii	lat	lng	country	iso2	iso3	admin_name	capital	population	id
2	Tokyo	Tokyo	35.6897	139.6922	Japan	JP	JPN	T≈çky≈ç	primary	37732000	1392685764
3	Jakarta	Jakarta	-6.175	106.8275	Indonesia	ID	IDN	Jakarta	primary	33756000	1360771077
4	Delhi	Delhi	28.61	77.23	India	IN	IND	Delhi	admin	32226000	1356872604
5	Guangzhou	Guangzhou	23.13	113.26	China	CN	CHN	Guangdong	admin	26940000	1156237133
6	Mumbai	Mumbai	19.0761	72.8775	India	IN	IND	MahƒÅrƒÅshtra	admin	24973000	1356226629
7	Manila	Manila	14.5958	120.9772	Philippines	PH	PHL	Manila	primary	24922000	1608618140
8	Shanghai	Shanghai	31.2286	121.4747	China	CN	CHN	Shanghai	admin	24073000	1156073548

Figure 1: WorldCities Dataset

In order to use the original Worldcities.csv dataset with GraphDB and other RDF tools, I needed to ensure that the geographical coordinates were in the correct WKT (Well-Known Text) format. The original dataset provided latitude and longitude values as simple numeric fields. I created a brief Python script to read the CSV, combine the coordinates into the WKT "Point" format (e.g., **"Point(longitude latitude)"**), and produce a new CSV file (worldcities_improved.csv).

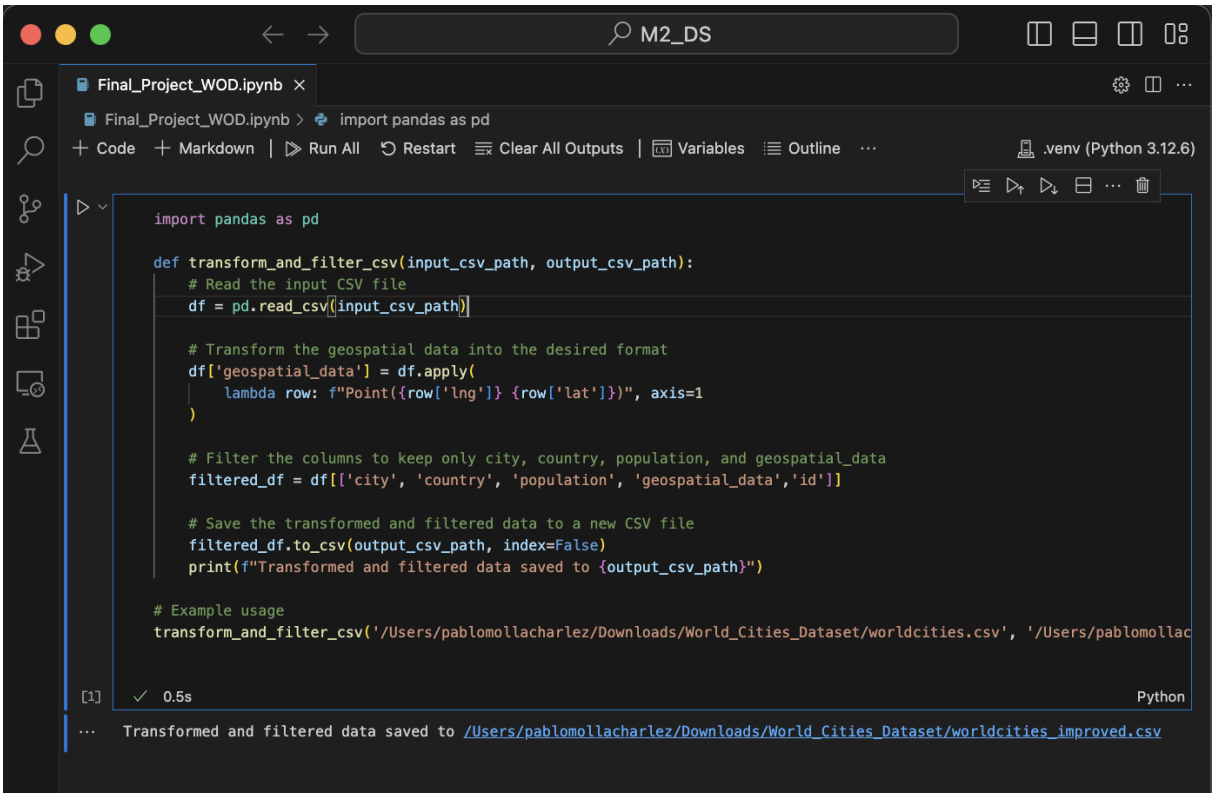


Figure 2: WorldCities.csv Modifications

This updated dataset now contains a geospatial_data column with properly formatted WKT geometries, making it ready for integration into the knowledge graph environment.

- **Economic Data (Wikidata and Wikipedia):** The data from the standard VAT rates per European country was initially found in *Tax_rates_in_Europe* and the GDP of European countries in the *List_of_sovereign_states_in_Europe_by_GDP_(nominal)*

All **datasets**, **results** as well as **video explanations** of this project can be found on the following link: Google Drive - Final Project WOD.

Country	Corporate tax	Maximum income tax rate	Standard VAT rate
Albania[2]	15%	23%[3]	20%
Andorra	10%	10%	4.5%
Armenia[4]	18%	22%	20%
Austria	25%	55%[5]	20%[6] (Reduced rates 10% + 13%)[7]
Belarus	18%	15%	20%[2]

Figure 3: Tax Rates in Europe

	Country	2024[4]	2023[5]	2022	2021[6]	2020[6]	201
1	Germany	4,683.233	4,429.840	4,256.540	4,230.172	3,780.553	3,863
2	United Kingdom	3,557.465	3,332.060	3,376.003	3,108.416	2,638.296	2,743
3	France	3,156.325	3,049.020	2,936.702	2,940.428	2,551.451	2,707
4	Italy	2,365.541	2,186.080	2,058.330	2,120.232	1,848.222	2,001
5	Russia	2,158.786	1,862.470	2,133.092	1,778.530	1,464.078	1,637
6	Spain	1,715.229	1,582.050	1,435.560	1,439.958	1,247.464	1,397

Figure 4: GDP in Europe

2 Metrics

To facilitate the answering of the question, I defined a single metric (based on previously calculated scores), which I will denote as the $final\_score$, that reflects both the economic conditions and the distance factor. In order to compute the $final\_score$, in advance I need to scale the distance and compute my own weighted score:

2.1 Scaled Distance

After retrieving the corresponding distances between all the possible european cities present in the WorldCities.csv dataset, I considered that it would be more useful to reuse such information if the value was scaled as follows:

$$scaledDistance = \frac{\text{distance - ?minDistance}}{\text{?maxDistance - ?minDistance}}$$

This normalizes the distance between the given city and Madrid into a 0-to-1 range. A value close to 0 means the city is relatively close to Madrid compared to all other considered cities. A value close to 1 means it is among the farthest cities.

2.2 Weighted Score

After having computed the **scaledDistance**, the **Weighted Score** is meant to group the information between the **scaledVAT** and **scaledGDP** as follows:

$$\text{weightedScore} = (0.2 \star ?scaledVAT) + (0.8 \star ?scaledGDP)$$

Here, ?scaledVAT) and ?scaledGDP are also normalized between 0 and 1, similarly to distance. By weighting **GDP** more heavily (0.8) than VAT (0.2), I emphasize that a strong economy (high GDP) is more important than the specific VAT rate. This choice reflects an assumption: a robust GDP might indicate a stable, opportunity-rich environment. Still, VAT is considered because tax policy does matter, just less so than overall economic output.

2.3 Final Score

Finally, the final score integrates the weighted economic conditions with the distance factor.

$$final\_score = (?weightedScore) + (0.5 \star scaledDistancee)$$

Here, the weightedScore is added in full, while the scaledDistance contributes half its value. This means:

- Economic conditions are the primary driver of the final score.
- Distance still matters significantly but does not overpower the economic dimension. A closer city gets a slight boost, while a faraway city gets a smaller boost.

To emphasize, the weights are not purely chosen and reflect the reasoning I follow:

- In the weightedScore, GDP is given more importance than VAT (4 times more, to be precise: 0.8 vs. 0.2). This indicates a belief that overall economic size and health (GDP) is a stronger determinant of favorable living conditions than just the VAT rate.
- In the *final_score*, I add the entire weightedScore plus half the scaledDistance. This means that while economic metrics define most of the "favorability," a city that is extremely close or far will tilt the final score slightly. If two cities have very similar economic scores, the one closer to Madrid will have a slightly higher final score.

Although I could not integrate all the initial economic indicators I wanted (due to data availability), I managed to construct a meaningful metric that incorporates both economic and spatial factors. By normalizing and scaling each attribute, then combining them with carefully chosen weights, I end up with a final score that can be used to identify which European cities might be both reasonably close to Spain and economically favorable to consider.

3 Ontology and RDF Implementation

In this section, I will explain the three main stages used to build and populate the RDF graph:

(1) Protégé (2) Ontotext Refine (3) GraphDB

Each stage builds on the previous one, ultimately producing a knowledge graph enriched with both local (city-level) data and external (Wikidata) economic indicators.

3.1 Protégé: Ontology Design

I began the project in Protégé, using it to define the baseline ontology structure. Protégé allows you to specify classes, properties, and prefixes (namespaces) for our RDF data in a user-friendly environment. The core ontology elements are:

- **Prefixes:** Various namespaces were declared (e.g., geo:, rdfs:, wod_final_project:) to keep the ontology well-structured and to reference well-known vocabularies like GeoSPARQL. Shown in image 5.
- **Classes:** For example, **wod_final_project:Cities** was defined as an owl:Class to represent all city entities in my dataset with label depicting the english language. Shown in image 6.
- **Data Properties:** The entities as **wod_final_project:city**, **wod_final_project:country**, **wod_final_project :population**, **wod_final_project:id**, and **wod_final_project:geospatial_data** were declared as properties by using owl:DatatypeProperty. These properties link city instances to literal values (strings, integers, etc.). Shown in image 7.

The below table summarizes the main elements from the **Final_Project_Ontology.ttl** file which allows to replicate the ontology structure that was considered:

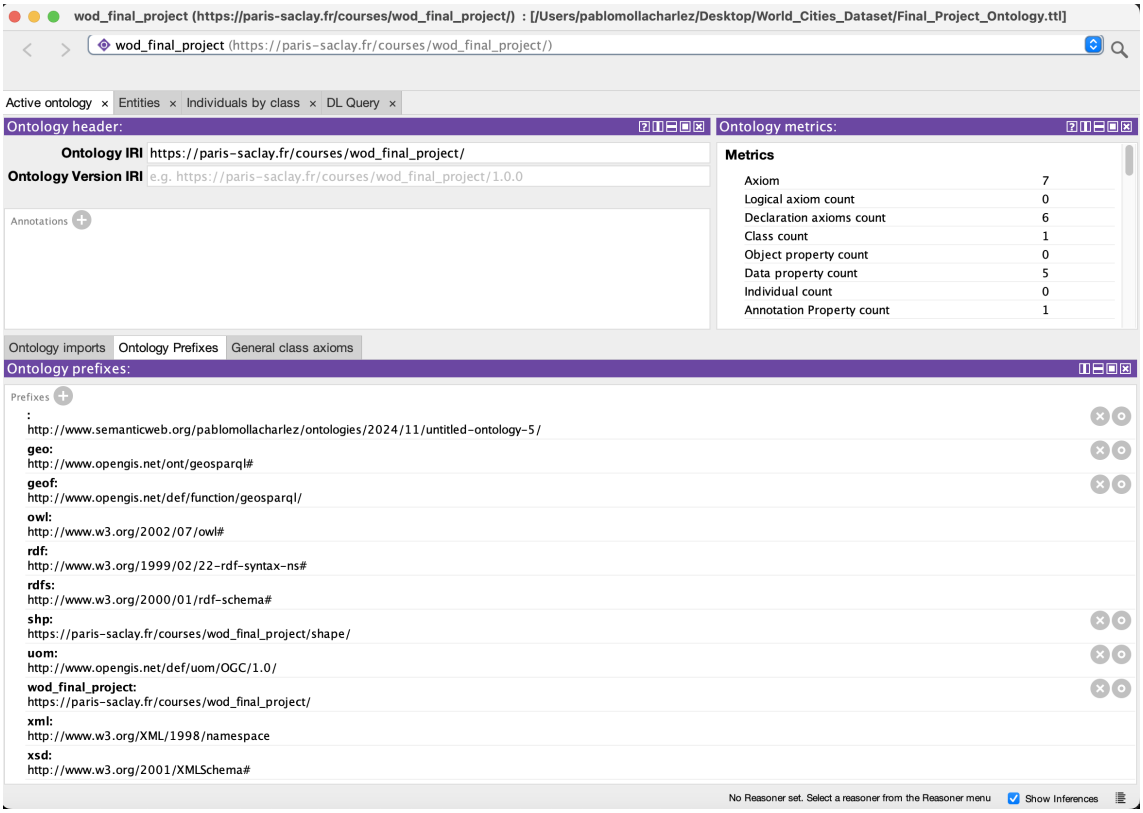


Figure 5: Ontology Prefixes

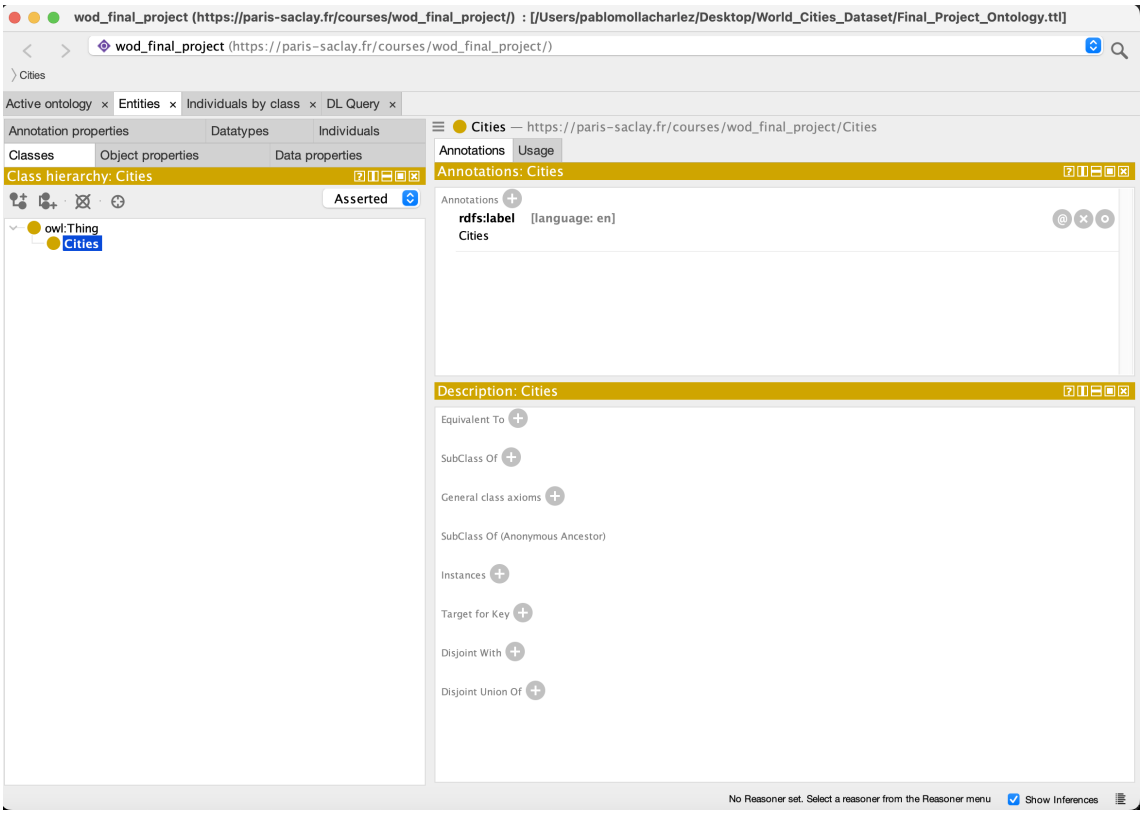


Figure 6: Ontology Classes

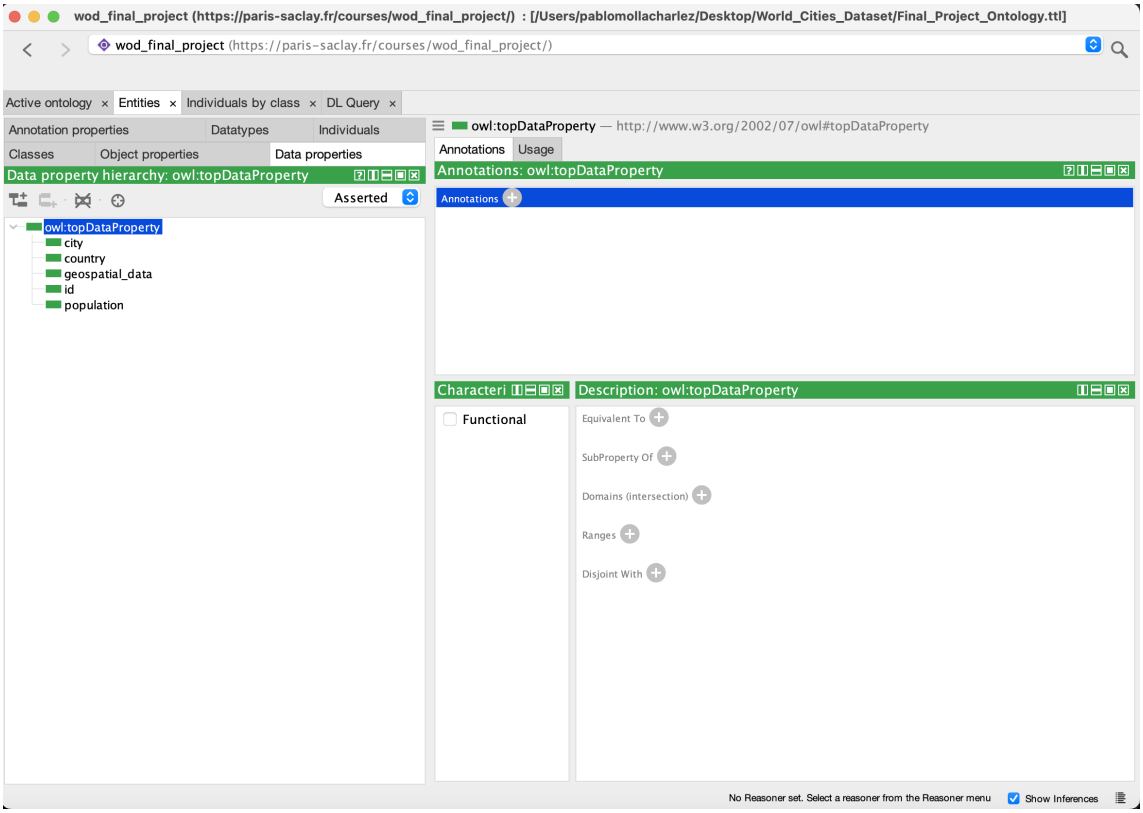


Figure 7: Ontology Properties

Ontology Overview	
Prefix	**Namespace**
:	http://www.semanticweb.org/pablomollacharlez/ontologies/2024/11/untitled-ontology-5/
geo:	http://www.opengis.net/ont/geosparql#
owl:	http://www.w3.org/2002/07/owl#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
wod_final_project:	https://paris-saclay.fr/courses/wod_final_project/
Classes	**Description**
wod_final_project:Cities	Class representing all city instances
Data Properties	**Description**
wod_final_project:city	Name of the city
wod_final_project:country	Name of the country
wod_final_project:population	Population count
wod_final_project:id	Unique city identifier
wod_final_project:geospatial_data	Geospatial WKT data

Table 1: Key Ontology Elements Defined in Protégé

3.2 Ontotext Refine: CSV Transformation and Mapping

After finalizing the ontology in Protégé, I moved on to Ontotext Refine to process and upload my CSV data. I started with a modified version of the worldcities.csv file (called worldcities_improved.csv), where I had converted latitude/longitude pairs into Well-Known Text (WKT) Point format to be compatible with GeoSPARQL. The dataset, which contained city, country, population, id, and a new geospatial_data column (with WKT coordinates), was imported into Ontotext Refine.

	A	B	C	D	E
1	city	country	population	geospatial_d	id
2	Tokyo	Japan	37732000	Point(139.69	1392685764
3	Jakarta	Indonesia	33756000	Point(106.82	1360771077
4	Delhi	India	32226000	Point(77.23 2	1356872604
5	Guangzhou	China	26940000	Point(113.26	1156237133
6	Mumbai	India	24973000	Point(72.877	1356226629
7	Manila	Philippines	24922000	Point(120.97	1608618140
8	Shanghai	China	24073000	Point(121.47	1156073548

Figure 8: worldcities_improved.csv

Using Ontotext Refine's mapping capabilities, I aligned each row of the CSV to instances of **wod_final_project:city**, assigning the appropriate properties (city, country, population, geospatial_data and id) based on the ontology. The following image 9 will describe the chosen mappings:

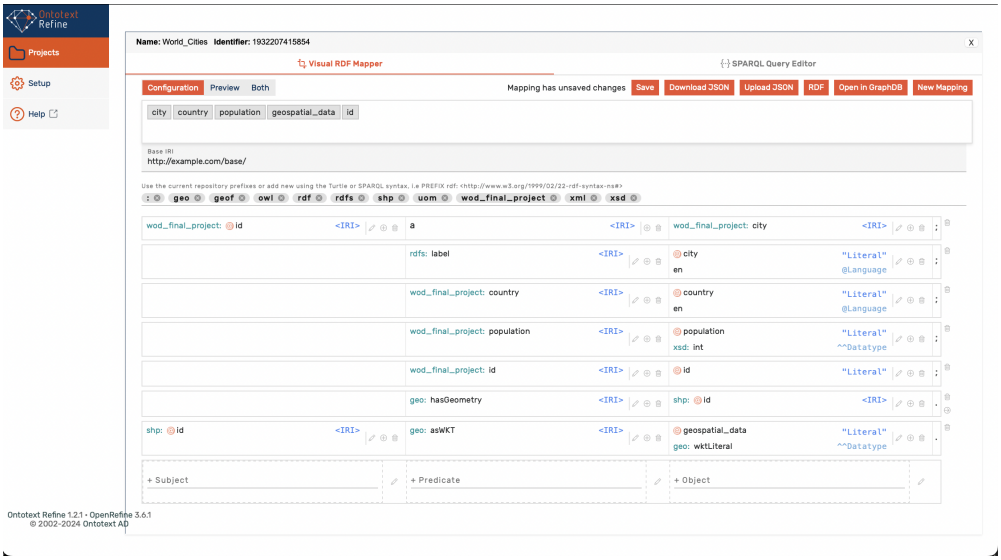


Figure 9: Enter Caption

In order to replicate without any doubt the mappings, the following link contains all the choices made: Ontology_Mappings. Finally, Ontotext Refine generated RDF triples (e.g., Final_Project_RDF_Loaded.ttl) that conform to the structure previously defined in Protégé. An example of the generated triples is:

```
wod\_final\_project:1392685764 a wod\_final\_project:city;
    rdfs:label "Tokyo"@en;
    wod\_final\_project:country "Japan"@en;
    wod\_final\_project:population "37732000.0"^^xsd:int;
    wod\_final\_project:id "1392685764";
    geo:hasGeometry shp:1392685764 .

shp:1392685764 geo:asWKT "Point(139.6922 35.6897)"^^geo:wktLiteral .

wod_final_project:1360771077 a wod_final_project:city;
    rdfs:label "Jakarta"@en;
    wod_final_project:country "Indonesia"@en;
    wod_final_project:population "33756000.0"^^xsd:int;
    wod_final_project:id "1360771077";
    geo:hasGeometry shp:1360771077 .

shp:1360771077 geo:asWKT "Point(106.8275 -6.175)"^^geo:wktLiteral .
```

Each city is typed as a **wod_final_project:city**, with its coordinates stored via **geo:hasGeometry**.

3.3 GraphDB: Uploading and Enriching the Graph

Finally, I imported the generated RDF from Ontotext Refine into GraphDB, an RDF database engine well-suited for SPARQL queries and further ontology management.

1. **Load Local RDF:** I first uploaded the **Final_Project_RDF_Loaded.ttl** into a newly created repository in GraphDB as shown in the below image 10. This step populated the graph with city instances and their respective geospatial data.

Figure 10: RDF Triples Imported into GraphDB

2. **Integrate External Data (Wikidata):** Using GraphDB's SPARQL endpoint, I retrieved additional economic indicators (VAT and GDP) from Wikidata. I filtered these indicators to focus on European countries, ensuring only relevant data (e.g., for countries in Europe) was integrated into my local knowledge graph.

Figure 11: SPARQL_Endpoint

3. **Final Queries and Scoring:** With both local (city) data and external (Wikidata) data in place, I wrote SPARQL queries to compute a weighted score (blending VAT and GDP), as well as a final score that also considers distance from a reference city (e.g., Madrid). By combining the city-level data (e.g., population, geometry) with country-level data (e.g., VAT, GDP), the final queries ranked or filtered cities based on multiple criteria.

This three-step workflow (Protégé for ontology definition, Ontotext Refine for data transformation and RDF generation, and GraphDB for data storage, integration, and querying) provided a robust pipeline for building a geospatially-aware knowledge graph.

4 GraphDB: SPARQL Implementation

The progression of SPARQL queries in this project reflects a step-by-step approach to integrating multiple data sources and transforming raw information into a meaningful final result. Initially, simpler queries focus on extracting basic data—like distances between cities or retrieving **VAT** and **GDP** values for European countries. Over time, these queries become more sophisticated, combining the extracted metrics, normalizing them for comparability, and then finally fusing all parameters into a single scoring metric. The progressive **SPARQL queries** I have tried and allowed me to reach the final query are the following:

4.1 Distances Extraction

The earliest queries simply retrieve the geographic distance between Madrid and other European cities. These queries rely on the city coordinates and geospatial functions to produce a raw distance value for each city. At this stage, the data is still unrefined and not normalized, serving primarily as a building block for later calculations.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wod\_final\_project: <https://paris-saclay.fr/courses/wod\_final\_project/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>

SELECT ?labelOtherCity ?otherCountry ?distance
WHERE {
    # Madrid's geometry
    ?madrid a wod\_final\_project:city ;
            rdfs:label "Madrid"@en ;
            wod\_final\_project:country "Spain"@en ;
            geo:hasGeometry ?madridGeom .
    ?madridGeom geo:asWKT ?madridWKT .

    # Other cities and their countries
    ?otherCity a wod\_final\_project:city ;
               rdfs:label ?labelOtherCity ;
               wod\_final\_project:country ?otherCountry ;
               geo:hasGeometry ?otherGeom .
    ?otherGeom geo:asWKT ?otherWKT .

    # Ensure cities are distinct
    FILTER(?otherCity != ?madrid)

    # Calculate distance
    BIND(geof:distance(?madridWKT, ?otherWKT, uom:metre) AS ?distance)
}
ORDER BY ?distance
LIMIT 10
```

4.2 Scaled Distance Extraction

Once the raw distances are obtained, the next step involves normalizing them. Instead of working with raw meters—which can vary widely—the query calculates a scaledDistance by mapping each distance onto a 0-to-1 range. This transformation simplifies comparisons between cities and ensures that distance can be integrated with other attributes more meaningfully later on.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wod\_final\_project: <https://paris-saclay.fr/courses/wod\_final\_project/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>

SELECT ?labelOtherCity ?otherCountry ?distance
       ((?distance - ?minDistance) / (?maxDistance - ?minDistance) AS ?scaledDistance)
WHERE {
  # Subquery to get the min and max distance from Madrid to any other city
  {
    SELECT (MIN(?dist) AS ?minDistance) (MAX(?dist) AS ?maxDistance)
    WHERE {
        ?m a wod\_final\_project:city ;
           rdfs:label "Madrid"@en ;
           wod\_final\_project:country "Spain"@en ;
           geo:hasGeometry ?mGeom .
        ?mGeom geo:asWKT ?mWKT .

        ?o a wod\_final\_project:city ;
           rdfs:label ?oLabel ;
           wod\_final\_project:country ?oCountry ;
           geo:hasGeometry ?oGeom .
        ?oGeom geo:asWKT ?oWKT .

        FILTER(?o != ?m)

        BIND(geof:distance(?mWKT, ?oWKT, uom:metre) AS ?dist)
    }
  }

  # Main query using the min and max values obtained above
  ?madrid a wod\_final\_project:city ;
          rdfs:label "Madrid"@en ;
          wod\_final\_project:country "Spain"@en ;
          geo:hasGeometry ?madridGeom .
  ?madridGeom geo:asWKT ?madridWKT .

  ?otherCity a wod\_final\_project:city ;
             rdfs:label ?labelOtherCity ;
             wod\_final\_project:country ?otherCountry ;
             geo:hasGeometry ?otherGeom .
  ?otherGeom geo:asWKT ?otherWKT .

  FILTER(?otherCity != ?madrid)

  BIND(geof:distance(?madridWKT, ?otherWKT, uom:metre) AS ?distance)
}
ORDER BY ?distance
```

4.3 Extraction of VAT and GDP

Parallel to the distance extraction, another query focuses on collecting economic indicators—specifically **VAT** and **GDP** values—at the country level from Wikidata. These values come in various ranges and units, so at this point, the query simply retrieves them as-is for each European country.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?country ?label (MAX(?vat) AS ?maxVAT) (MAX(?gdp) AS ?uniqueGDP)
WHERE {
  SERVICE <https://query.wikidata.org/sparql> {
    # Filter for countries in Europe
    ?country wdt:P31 wd:Q6256 ;        # Is a country
             wdt:P30 wd:Q46 ;          # Located in Europe
             rdfs:label ?label .       # Get the country label

    # Retrieve tax rate and GDP
    OPTIONAL { ?country wdt:P2855 ?vat }     # Value-added tax rate
    OPTIONAL { ?country wdt:P2131 ?gdp }     # GDP
    FILTER(LANG(?label) = "en")              # Ensure labels are in English
  }
}
GROUP BY ?country ?label                     # Group by country and its label
ORDER BY DESC(?maxVAT) DESC(?uniqueGDP)      # Order by VAT and then GDP
```

4.4 Weighted Score for VAT and GDP

After gathering **VAT** and **GDP**, a subsequent query normalizes these economic indicators by computing minimum and maximum values across Europe. Using these global bounds, **VAT** and **GDP** are both scaled to a 0-to-1 range. With these normalized values, a weightedScore is calculated by applying chosen weightings to **VAT** and **GDP**. This results in a single metric that encapsulates a country's economic condition as defined by these two factors.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?country ?label ?maxVAT ?uniqueGDP ?scaledVAT
       ?scaledGDP ((0.7 * ?scaledVAT) + (0.3 * ?scaledGDP) AS ?weightedScore)
WHERE {
  # Subquery to find global min/max VAT and GDP in Europe
  {
    SELECT (MIN(?vatVal) AS ?minVAT) (MAX(?vatVal) AS ?maxVATVal)
           (MIN(?gdpVal) AS ?minGDP) (MAX(?gdpVal) AS ?maxGDPVal)
    WHERE {
      SERVICE <https://query.wikidata.org/sparql> {
        ?ctr wdt:P31 wd:Q6256;
             wdt:P30 wd:Q46;
             rdfs:label ?lbl.
        FILTER(LANG(?lbl) = "en")
        OPTIONAL { ?ctr wdt:P2855 ?vatVal }
        OPTIONAL { ?ctr wdt:P2131 ?gdpVal }
      }
    }
  }

  # Subquery to get each country's VAT and GDP
  {
    SELECT ?country ?label (MAX(?vat) AS ?maxVAT) (MAX(?gdp) AS ?uniqueGDP)
    WHERE {
      SERVICE <https://query.wikidata.org/sparql> {
        ?country wdt:P31 wd:Q6256;
                 wdt:P30 wd:Q46;
                 rdfs:label ?label.
        FILTER(LANG(?label) = "en")

        OPTIONAL { ?country wdt:P2855 ?vat }  # VAT
        OPTIONAL { ?country wdt:P2131 ?gdp }  # GDP
      }
    }
    GROUP BY ?country ?label
  }

  # Ensure values are present
  FILTER(BOUND(?maxVAT) && BOUND(?uniqueGDP))

  # Scale VAT and GDP to [0,1]
  BIND((?maxVAT - ?minVAT) / (?maxVATVal - ?minVAT) AS ?scaledVAT)
  BIND((?uniqueGDP - ?minGDP) / (?maxGDPVal - ?minGDP) AS ?scaledGDP)
}
ORDER BY DESC(?weightedScore)
```

4.5 Fusion of Weighted Score with City-Level Data

As the queries progress, the focus shifts to fusing the economic indicators at the country level with city-level geographic data. In this **SPARQL query**, the previously computed weightedScore (based on **VAT** and **GDP**) is linked to the cities belonging to those countries. This ensures each city now carries not only its location but also an economic attractiveness measure derived from its country's indicators.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wod\_final\_project: <https://paris-saclay.fr/courses/wod\_final\_project/>

SELECT ?country ?villename ?label ?maxVAT ?uniqueGDP
       ((0.7 * ?scaledVAT) + (0.3 * ?scaledGDP) AS ?weightedScore)
WHERE {# Subquery to find global min/max VAT and GDP in Europe
  {SELECT (MIN(?vatVal) AS ?minVAT) (MAX(?vatVal) AS ?maxVATVal)
          (MIN(?gdpVal) AS ?minGDP) (MAX(?gdpVal) AS ?maxGDPVal)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
            ?ctr wdt:P31 wd:Q6256;
                 wdt:P30 wd:Q46;
                 rdfs:label ?lbl.
            FILTER(LANG(?lbl) = "en")
            OPTIONAL { ?ctr wdt:P2855 ?vatVal }
            OPTIONAL { ?ctr wdt:P2131 ?gdpVal }
          }}}

  # Subquery to our localdata to filter: ?countryname = ?label
  {?localville a wod\_final\_project:city ;
      rdfs:label ?villename ;
      wod\_final\_project:country ?countryname ;
      geo:hasGeometry ?shape .
  }

  # Main query to get each country's GDP and VAT
  {SELECT ?country ?label (MAX(?vat) AS ?maxVAT) (MAX(?gdp) AS ?uniqueGDP)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
        ?country wdt:P31 wd:Q6256;     # Is a country
                 wdt:P30 wd:Q46;       # Located in Europe
                 rdfs:label ?label.

        OPTIONAL { ?country wdt:P2855 ?vat }  # VAT
        OPTIONAL { ?country wdt:P2131 ?gdp }  # GDP
        FILTER(LANG(?label) = "en")
      }}
    GROUP BY ?country ?label
  }
  FILTER(BOUND(?maxVAT) && BOUND(?uniqueGDP))
  FILTER(LCASE(STR(?label))=LCASE(STR(?countryname)))
  FILTER(?uniqueGDP > 0)
  # Scale VAT and GDP to [0,1]
  BIND((?maxVAT - ?minVAT) / (?maxVATVal - ?minVAT) AS ?scaledVAT)
  BIND((?uniqueGDP - ?minGDP) / (?maxGDPVal - ?minGDP) AS ?scaledGDP)
}
ORDER BY DESC(?weightedScore)
```

4.6 Integrating the Distance Factor

Building upon the weighted economic score, the following **SPARQL query** incorporates the normalized distance measure. This step enables each city to be ranked not only by its economic conditions but also by its proximity to Madrid. The query now returns a composite view: for each city, you have the country-level economic score and the actual geographic distance, setting the stage for a final combined metric.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wod\_final\_project: <https://paris-saclay.fr/courses/wod\_final\_project/>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>

SELECT ?country ?villename ?label ?maxVAT ?uniqueGDP
       ((0.7 * ?scaledVAT) + (0.3 * ?scaledGDP) AS ?weightedScore) ?distance
WHERE {?madrid a wod\_final\_project:city ;
            rdfs:label "Madrid"@en ;
            wod\_final\_project:country "Spain"@en ;
            geo:hasGeometry ?madridGeom .
       ?madridGeom geo:asWKT ?madridWKT .
       # Other cities and their countries
       ?otherCity a wod\_final\_project:city ;
            rdfs:label ?villename ;
            wod\_final\_project:country ?countryname ;
            geo:hasGeometry ?otherGeom .
       ?otherGeom geo:asWKT ?otherWKT .

  # Subquery to find global min/max VAT and GDP in Europe
  {SELECT (MIN(?vatVal) AS ?minVAT) (MAX(?vatVal) AS ?maxVATVal)
          (MIN(?gdpVal) AS ?minGDP) (MAX(?gdpVal) AS ?maxGDPVal)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
            ?ctr wdt:P31 wd:Q6256;
                 wdt:P30 wd:Q46;
                 rdfs:label ?lbl.
            FILTER(LANG(?lbl) = "en")
            OPTIONAL { ?ctr wdt:P2855 ?vatVal }
            OPTIONAL { ?ctr wdt:P2131 ?gdpVal }}}}

  # Main query to get each country's GDP and VAT
  {SELECT ?country ?label (MAX(?vat) AS ?maxVAT) (MAX(?gdp) AS ?uniqueGDP)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
            ?country wdt:P31 wd:Q6256;     # Is a country
                     wdt:P30 wd:Q46;       # Located in Europe
                     rdfs:label ?label.
            OPTIONAL { ?country wdt:P2855 ?vat }  # VAT
            OPTIONAL { ?country wdt:P2131 ?gdp }  # GDP
            FILTER(LANG(?label) = "en")}}
    GROUP BY ?country ?label}
  FILTER(BOUND(?maxVAT) && BOUND(?uniqueGDP))
  FILTER(LCASE(STR(?label))=LCASE(STR(?countryname)))
  FILTER(?uniqueGDP > 0)
  FILTER(?otherCity != ?madrid)
  # Scale VAT and GDP to [0,1]
  BIND((?maxVAT - ?minVAT) / (?maxVATVal - ?minVAT) AS ?scaledVAT)
  BIND((?uniqueGDP - ?minGDP) / (?maxGDPVal - ?minGDP) AS ?scaledGDP)
  # Calculate distance
  BIND(geof:distance(?madridWKT, ?otherWKT, uom:metre) AS ?distance)}

ORDER BY DESC(?weightedScore)
```

4.7 The Final Combined Metric

The final query, brings together all the refined components: the scaledDistance and the weightedScore (derived from **VAT** and **GDP**). At this stage, a final formula integrates these attributes into a single *final_score* for each city. By adjusting the relative weights of the economic score and the distance, this query captures the overall suitability of a city based on both its economic landscape and its proximity to Madrid.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wod\_final\_project: <https://paris-saclay.fr/courses/wod\_final\_project/>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>

SELECT ?country ?villename ?label ?maxVAT ?uniqueGDP ?distance
       ((?distance - ?minDistance)/(?maxDistance - ?minDistance) AS ?scaledDistance)
       ((0.2 * ?scaledVAT) + (0.8 * ?scaledGDP) AS ?weightedScore)
       ((?weightedScore)+(0.5*?scaledDistance) AS ?final_score)

WHERE {# Madrid's geometry
       ?madrid a wod\_final\_project:city ;
            rdfs:label "Madrid"@en ;
            wod\_final\_project:country "Spain"@en ;
            geo:hasGeometry ?madridGeom .
        ?madridGeom geo:asWKT ?madridWKT .
        # Other cities and their countries
        ?otherCity a wod\_final\_project:city ;
                   rdfs:label ?villename ;
                   wod\_final\_project:country ?countryname ;
                   geo:hasGeometry ?otherGeom .
        ?otherGeom geo:asWKT ?otherWKT .

  # Subquery to find global min/max VAT and GDP in Europe
  {SELECT (MIN(?vatVal) AS ?minVAT) (MAX(?vatVal) AS ?maxVATVal)
          (MIN(?gdpVal) AS ?minGDP) (MAX(?gdpVal) AS ?maxGDPVal)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
            ?ctr wdt:P31 wd:Q6256;
                 wdt:P30 wd:Q46;
                 rdfs:label ?lbl.
            FILTER(LANG(?lbl) = "en")
            OPTIONAL { ?ctr wdt:P2855 ?vatVal }
            OPTIONAL { ?ctr wdt:P2131 ?gdpVal } }}}

  # Subquery to get the min and max distance from Madrid to any other city
  {SELECT (MIN(?dist) AS ?minDistance) (MAX(?dist) AS ?maxDistance)
    WHERE {?madrid a wod\_final\_project:city ;
                 rdfs:label "Madrid"@en ;
                 wod\_final\_project:country "Spain"@en ;
                 geo:hasGeometry ?madridGeom .
           ?madridGeom geo:asWKT ?madridWKT .

           ?o a wod\_final\_project:city ;
                rdfs:label ?oLabel ;
                wod\_final\_project:country ?oCountry ;
                geo:hasGeometry ?oGeom .
           ?oGeom geo:asWKT ?oWKT .
           FILTER(?o != ?m)
           BIND(geof:distance(?madridWKT, ?oWKT, uom:metre) AS ?dist) }}
```

```
  # Main query to get each country's GDP and VAT
  {SELECT ?country ?label (MAX(?vat) AS ?maxVAT) (MAX(?gdp) AS ?uniqueGDP)
    WHERE {SERVICE <https://query.wikidata.org/sparql> {
        ?country wdt:P31 wd:Q6256;     # Is a country
                 wdt:P30 wd:Q46;       # Located in Europe
                 rdfs:label ?label.
        OPTIONAL { ?country wdt:P2855 ?vat }  # VAT
        OPTIONAL { ?country wdt:P2131 ?gdp }  # GDP
        FILTER(LANG(?label) = "en")}}
    GROUP BY ?country ?label}

  FILTER(BOUND(?maxVAT) && BOUND(?uniqueGDP))
  FILTER(LCASE(STR(?label))=LCASE(STR(?countryname)))
  FILTER(?uniqueGDP > 0)
  # Ensure cities are distinct
  FILTER(?otherCity != ?madrid)
  # Scale VAT and GDP to [0,1]
  BIND((?maxVAT - ?minVAT) / (?maxVATVal - ?minVAT) AS ?scaledVAT)
  BIND((?uniqueGDP - ?minGDP) / (?maxGDPVal - ?minGDP) AS ?scaledGDP)
  # Calculate distance
  BIND(geof:distance(?madridWKT, ?otherWKT, uom:metre) AS ?distance)
}
ORDER BY DESC(?final_score)
```

5 Visualization of Results

Yasgui is a popular browser-based SPARQL editor that allows the users to easily query various SPARQL endpoints and visualize the results (e.g., as tables or maps). It operates on a publicly accessible endpoint or via a local installation that can directly communicate with the triple store of the user. In my case, I encountered difficulties downloading Yasgui and installing it locally on macOS, which is not as straightforward. I discovered that the online Yasgui website cannot access GraphDB running locally on my machine, because it needs a publicly accessible or CORS-configured endpoint in order to communicate.

Rather than spending time on network configuration or local installs, I opted to directly integrate your SPARQL query results (or CSV exports from GraphDB) into a Folium map in Python. Folium gives the ability to take locally stored data (including latitude/longitude or WKT geometry) and produce an interactive HTML map without depending on an external, browser-based SPARQL editor. This approach circumvents the networking issues and platform-specific hurdles by keeping everything on the local machine, in a controlled Python environment. Thus, Folium became a convenient replacement for Yasgui when it came to visualizing the geospatial RDF data. The Python code implemented is:

```
import pandas as pd
import folium

# 1. Reading the CSV file
df = pd.read_csv("/Users/pablomollacharlez/Desktop/World_Cities_Dataset/query-result.csv")

# 2. Defining a function to parse WKT "Point(lon lat)"
def parse_wkt_point(wkt_point):
    # Example of wkt_point: "Point(13.6411 54.5164)"
    coords = wkt_point.replace("Point(", "").replace(")", "")
    lon_str, lat_str = coords.split()

    # Return latitude, longitude for Folium
    return float(lat_str), float(lon_str)

# 3. Parsing the WKT data into separate latitude and longitude columns
df["latitude"], df["longitude"] = zip(*df["otherWKT"].apply(parse_wkt_point))
```

```python
# 4. Filtering to get subsets of data according to the user's criteria
df_first_10_germany = df[df["label"] == "Germany"].head(50)

df_next_10_excluding_germany = df[df["label"] != "Germany"].head(50) # Russia

df_next_10_excluding_germany_russia = df[(df["label"] != "Germany")
                                         & (df["label"] != "Russia")].head(50) # UK

df_next_10_excluding_germany_russia_uk = df[(df["label"] != "Germany") & (df["label"] != "Russia")
                                            & (df["label"] != "United Kingdom")].head(50) # France

df_next_10_excluding_germany_russia_uk_france = df[(df["label"] != "Germany") & (df["label"] != "Russia")
                        & (df["label"] != "United Kingdom") & (df["label"] != "France")].head(50) # Italy

df_next_10_excluding_germany_russia_uk_france_italy = df[(df["label"] != "Germany") & (df["label"] != "Russia")
                        & (df["label"] != "United Kingdom") & (df["label"] != "France")
                        & (df["label"] != "Italy")].head(50) # Spain


# 5. Creating a base Folium map centered on Madrid, Spain
# Madrid's coordinates: (lat=40.4169, lon=-3.7033)
m = folium.Map(location=[40.4169, -3.7033], zoom_start=4)

# 6. Function to add markers from a given DataFrame to the map
def add_markers(df_subset, map_obj, color):
    for _, row in df_subset.iterrows():
        folium.Marker(
            location=[row["latitude"], row["longitude"]],
            popup=f"{row['villename']} ({row['label']})",
            icon=folium.Icon(color=color)
        ).add_to(map_obj)

# 7. Adding subsets of cities with different marker colors
add_markers(df_first_10_germany, m, "darkgreen")
add_markers(df_next_10_excluding_germany, m, "green")
add_markers(df_next_10_excluding_germany_russia, m, "lightgreen")
add_markers(df_next_10_excluding_germany_russia_uk, m, "orange")
add_markers(df_next_10_excluding_germany_russia_uk_france, m, "red")
add_markers(df_next_10_excluding_germany_russia_uk_france_italy, m, "darkred")

# 7.5. Adding a pink marker for Madrid explicitly
folium.Marker(
    location=[40.4169, -3.7033],
    popup="Madrid (Spain)",
    icon=folium.Icon(color="pink")
).add_to(m)

# 8. Saving the map as an HTML file
m.save("colored_cities_map.html")
print("Map saved to colored_cities_map.html")
```

The rationale behind the chosen marker colors are as follows:

Dark Green (Germany) → Green (Russia) → Light Green (UK)
→ Orange (France) → Red (Italy) → Dark Red (Spain)

The idea is that greener tones indicate cities that obtained a higher score. As cities are graded with a lower final_score, the marker color shifts from greenish shades to orange and then to red. Ultimately, dark red markers highlight the lowest graded group of cities, even though, as it can be seen, are the closest to Spain.

Figure 12: Folium Map Visualization: WorldWide

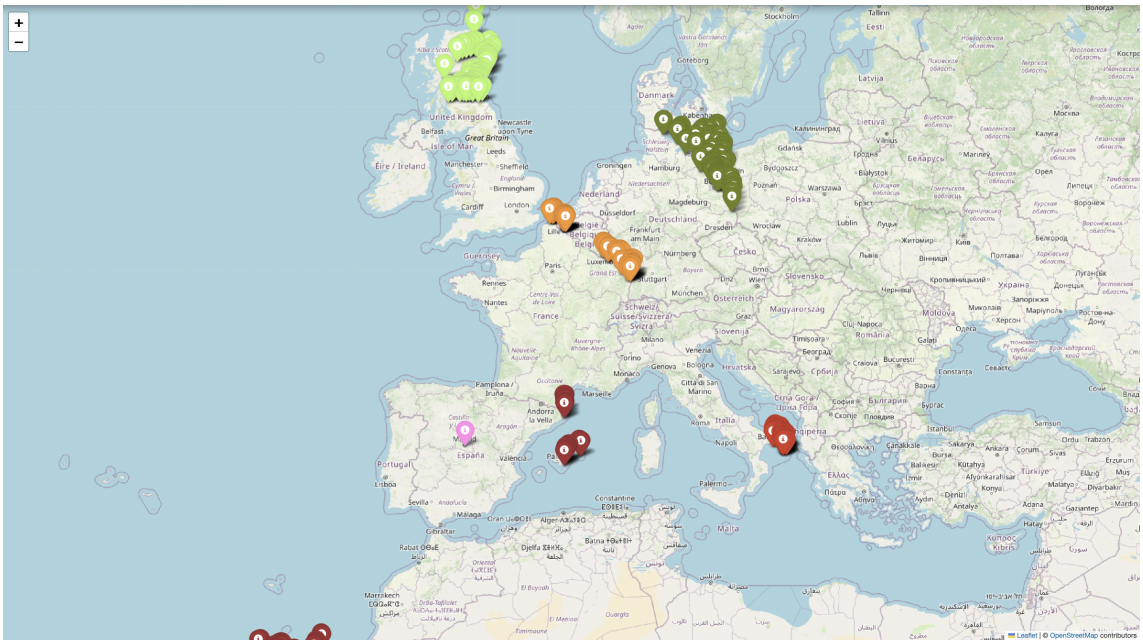
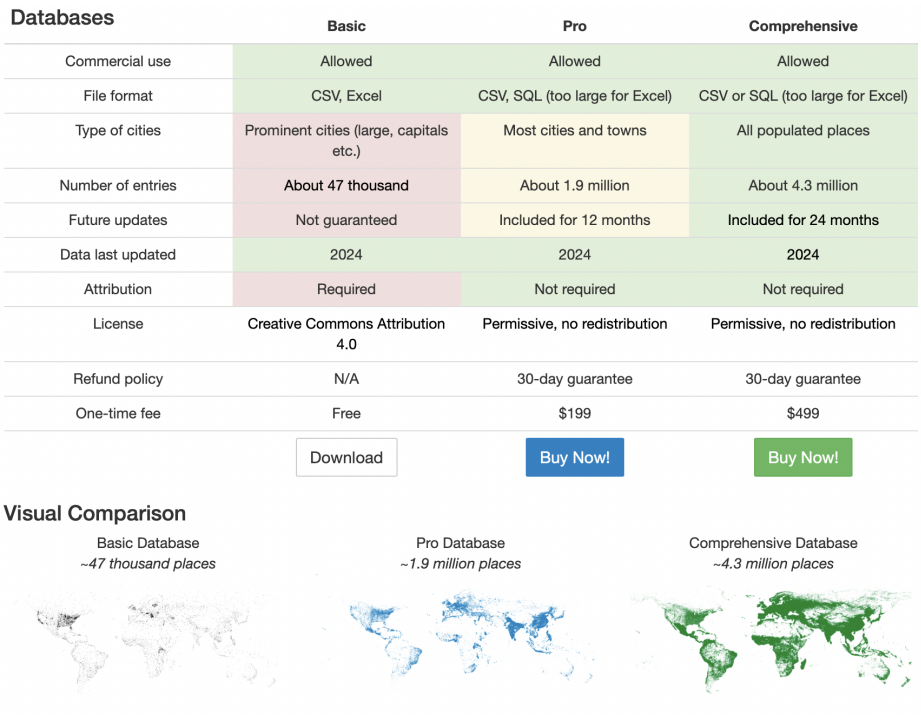


Figure 13: Folium Map Visualization: Europe

6 Results

The final results highlight an interesting pattern: when we look at the top results for the *final_score* metric, the initial highest-ranking cities are all from Germany. Examining the full dataset after downloading the complete CSV file shows the subsequent set of countries following Germany are Russia, the United Kingdom, France, Italy, and Spain. This outcome suggests that the chosen weighting—where economic indicators (VAT and GDP) dominate—favors countries with relatively strong economic conditions under the assumptions made. Since these economic values (**VAT** and **GDP**) are assigned at the national level, all cities within the same country end up with the same initial weightedScore. Thus, the final differentiation between cities from the same country comes primarily from their distance to Madrid. Cities that are closer to Madrid gain a slight edge in the *final_score*, resulting in a subtle but meaningful order among cities sharing identical national economic metrics.

One of the main reasons we see cities on the map that are far from Madrid (despite having the same country-level VAT and GDP scores) is that the original 47k-city World Cities dataset does not include every city for each country. In fact, the source website indicates that they offer **more comprehensive versions** (with up to 1.9M or even 4.3M places), containing more thorough coverage of each nation. Because our financial score is assigned at the country level, all cities in the same country end up with the same weighted score. Under ideal circumstances, the "closest" city from each country to Madrid should appear toward the top of our final ranking. However, our dataset may only include certain far-flung cities in some countries, while closer ones aren't in the 47k dataset. Hence, we will sometimes see results highlighting cities that are further away, purely because those happen to be the entries included in the publicly available dataset.



Databases

	Basic	Pro	Comprehensive
Commercial use	Allowed	Allowed	Allowed
File format	CSV, Excel	CSV, SQL (too large for Excel)	CSV or SQL (too large for Excel)
Type of cities	Prominent cities (large, capitals etc.)	Most cities and towns	All populated places
Number of entries	About 47 thousand	About 1.9 million	About 4.3 million
Future updates	Not guaranteed	Included for 12 months	Included for 24 months
Data last updated	2024	2024	2024
Attribution	Required	Not required	Not required
License	Creative Commons Attribution 4.0	Permissive, no redistribution	Permissive, no redistribution
Refund policy	N/A	30-day guarantee	30-day guarantee
One-time fee	Free	$199	$499
	Download	Buy Now!	Buy Now!

Figure 14: WorldCities Datasets

It's important to acknowledge as well, the simplifying assumption made in this analysis: all cities within a country inherit the same **VAT** tax and **GDP** values. In reality, tax policies often vary significantly at more granular levels—such as states, provinces or regions within a single country. Similarly, local economic conditions and corporate tax policies can differ dramatically from one locality to another. If we were able to incorporate this more detailed, location-specific data, each city could be scored on its own unique economic conditions, leading to a much more diverse and nuanced ranking. However, given the current data constraints, each city is evaluated using identical country-level values, and only the distance factor creates subtle variations in their final scores.

7 Conclusion

Overall, the final ranking of cities highlights an expected pattern: countries with relatively higher economic indicators (VAT and GDP) tend to dominate the top positions, causing cities from those countries to appear first in the results. Because each country's entire set of cities inherits the same economic score, the primary factor separating them within a country is their distance to Madrid, giving nearby cities a slight advantage. However, the limited coverage of the 47k-city dataset means some geographically closer cities are simply missing, occasionally yielding cities far from Madrid in the top rankings. Additionally, the assumption that all cities in a country share the same VAT and GDP simplifies real-world nuances, where regional tax policies can differ substantially. Incorporating more localized economic information and a more comprehensive dataset would enable an even more refined, realistic, and diverse ranking.