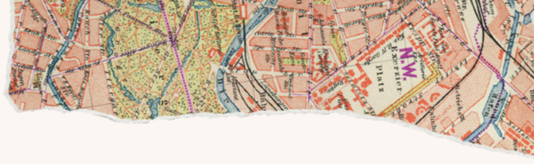
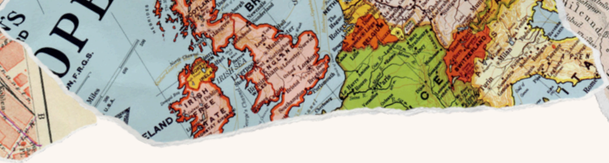
Travel & Tourism

Ontology and Knowledge Graph

Leahu-Morie Robert-Andrei
Diță Alin-Gabriel










Project Scope

Domain: Travel & Tourism

Objectives:

- Build an ontology and knowledge graph for tourist destinations

- Enable intelligent recommendations based on user preferences

- Connect destinations with attractions, activities, climate, and costs

WHAT WE HAVE DONE SO FAR...

Creating a dataset containing travel destinations

- The dataset includes the following attributes:

City, Country, Accommodation type, Accommodation cost, Transportation type, Transportation cost, Category, Cost of Living Index, Rent Index, Cost of Living Plus Rent Index Groceries Index, Restaurant Price Index, Local Purchasing Power Index, Attractions

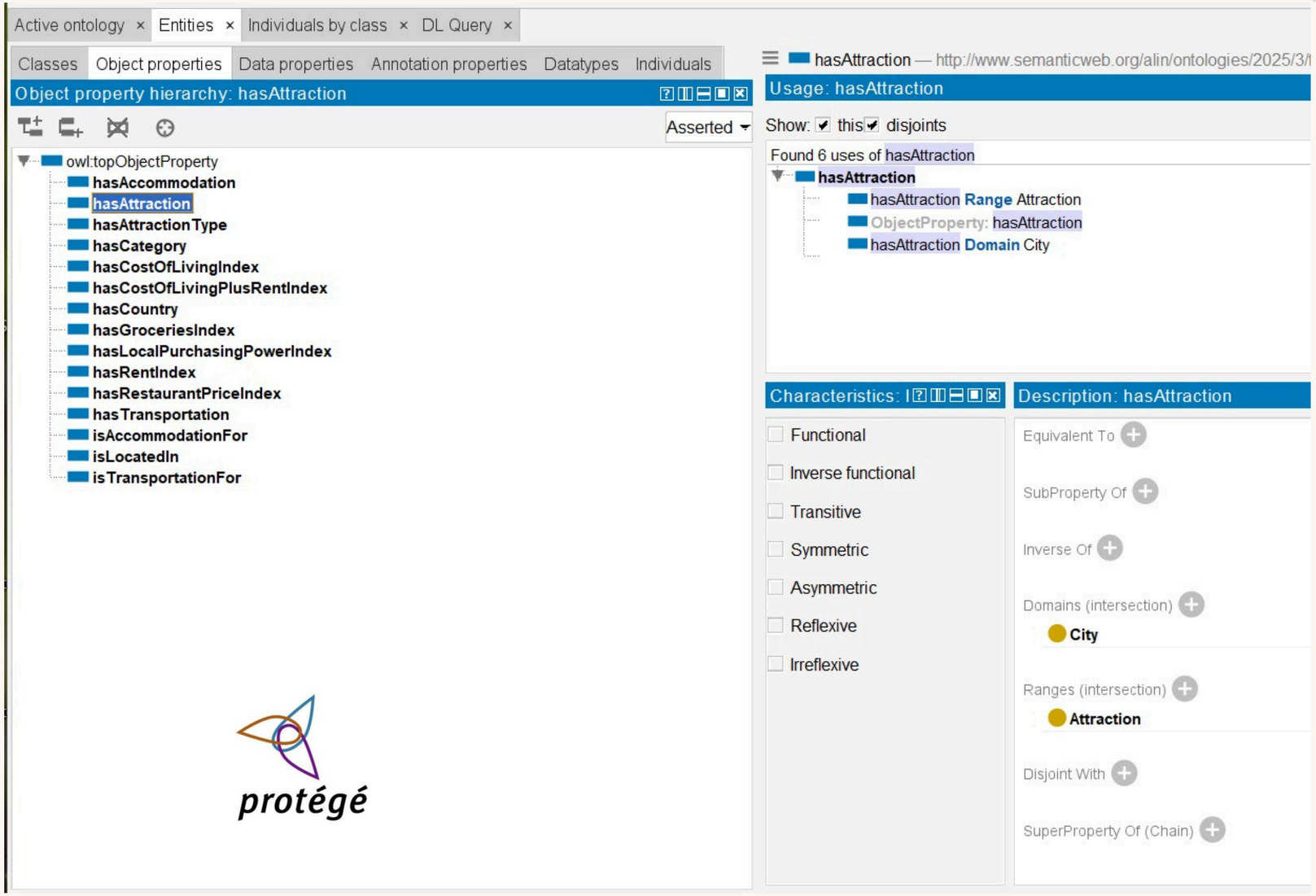
- Sources used:
  - A base dataset from Kaggle
  - Enriched via web scraping using Python + BeautifulSoup from travel websites (e.g., tripadvisor.com)

- Data scraping approach:
  - HTML parsing with BeautifulSoup
  - Cleaned and normalized into a structured format (CSV)
  - Prepared for conversion into RDF triples

WHAT WE HAVE DONE SO FAR...

- Creating an ontology (creating classes, defining relationships between classes)
- Ontology was created using Protégé

- City
- Country
- Accommodation
- Transportation
- Category
- Attraction
- CostIndex
  - CostOfLivingIndex
  - RentIndex
  - CostOfLivingPlusRentIndex
  - GroceriesIndex
  - RestaurantPriceIndex
  - LocalPurchasingPowerIndex



WHAT WE HAVE DONE SO FAR...

- Combine data from Kaggle to develop our dataset.
- Clean the dataset to remove any duplicates.

```python
import pandas as pd

# Reading a CSV file into a DataFrame
travel_details = pd.read_csv('Travel details dataset.csv')
#print(travel_details.head(10))

tourism_df = pd.read_csv('tourism_dataset.csv')
#print(tourism_df.head(10))

cost_of_living_df = pd.read_csv('Cost_of_Living_Index_by_Country_2024.csv')
# print(cost_of_living_df.head(10))

df1_partial = travel_details[["Destination", "Accommodation type", "Accommodation cost", "Transportation type", "Transportation cost"]]
df2_partial = tourism_df[["Country", "Category"]]
df3_partial = cost_of_living_df.drop(columns=["Rank"])


df_final = pd.concat([df1_partial, df2_partial, df3_partial], axis=1)
df_final.to_csv("dataset_combinat.csv", index=False)
```

```python
df_EUR = pd.read_csv("dataset_combinat.csv")
# print(df_EUR.index[:150])

df_EUR = df_EUR.dropna(subset=['City'])


print(df_EUR.shape)

print(df_EUR['City'].duplicated().sum())
df_EUR = df_EUR.drop_duplicates(subset='City', keep='first')
print(df_EUR['City'].duplicated().sum())

df_EUR.to_csv("dataset_tourism.csv", index=False)
```

WHAT WE HAVE DONE SO FAR...

- Transformed data into RDF triples to facilitate semantic integration and querying.
- Utilized rdflib for the creation and serialization of RDF graphs.
- Extracted data from a CSV file pertaining to tourism.
- Defined a custom namespace: http://example.org/tourism#.
- Created RDF triples for each city, incorporating the following attributes:
  a. Accommodation type and cost
  b. Transportation type and cost
  c. Cost of living, including rent, groceries etc.

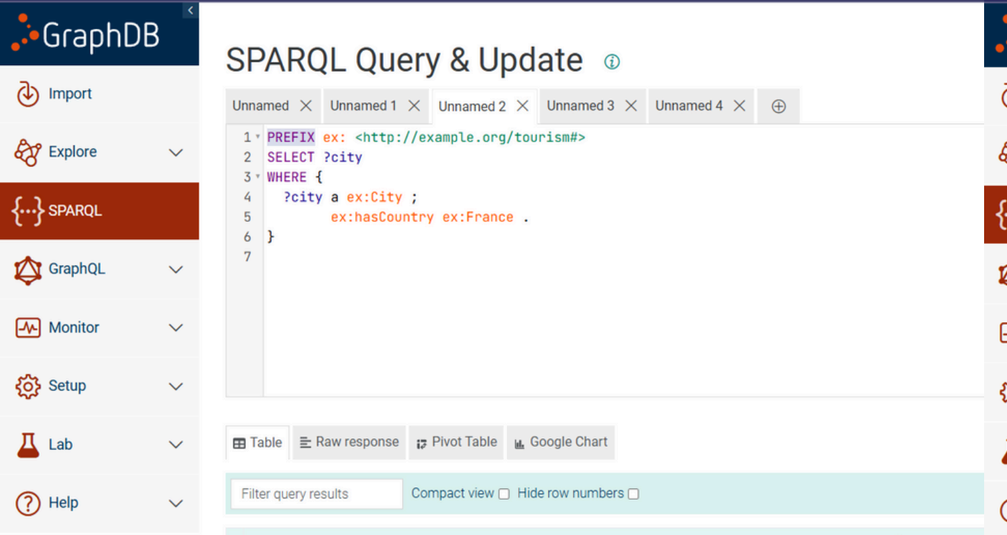
- eval() used to parse the Attractions list from the CSV
- Each attraction becomes an RDF resource of type ex:Attraction
- Added labels using rdfs:label for human-readable names
- Linked to cities via the ex:hasAttraction property

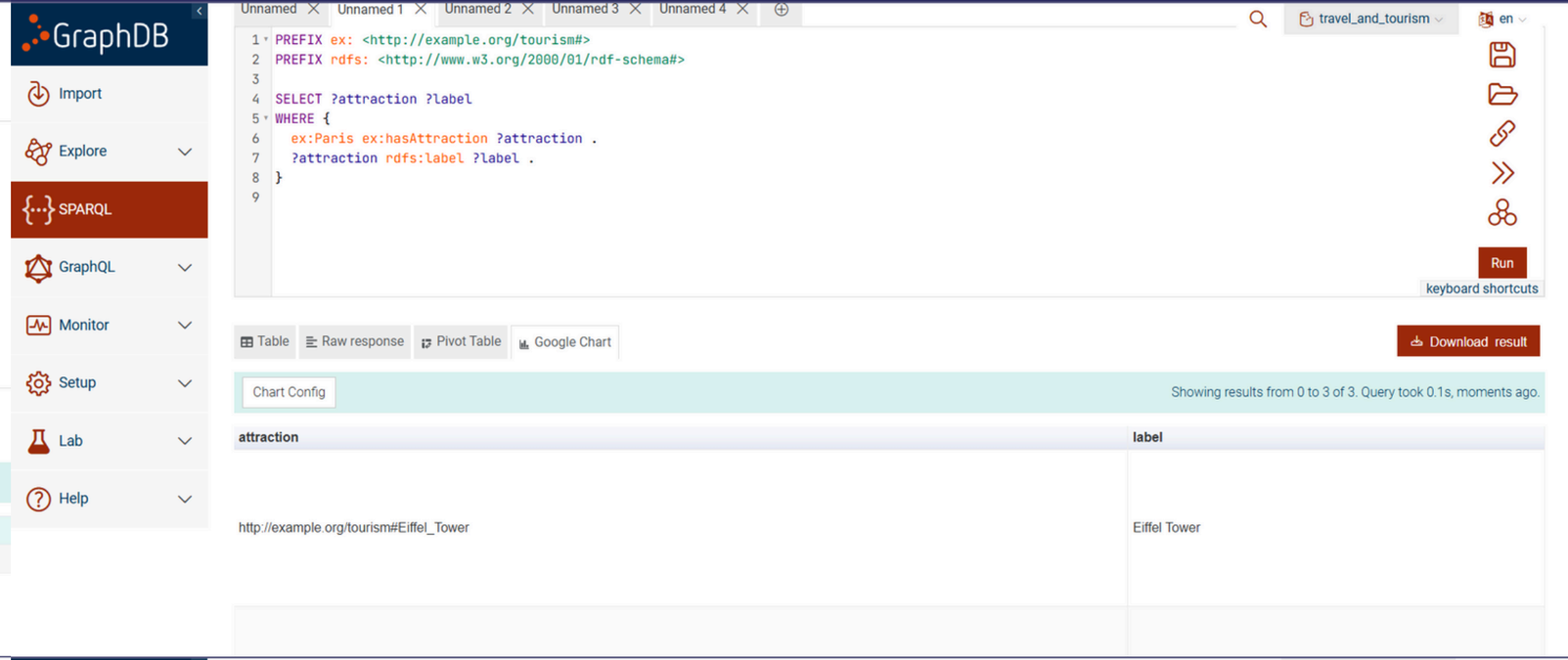
generate_graph.py

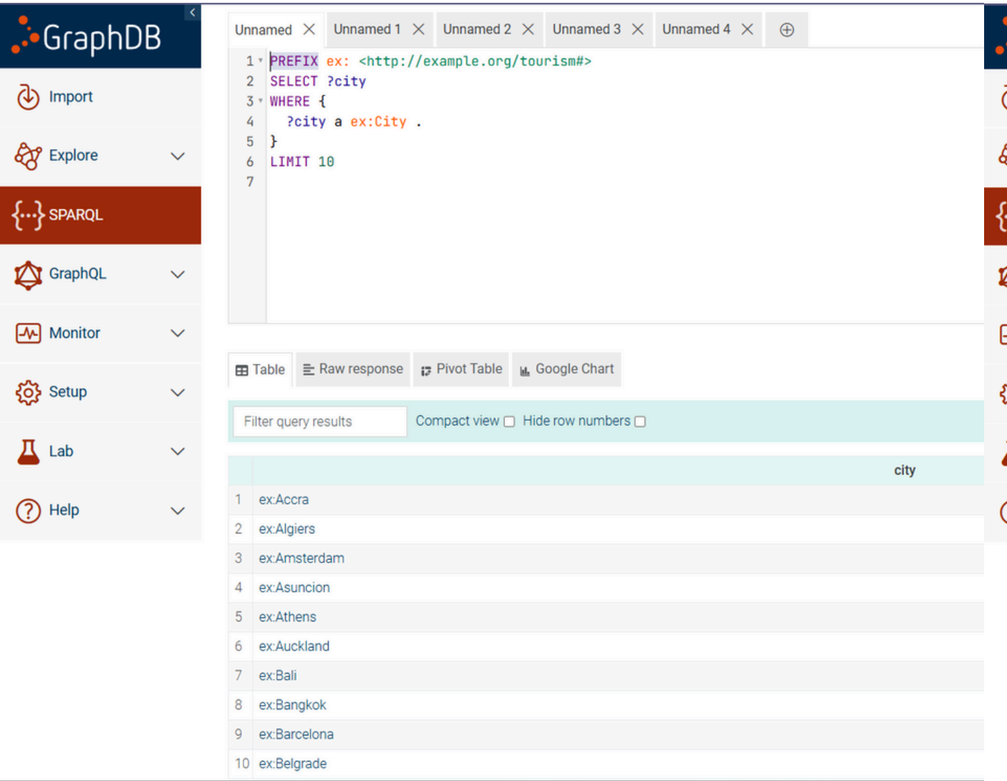
```python
import pandas as pd
from rdflib import Graph, Literal, Namespace, URIRef
from rdflib.namespace import RDF, RDFS, XSD

df = pd.read_csv("dataset_tourism.csv")

EX = Namespace("http://example.org/tourism#")
g = Graph()
g.bind( prefix: "ex", EX)

for _, row in df.iterrows():
    city_uri = URIRef(EX + row["City"].replace(" ", "_"))

    g.add((city_uri, RDF.type, EX.City))
    g.add((city_uri, EX.country, Literal(row["Country"], datatype=XSD.string)))
    g.add((city_uri, EX.accommodationType, Literal(row["Accommodation type"], datatype=XSD.string)))
    g.add((city_uri, EX.accommodationCost, Literal(row["Accommodation cost"], datatype=XSD.float)))
    g.add((city_uri, EX.transportationType, Literal(row["Transportation type"], datatype=XSD.string)))
    g.add((city_uri, EX.transportationCost, Literal(row["Transportation cost"], datatype=XSD.float)))
    g.add((city_uri, EX.category, Literal(row["Category"], datatype=XSD.string)))
    g.add((city_uri, EX.costOfLivingIndex, Literal(row["Cost of Living Index"], datatype=XSD.float)))
    g.add((city_uri, EX.rentIndex, Literal(row["Rent Index"], datatype=XSD.float)))
    g.add((city_uri, EX.costOfLivingPlusRentIndex, Literal(row["Cost of Living Plus Rent Index"], datatype=XSD.float)))
    g.add((city_uri, EX.groceriesIndex, Literal(row["Groceries Index"], datatype=XSD.float)))
    g.add((city_uri, EX.restaurantPriceIndex, Literal(row["Restaurant Price Index"], datatype=XSD.float)))
    g.add((city_uri, EX.localPurchasingPowerIndex, Literal(row["Local Purchasing Power Index"], datatype=XSD.float)))

    try:
        attractions = eval(row["Attractions"]) if isinstance(row["Attractions"], str) else []
        # print(f"{row['City']} -> {attractions}")
```

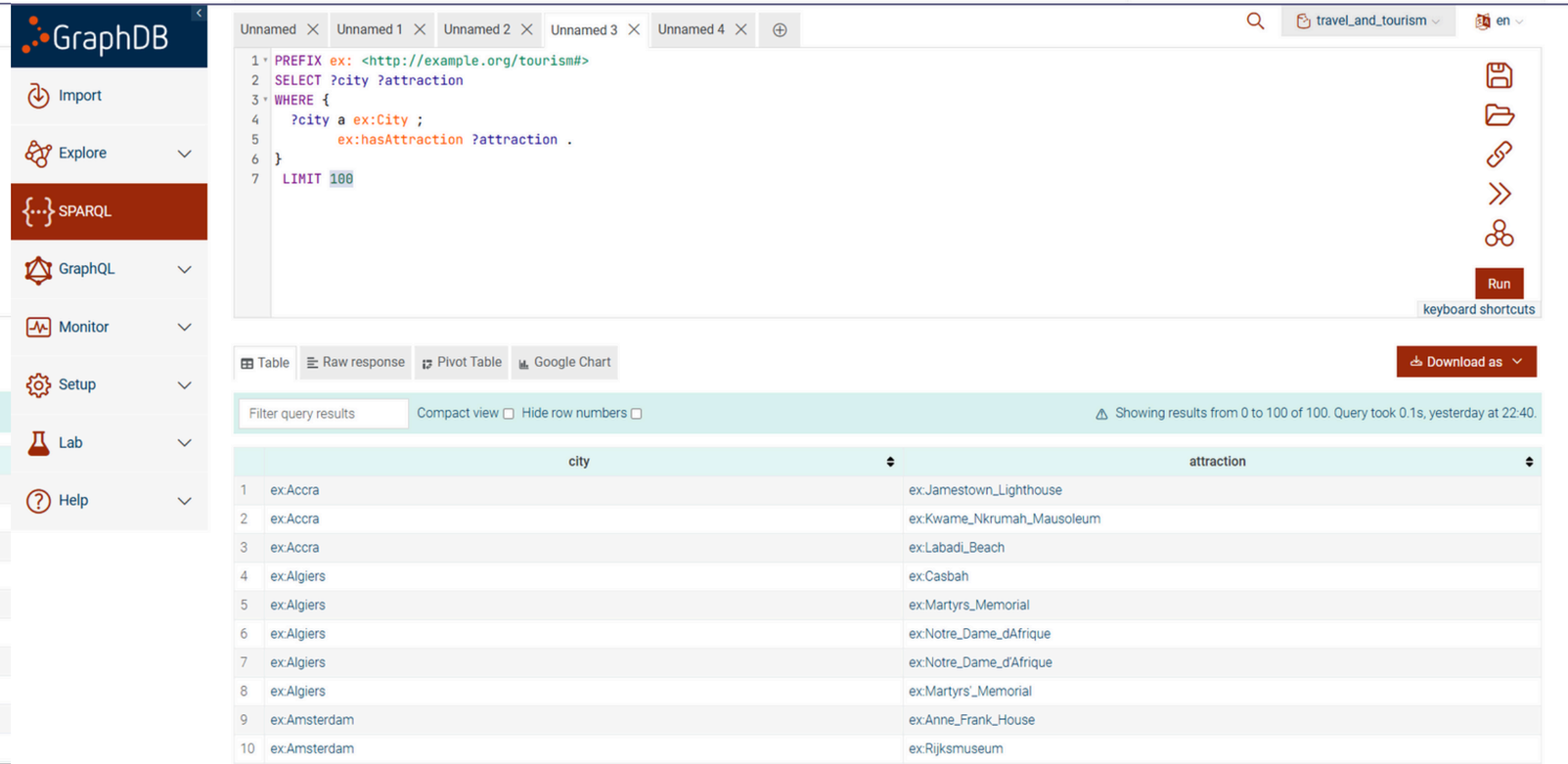
WHAT WE HAVE DONE SO FAR...

```python
    try:
        attractions = eval(row["Attractions"]) if isinstance(row["Attractions"], str) else []
        # print(f"{row['City']} -> {attractions}")

        for attr in attractions:
            attr_uri_safe = attr.strip().replace(" ", "_").replace(",", "")
            attraction_uri = URIRef(EX + attr_uri_safe)
            # labels for attractions
            g.add((attraction_uri, RDFS.label, Literal(attr, datatype=XSD.string)))
            g.add((attraction_uri, RDF.type, EX.Attraction))
            g.add((city_uri, EX.hasAttraction, attraction_uri))
    except Exception as e:
        pass

g.serialize(destination="tourism_data_generated.ttl", format="turtle")
g.serialize(destination="tourism_data_generated.rdf", format="xml")
g.serialize(destination="tourism_data_generated.owl", format="xml")
```


GraphDB
Import
Explore
SPARQL
GraphQL
Monitor
Setup
Lab
Help
SPARQL Query & Update
Unnamed
Unnamed 1
Unnamed 2
Unnamed 3
Unnamed 4
PREFIX ex: <http://example.org/tourism#>
SELECT ?city
WHERE {
?city a ex:City ;
ex:hasCountry ex:France .
}
Table
Raw response
Pivot Table
Google Chart
Filter query results
Compact view
Hide row numbers
city
1 ex:Paris


GraphDB
Import
Explore
SPARQL
GraphQL
Monitor
Setup
Lab
Help
Unnamed
Unnamed 1
Unnamed 2
Unnamed 3
Unnamed 4
travel_and_tourism
en
PREFIX ex: <http://example.org/tourism#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?attraction ?label
WHERE {
ex:Paris ex:hasAttraction ?attraction .
?attraction rdfs:label ?label .
}
Run
keyboard shortcuts
Table
Raw response
Pivot Table
Google Chart
Download result
Chart Config
Showing results from 0 to 3 of 3. Query took 0.1s, moments ago.
attraction
label
http://example.org/tourism#Eiffel_Tower
Eiffel Tower


GraphDB
Import
Explore
SPARQL
GraphQL
Monitor
Setup
Lab
Help
Unnamed
Unnamed 1
Unnamed 2
Unnamed 3
Unnamed 4
PREFIX ex: <http://example.org/tourism#>
SELECT ?city
WHERE {
?city a ex:City .
}
LIMIT 10
Table
Raw response
Pivot Table
Google Chart
Filter query results
Compact view
Hide row numbers
city
1 ex:Accra
2 ex:Algiers
3 ex:Amsterdam
4 ex:Asuncion
5 ex:Athens
6 ex:Auckland
7 ex:Bali
8 ex:Bangkok
9 ex:Barcelona
10 ex:Belgrade

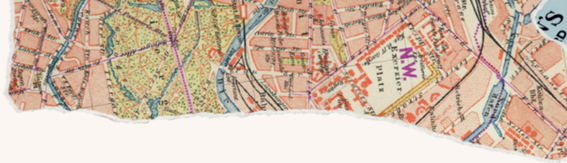

GraphDB
Import
Explore
SPARQL
GraphQL
Monitor
Setup
Lab
Help
Unnamed
Unnamed 1
Unnamed 2
Unnamed 3
Unnamed 4
travel_and_tourism
en
PREFIX ex: <http://example.org/tourism#>
SELECT ?city ?attraction
WHERE {
?city a ex:City ;
ex:hasAttraction ?attraction .
}
LIMIT 100
Run
keyboard shortcuts
Table
Raw response
Pivot Table
Google Chart
Download as
Filter query results
Compact view
Hide row numbers
Showing results from 0 to 100 of 100. Query took 0.1s, yesterday at 22:40.
city
attraction
1 ex:Accra ex:Jamestown_Lighthouse
2 ex:Accra ex:Kwame_Nkrumah_Mausoleum
3 ex:Accra ex:Labadi_Beach
4 ex:Algiers ex:Casbah
5 ex:Algiers ex:Martyrs_Memorial
6 ex:Algiers ex:Notre_Dame_dAfrique
7 ex:Algiers ex:Notre_Dame_d'Afrique
8 ex:Algiers ex:Martyrs'_Memorial
9 ex:Amsterdam ex:Anne_Frank_House
10 ex:Amsterdam ex:Rijksmuseum

Next Steps

- Improving the database
- Ontology improvement
- Using inference
- Complex querries

Thank You

Questions?

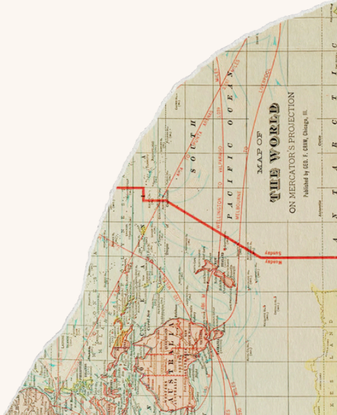











Bibliography

- https://www.atlasobscura.com/destinations
- https://www.tripadvisor.com/
- https://www.numbeo.com/cost-of-living/
- https://www.w3.org/TR/rdf-primer/
- https://www.w3.org/TR/owl-guide/
- https://www.w3.org/TR/owl-guide/
- https://www.kaggle.com/datasets/myrios/cost-of-living-index-by-country-by-number-2024
- https://www.kaggle.com/datasets/umeradnaan/tourism-dataset
- https://www.booking.com/?msockid=37e53acf232c66e339e62e7f22d6676d
- https://www.airbnb.com/