Comparing collections of Rijksmusuem and Moma using knowledge grpahs

Pawel Piwowarski
2729221@student.vu.net

Adam Rydzinski
2739038@student.vu.nl

Keira Zieboll

Teodora Taran
2737833@student.vu.nl

Group_99 on October 28 2022

1 Introduction

1.1 Defining the goal of our analysis

Our analysis is primarily focused on works of art curated by the Museum of Modern Art (MoMA) in New York and the Rijksmuseum in Amsterdam. It aims to collect and display information in a visual manner, with regards to the works of art contained within these museums. We would like to show the differences and similarities between the artworks of the two museums in categories such as, artists characteristics and the time period in which the artworks were created. We'd like to visualize these in order to make it easier to compare how the two museums differ, and to show what aspects of the artworks are most and least prevalent among the datasets we have gathered.

1.2 Identification of the Stakeholders

Given that our analysis aims to provide a comprehensive overview of the different types of artworks located in the MoMA and the Rijksmuseum, the main stakeholders are tourists and other travel providers. Tourists can use the information provided by the analysis and visualization in order to better inform their travel plans, by showing them the museum location they should visit in order to find artworks corresponding to their preferred genre or period. Furthermore, travel providers such as travel agents and tour companies can use this information to construct personalized tour packages for their customers, which will increase firm revenue through improvements in customer value proposition [1].

A secondary consequence of this analysis is that determining the most suitable museum to visit may also reduce the negative environmental effects of tourism. By recommending only the museum that fits the tourist's interests,

the tourist will be less likely to want to visit other museums that they will not be as satisfied by, thereby reducing carbon emissions from (air) transportation to the museum location.

1.3 Designing the Analysis

The analysis is a comparison of artworks and artists from both museums and which sort of periods and nationalities are the most prevalent in a given museum. We want to analyze the data by means of visualization using the pandas and folium libraries in python. The first step of our analysis would be to specify the factors that we want to compare between both museums. We then create visualizations showcasing these differences. Lastly, we draw conclusions from the visualizations and summarize our work. The visualizations that we want to provide will focus on displaying information such as the period of creation of a particular work of art or the birth date of the artist, as well their nationality. We also want to create an interactive way of displaying the artworks based on a particular year or the name of an artist.

For statistical information regarding nationalities, genders and data concerning the periods of artists, we want to utilise bar graphs or pie graphs which will help visualise which museum has the biggest impact on a given value. e.g: which museum has the most works from the year 1882.

For the interactive visual information about the artwork, we want to create a simple search mechanism were users can present their search criteria and an artwork will be displayed based on those criteria.

1.4 Motivation of Ontology Choice

We chose our ontologies based on how suitable they were for use with our domain of choice. We wanted to describe specific aspects of artworks, the artist responsible for the creation of the artwork and the location where it is currently kept. Our first ontology was the GeoSPARQL ontology[2] which was able to support geolocation data for each of our artworks. It will also be useful for categorizing artworks as spacial objects and nationalities as abstract concepts. We also used the FOAF[3] vocabulary through which we were able to create artists as persons. Lastly we used the dbo:Artwork ontology[4] for creating the sub-ontology for our all of our artworks.

1.5 Description of External Datasets

Similarly to our choice of ontologies, we found appropriate datasets that would fit our analysis domain. We narrowed down our choice of external datasets to 3 different sources: the MoMA data service[5], the Rijkmuseum data service[6] and DBPedia, which is an external SPARQL endpoint[7].

With an ever-changing collection of around 200,000 pieces from all over the world, painting, sculpture, printing, drawing, photography, architecture, design,

video, media, and performance art are among the many forms of visual expression represented in the collection of MoMA.

Each work has associated basic metadata, such as its title, artist, date of creation, medium, size, and the date of acquisition by the Museum." The artists dataset comprises 15,091 entries, which reflect all of the artists whose work is in the MoMA collection and has been cataloged in the database. Each artist's basic metadata is included, such as their name, country of origin, gender, birth year, and death year.

Likewise, the data services of the Rijksmuseum enable access to object metadata, bibliographic data, controlled vocabularies, and user contributed material. The data which is found in this dataset, similar to the one of MoMA, displays ranging types of information about artworks such as their title, artist, date of creation and category of art.

Lastly, DBpedia qualifies for querying relationships and properties of Wikipedia resources, including links to other related datasets.

2 Domain Modeling and Data Integration

2.1 Domain and Scope of the Ontology

The data domain of our project is mainly focused on artists and artworks displayed at the Museum of Modern Art (MoMA) in New York and the Rijksmuseum in Amsterdam. We hope to deliver an understanding of how information regarding the artworks exhibited in both museums compares to one another by providing a visual representation of the data in our domain. Through ontology alignment, we determined correspondences between concepts in two previously found ontologies. The DBPedia Artwork has information relevant to the vocabulary of artworks, which alongside the addition of some supplementary classes including FOAF:person, manages to link them to artists responsible for their creation. The GeoSPARQL ontology is able to link concepts related to the location of the objects.

2.2 Methodology Description

Concerning the methodology of our ontology, our primary goal was to create an ontology that is as intuitive as possible. We also wanted to allow for a rich use of imported ontologies through mapping constructs. Additionally, we wanted to infer rich information that will be crucial in the domain of our ontology. We knew from the beginning that our ontology will use a more substantial amount of properties than classes, since our data does not differ greatly by type, but rather by the properties it has. The main factor that played a big role in the creation of the ontology was the correct alignment of classes and properties with the datasets provided under external sources.

2.3 Domain Conceptualization

For clarity of presentation, each class will be underlined and each property will be written using *italics*. In our model, we have defined two main classes, Artist and Artwork. These classes encapsulate the most important concepts in our ontology. They are subclasses of Person and Spatial Object respectively. Another important class is the Museum class which will have two instances "RijksMuseum" and "Moma". Another important classes are the Country and City classes. There exists an Object property *lies_in* which has as the domain City and as the range Country. Another object property is *born_in* which has as the domain class Person and has range City. Furthermore there exist data properties *bornAt* and *hasName* where both have Person as the domain and xsd:nonNegativeInteger and rdfs:Literal respectively as ranges. People also *hasGender* which is a rdfs:Literal. A data property which has Artist as domain is *hasOccupation* it also has rdfs:Literal as range. The Artwork class has the data properties *hasDimensions*, *hasUrl*, *hasTitle*, *hasMaterial* and *hasColor* where all of these data properties have rdfs:Literal as range. A very important object property is the *made* property which has the domain Artist and the range Artwork.

2.3.1 Classes

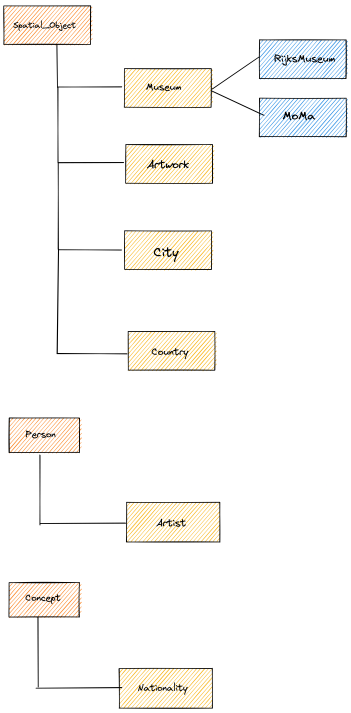


Figure 1: Class relation hierarchical graph.

2.3.2 Data properties

Subject/Domain	Name	DataType/Range
Person	bornAt	xsd:nonNegativeInteger
Person	diedAt	xsd:nonNegativeInteger
Artwork	hasColor	rdfs:Literal
Artwork	hasDimensions	rdfs:Literal
Person	hasGender	rdfs:Literal
Artwork	hasMaterial	rdfs:Literal
Person	hasName	rdfs:Literal
Artist	hasOccupation	rdfs:Literal
Artwork	hasTitle	rdfs:Literal
Artwork	hasUrl	rdfs:Literal

Figure 2: Table of data properties in our ontology.

2.3.3 Object properties

Subject/Domain	Name	Range
Person	bornIn	City
City	Lies_in	Country
Artist	made	Artwork
Person	hasNationality	Nationality

Figure 3: Table of object properties in our ontology.

2.3.4 Relation Graph

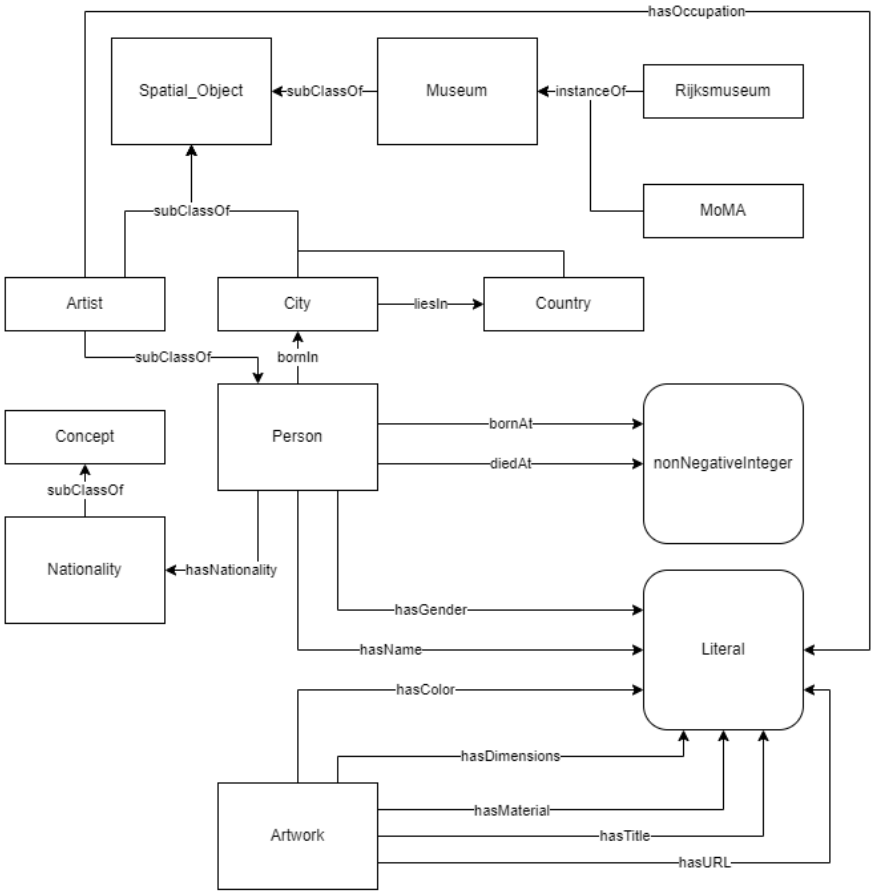


Figure 4: Relation Graph of every class, data and object property.

2.4 Ontology Reuse

Our ontology makes use of 3 existing ontologies. GeoSPARQL ontology for capturing the geographical presence of our objects. A second ontology dbo:Artwork, from which we reused the wording of concepts. The third ontology is FOAF for description of the personhood of artists.

By reusing the ontologies we have created, we added dbo:work as a subclass of geo:SpatialObject. Another subclass relationship is that of the Artist class being a subClassof Person class, which is equivalent to the Foaf:Person class. There also exist a subclass of the Concept class taken from the GeoSPARQL vocabulary, Nationality. We also added an owl:equivalenceClass relation between the class Artwork and dbo:Artwork We created a few mapping constructs using the subClassOf relationship with the GeoSPARQL ontology. These are Museum, Artwork, Country and City, which are all subclasses of the geo:SpatialObject metaclass taken from GeoSPARQL.

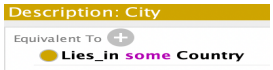
2.5 Class Restrictions Overview

The final step before populating our ontology with relevant data was to create class restrictions from which we could draw meaningful inferences. Our goal while making these restriction is to make them as intuitive as possible. We wanted to capture real world relationships and thought processes that humans take when assigning a concept or an object to a particular class.

The first restriction that we thought would be beneficial for our ontology is the necessary and sufficient restriction on the class Artist. This is Artist *made* some Artwork. Our thought process was the following: we deduced that if an entity made an Artwork it is sufficient to deduce that this entity is in fact an Artist

The second restriction that we made in our ontology is the necessary and sufficient restriction on the class Person. This is Person *bornAt* some xsd:nonNegativeInteger. Again we thought of a condition that allows us to deduce that some entity is a Person. We judged that a sufficient condition for this will be that this entity is born at some date in time.

The final restriction that we made in our ontology is the necessary and sufficient restriction on the class City. This is City *Lies_in* some Country. We thought of a scenario that would help qualify that an entity is in fact a city, in the case of our ontology only Cities have the relation *LiesIn* to some Country.



As a result of all of the previous steps, a base ontology called *base.ttl*[8] was created, this ontology will become populated with data from both of our sourced Datesets.

2.6 Integration of External Datasets and Populating our Ontology

Since the data that was made available for us was not entirely homogeneous in characteristics, we needed to prepossess the data and make the csv files that combined different data sources.

2.6.1 Rijksmuseum - Artist Ontology

The main .csv file which we decided to integrate with our ontology is located in the Rijksmuseum Github [9] . The data contained a lot of unnecessary fields. Therefore, we decided to extract only the most relevant information. The extraction script can be accessed on our Github [10] . We used both the SPARQL DBPedia endpoint alongside the standard REST API provided by the Rijksmusuem [11]with an API key . The final .csv files which were used as a base for the ontology of artists whose works are exhibited at Rijksmusem is found once again in our Github [12] . To build an ontology based on this data, we used a simple python script which opens the base ontology base.ttl and populates it with the data from the .csv file [13].

2.6.2 Rijksmuseum - Artwork Ontology

The Artwork ontology for Rijksmuseum was created in a similar manner to the artist ontology. A script to extract all of the relevant data to the .csv file was written first [14]. Based on this, the ontology that combines both artists and artworks from Rijksmuseum was created[15].

2.6.3 MoMA - Artist Ontology

We used a similar approach to create the MoMA artist ontology [16]. This time we also used a SPARQL endpoint at the level of the creation of the knowledge graph itself. We wanted to convey all of the information necessary for a homogeneous dataset. We queried information regarding the birthplaces of specific artists based on their name and occupation. The code requires around 15,000 artists to process through the endpoint. It takes around 30 minutes for the final knowledge graph with MoMA artists to be outputted.

2.6.4 Moma - Artwork Ontology

The creation of Artwork ontology for Moma required writing a simple python script which used the .csv file. This suited our needs perfectly [17].

2.7 Inference Description

The first inference that the reasoner picked up correctly inferred that all entities with the bornAt predicate are Persons. When we created the class restriction that every entity that was bornAt some date is a Person, the reasoner correctly inferred that all of the entities that we added to the graph are in fact Persons. The second inference was that Artist is an equivalence class of Person. This is simply because there are no other entities with the bornAt relation in our ontology

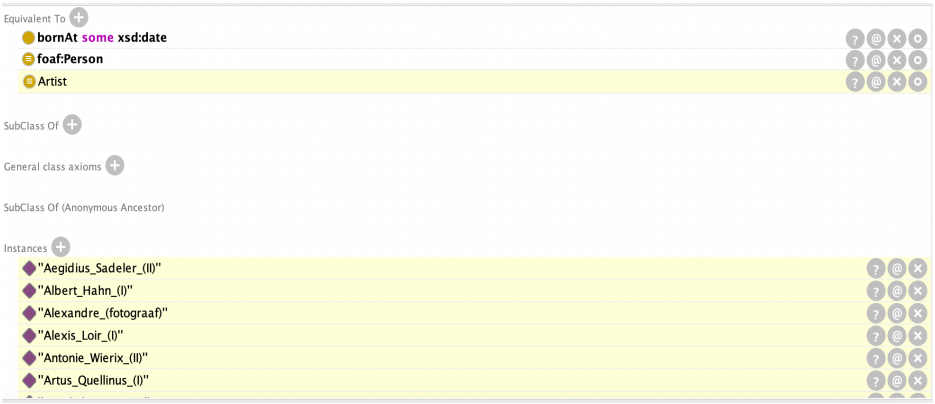


Figure 5: Reasoner inferring that all entities with "bornAt" predicate are "Persons".

The third inference that we noted was that every entity that made an artwork is an artist. This is definitely the case.

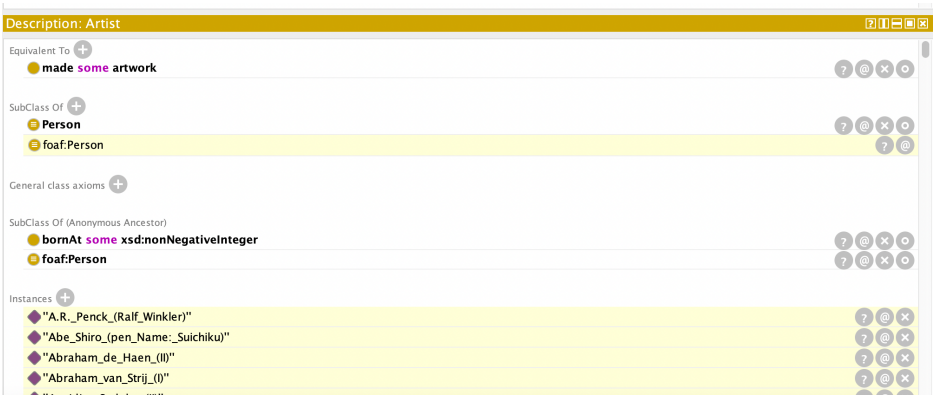


Figure 6: Reasoner inferring that all entities which "made some artwork" are artists.

The fourth inference that we noted was the inference based on the domain and range restrictions. The class Nationality is populated with instances because of the restriction *hasNationality* that has the range Nationality

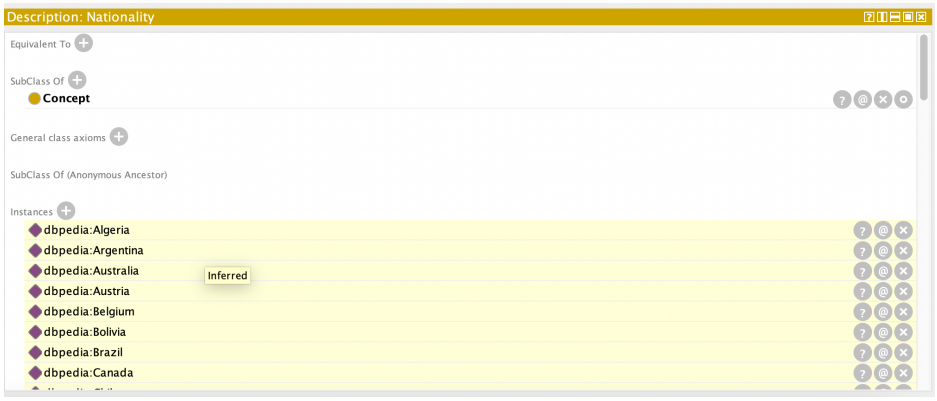


Figure 7: Class "Nationality" is populated with instances.

The last inference that we wanted to include is the inference on the class City. All of the entities that have the *liesIn* relation have to be a city (based on the domain/range characteristics).

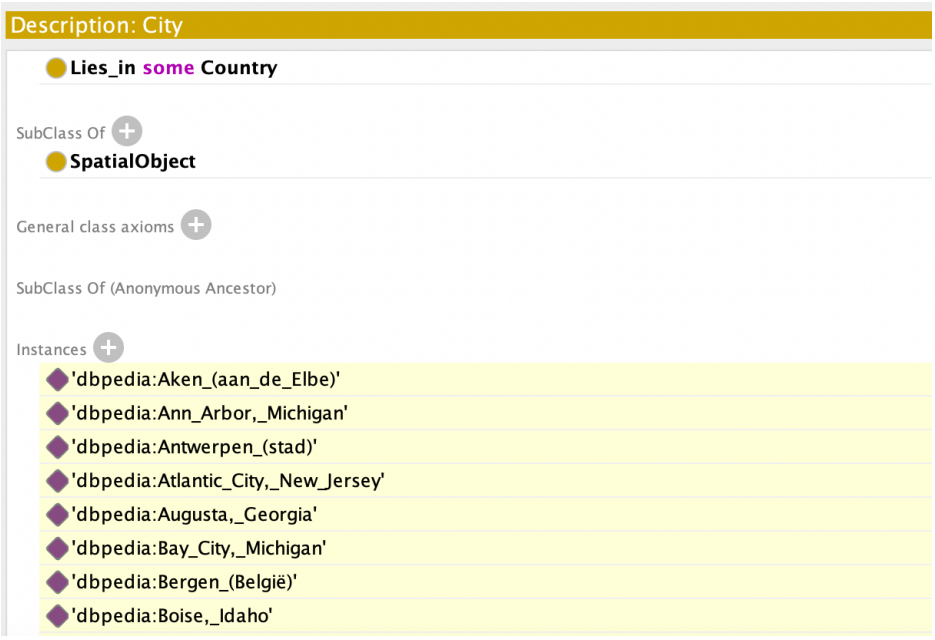


Figure 8: All entities with "liesIn" relation are in class "City".

This inferences are very crucial for our final ontology. This is why we have exported them and added them to the final .ttl file, which we have added to GraphDB. This file will be the ontology against which we'll execute our SPARQL queries.

2.8 SPARQL Queries

We made a lot of choices concerning the final shape of the characteristics that we wanted to extract from our dataset. All of the queries reflect our initial intention of displaying the different characteristics of artworks present at both museums . The queries display how different periods in time are represented in the MoMa and Rijksmusuem collections.

2.8.1 Counting all Artworks

The first query that we judged to be relevant is getting the number of works located at both museums. This query is meant to give the user the ability to see the scope of our ontology i.e the amount of works that are present in it with the classification of how many works belong to a specific museum. This query is a very simple one using one GROUP BY and COUNT statements.

```
PREFIX art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT  ?museum  (count(?x) as  ?number_of_artworks) WHERE {
?x art:locatedIn ?museum .
}

GROUP BY(?museum)
```

2.8.2 Nationality Data for Artists

Another interesting query that we needed was the query for getting the nationality of every artist that made works that are locatedIn each of the museums. The informations are split up into two actual queries and their results are to be aggregated into one result on the client side. This queries additionally use the ORDER BY statements.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>
```

```
SELECT ?nationality (count(?nationality) as ?number_of_nationalities) WHERE {

?artist art:hasNationality ?nationality .
?artist art:made ?artwork .
?artwork art:locatedIn art:RijksMuseum


}
GROUP BY ?nationality
ORDER BY DESC(?number_of_nationalities)

LIMIT 30
```

And the same query for Moma

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>

SELECT ?nationality (count(?nationality) as ?number_of_nationalities) WHERE {

?artist art:hasNationality ?nationality .
?artist art:made ?artwork .
?artwork art:locatedIn art:Moma


}
GROUP BY ?nationality
ORDER BY DESC(?number_of_nationalities)

LIMIT 30
```

2.8.3 Data concerning the birth years of the Artists

Since we wanted to compare which museum contain what periods in history, we needed appropriate SPARQL queries. This query acts as a JOIN condition from the SQL world and joins the results of the two subordinate queries on a specific value in our case it is the date of birth of an artist.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
```

```
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT * WHERE {
{
select distinct ?date (count(?date) as ?rijks_number_of_specific_dates) WHERE {

?artist rdf:type art:Artist .
?artist art:bornAt ?date .
?artist art:made ?work .
?work art:locatedIn art:RijksMuseum

}
group by (?date)
}
 {
select distinct ?date (count(?date) as ?moma_number_of_specific_dates) WHERE {

?artist rdf:type art:Artist .
?artist art:bornAt ?date .
?artist art:made ?work .
?work art:locatedIn art:Moma

}
group by (?date)
}
}

ORDER BY DESC(?date)
```

2.8.4 Gender Data (MoMa)

We wanted to showcase the disparities in how women are represented in the art world. This query showcases this on the example of the MoMA Art collection. Again we are aggregating the results of two queries into one single table.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>

SELECT * WHERE {
```

```
{
select (count(?artist) as ?Males) {
    ?artist art:hasGender "Male"
}
}
{
select (count(?artist) as ?Females) {
    ?artist art:hasGender "Female"
}
}
}
```

2.8.5 Artwork Data - based on the creator's birth year

Here our goal was to fetch the data for the artwork based on the author's birth year. We used the FILTER condition for this.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>


select distinct ?museum ?name ?url ?died_at ?city ?born_at ?title WHERE {

?artist rdf:type art:Artist .
?artist art:hasName ?name  .
?artist art:made ?artwork .
?artist art:bornIn ?city .
?artist art:diedAt ?died_at .
?artwork art:hasUrl ?url .
?artwork art:hasTitle ?title .
?artwork art:locatedIn ?museum
    filter(?url != "NaN")
    filter(?city != dbr:None && ?city != dbr:nan)
?artist art:bornAt ?born_at
  FILTER (?born_at="1812"^^xsd:nonNegativeInteger)



}
LIMIT 10
```

2.8.6 Artwork Data - based on the creator's name

A similar concept to the previous query. This time we wanted to fetch all of the artworks created by a particular artist. We used regular expressions for that.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>

select distinct ?name ?url ?died_at ?city ?born_at ?title WHERE {

?artist rdf:type art:Artist .
?artist art:hasName ?name  .
?artist art:made ?artwork .
?artist art:bornIn ?city .
?artist art:diedAt ?died_at .
?artwork art:hasUrl ?url .
?artwork art:hasTitle ?title .
?artist art:bornAt ?born_at

    FILTER regex(?name, "Rembrandt")
    filter(?url != "NaN")
    filter(?city != dbr:None && ?city != dbr:nan)
}
```

2.9 Summary Of The Findings

The readers of this work are welcomed to try and run the queries, using our notebook[18],and see the results that will be presented in a pleasant manner. We as a group have greatly developed as a team with this project and it has allowed us to deepen our knowledge and understanding of the course Knowledge Data in the year 2022.

3 Honorary title: Visualization Guru

Our goal for this part of the visualisation was to create an immersive map. For that reason we've come up with the following set of SPARQL queries. The first query is executed in the local environment of the GraphDB local instance. It is a simple query that fetches 100 cities with the highest amount of artists born in them. It is worth noting that this information is avaliable only for the artists contained within the Rijksmusem data set.

```
prefix art: <http://www.semanticweb.org/ontologies/2022/9/group_99#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbr: <http://dbpedia.org/resource/>
prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

select  ?city (count(?city) as ?number_of_cites) WHERE {
?artist art:bornIn ?city .
    FILTER(?city != dbr:None && ?city != dbr:none && ?city != dbr:nan)


}

GROUP BY(?city)

ORDER BY DESC(?number_of_cites)


LIMIT 100
```

Next, the results of this query are fed into the second query, which runs on the DPpedia's SPARQL endpoint, which fetches the longitude and latitude for each of the cities present in the first query.

```
prefix dbr: <http://dbpedia.org/resource/>
    prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

    select  ?lat ?long WHERE { ?city_from_the_first_query +  geo:lat ?lat .
    ?city_from_the_first_query  + geo:long ?long .}
```

We visualised the output of the first query using a standard bar graph.

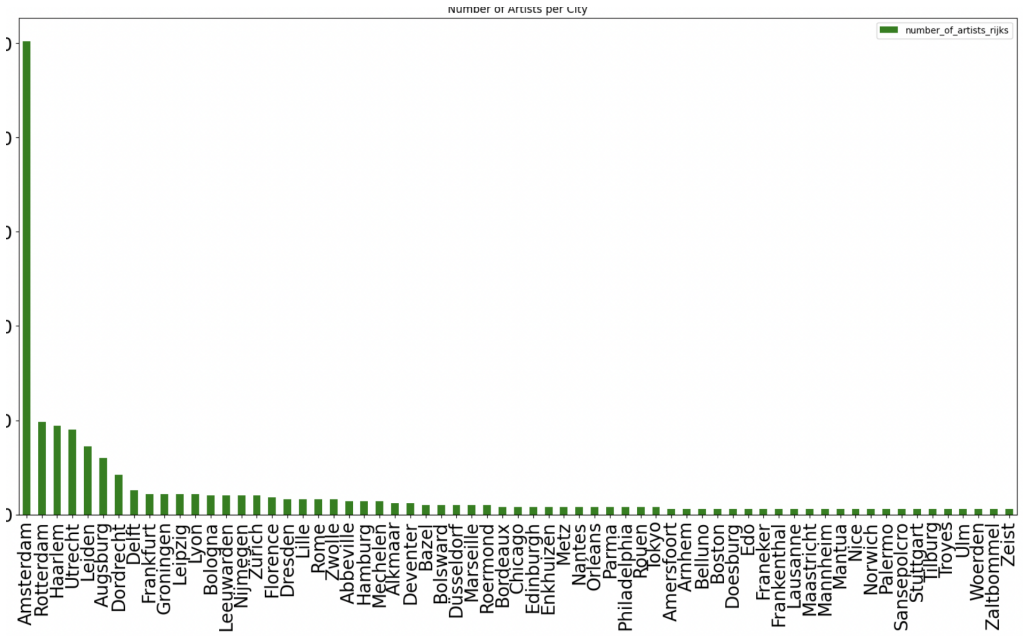


Figure 9: Bar graph visualisation of first query.

For the second query we decided that the best way to display this kind of data is with a help of a map - using the folium package for python. That is why we came up with the following interactive representation of the matter.

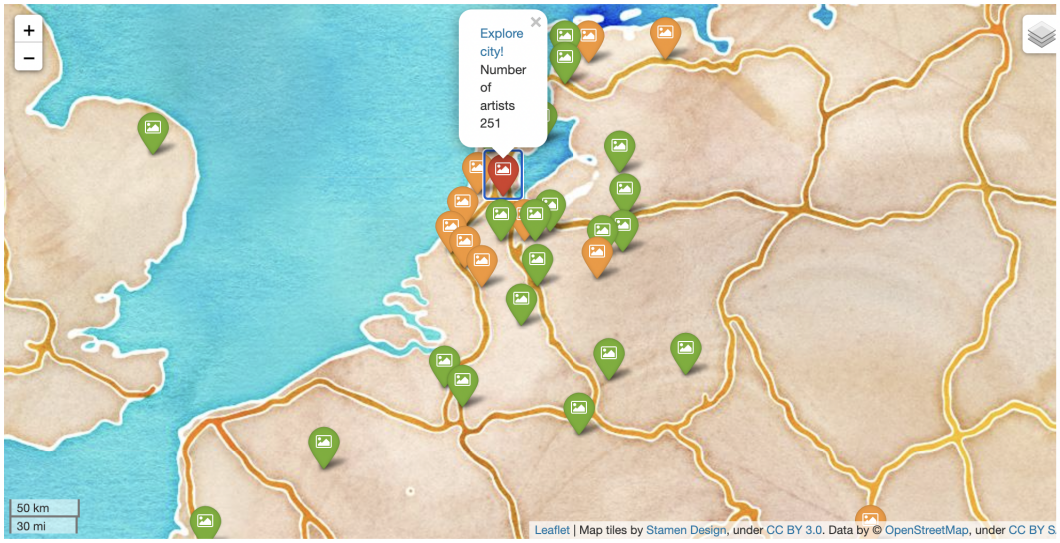


Figure 10: Example image from our map visualisation of the second query.

References

[1] M. Abraham, S. Mitchelmore, S. Collins, M. J., M. Kistulinec, S. Khodabandeh, and J. Visser, *Profiting from personalization*. Boston Consulting Group., 2017.

[2] GeoSparql, "GeoSparql Ontology." `http://www.opengis.net/ont/geosparql#`.

[3] Foaf, "Foaf Ontology." `http://xmlns.com/foaf/0.1/`.

[4] "Dbpedia Ontology on Artworks." `https://dbpedia.org/data3/Artwork.ttl`.

[5] "Museum of Modern Art Collection Database via Kaggle." `https://www.kaggle.com/datasets/momanyc/museum-collection`.

[6] Rijksmuseum, "Rijksmuseum Object Metadata Database." `https://data.rijksmuseum.nl/object-metadata/`.

[7] DBPedia, "SPARQL Endpoint." `https://dbpedia.org/sparql`.

[8] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/turtle_files/base.ttl`.

[9] `https://github.com/Rijksmuseum/rijksmuseum.github.io/releases/download/1.0.0/202001-rma-csv-collection.zip`.

[10] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/get_csv_file_artists_rijks.py`.

[11] `https://data.rijksmuseum.nl/object-metadata/api/`.

[12] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/Collections/Rijks_Collection/Final_Artist_Data_Rijks.csv`.

[13] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/rijks_artists_ontology_creation.py`.

[14] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/get_csv_file_artworks_rijks.py`.

[15] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/rijks_artworks_ontology_creation.py`.

[16] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/moma_artists_ontology_creation.py`.

[17] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/moma_artwork_ontology_creation.py`.

[18] `https://github.com/pawelpiwowarski/Knowledge_and_Data/blob/main/Group_99.ipynb`.