Web Datamining Report

Food Delivery App

Group Members:

- QIAN Victor
- RAMPONT Martin

Objective: This report aims to create a food delivery app based on an ontology whose instances are parsed from several sources.

GitHub Link: https://github.com/Mcrash01/web-semantics-food-delivery-app

NB: the problem statement is uploaded on GitHub.

Table of contents

Part 1 | Creating the ontology

Data Structure

We have chosen to make the following data structure for our ontology. This data structure is essential for our future SPARQL queries when querying the ontology.

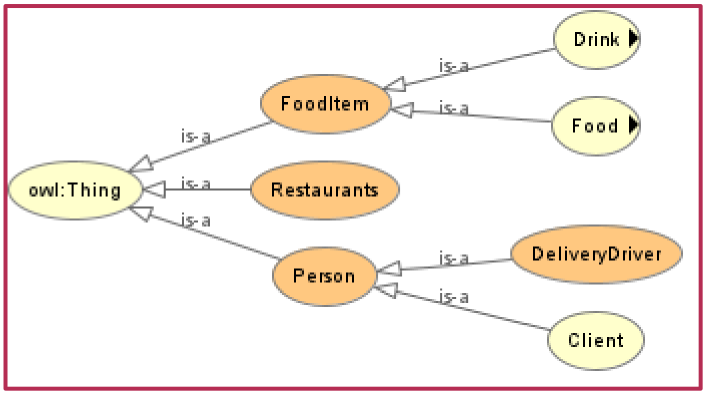
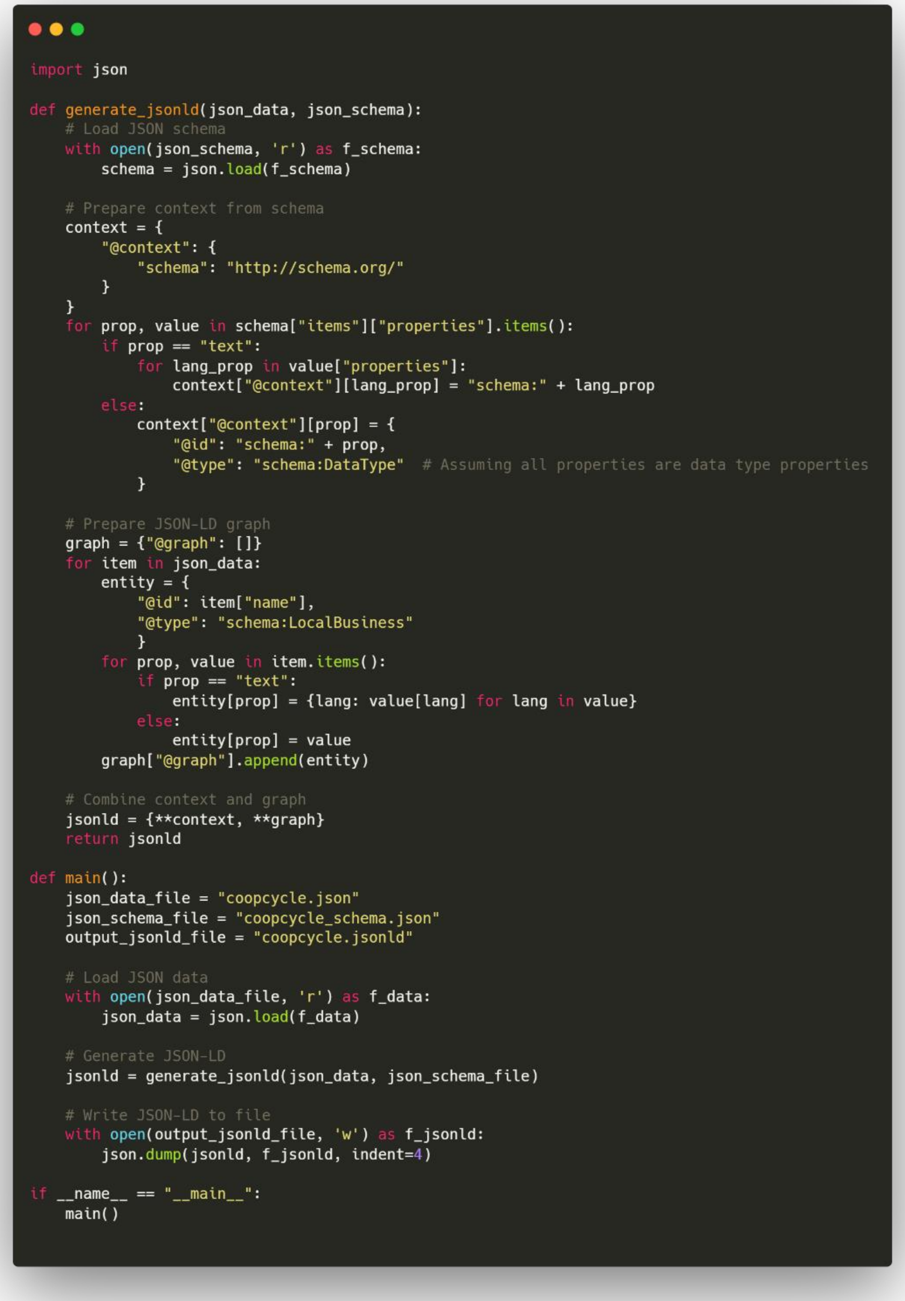


Figure 1: Ontology Structure

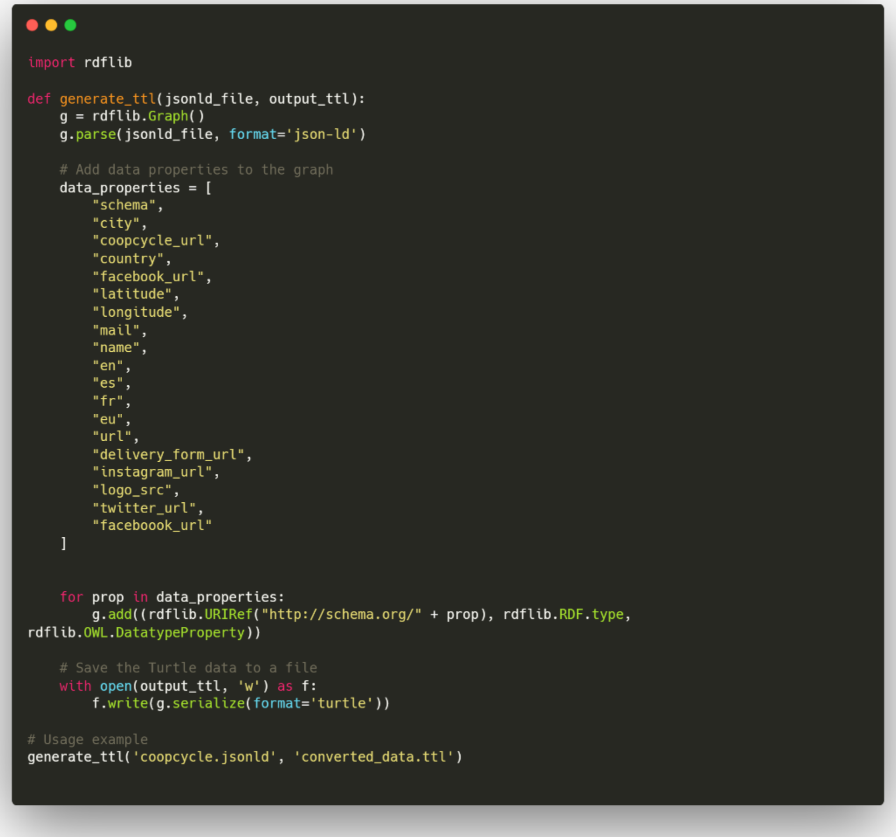
PART 2 | Data Import

Delivery Services

To import Delivery Services, we retrieved the coopcycle json file, converted it to json-ld (adding the json schema as a context), then converted it to ttl using the following scripts:

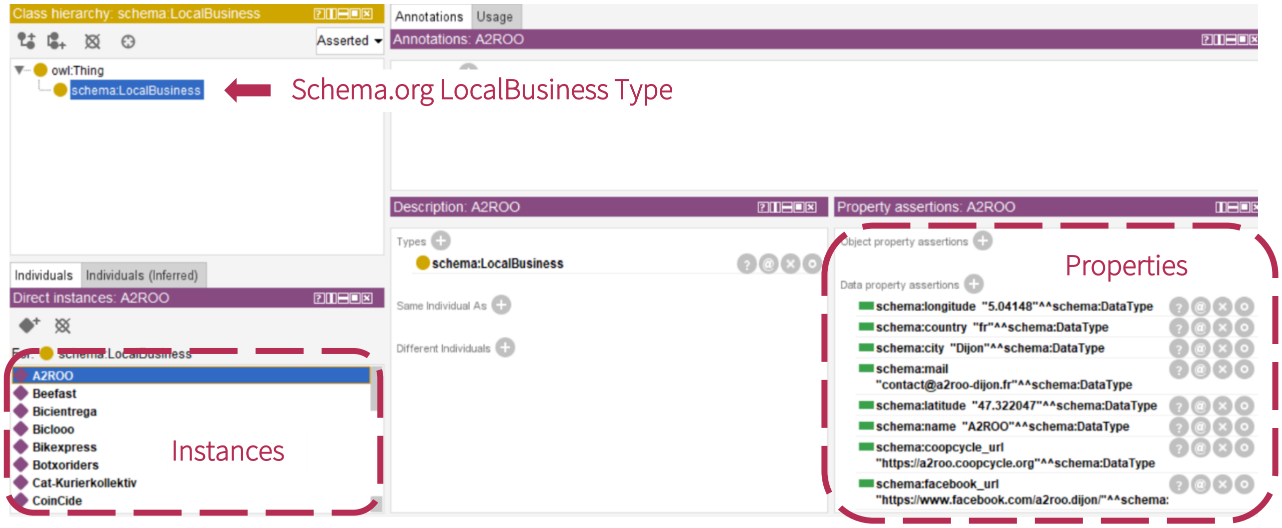


```python
import json

def generate_jsonld(json_data, json_schema):
    # Load JSON schema
    with open(json_schema, 'r') as f_schema:
        schema = json.load(f_schema)

    # Prepare context from schema
    context = {
        "@context": {
            "schema": "http://schema.org/"
        }
    }
    for prop, value in schema["items"]["properties"].items():
        if prop == "text":
            for lang_prop in value["properties"]:
                context["@context"][lang_prop] = "schema:" + lang_prop
        else:
            context["@context"][prop] = {
                "@id": "schema:" + prop,
                "@type": "schema:DataType"  # Assuming all properties are data type properties
            }

    # Prepare JSON-LD graph
    graph = {"@graph": []}
    for item in json_data:
        entity = {
            "@id": item["name"],
            "@type": "schema:LocalBusiness"
            }
        for prop, value in item.items():
            if prop == "text":
                entity[prop] = {lang: value[lang] for lang in value}
            else:
                entity[prop] = value
        graph["@graph"].append(entity)

    # Combine context and graph
    jsonld = {**context, **graph}
    return jsonld

def main():
    json_data_file = "coopcycle.json"
    json_schema_file = "coopcycle_schema.json"
    output_jsonld_file = "coopcycle.jsonld"

    # Load JSON data
    with open(json_data_file, 'r') as f_data:
        json_data = json.load(f_data)

    # Generate JSON-LD
    jsonld = generate_jsonld(json_data, json_schema_file)

    # Write JSON-LD to file
    with open(output_jsonld_file, 'w') as f_jsonld:
        json.dump(jsonld, f_jsonld, indent=4)

if __name__ == "__main__":
    main()
```



```python
import rdflib

def generate_ttl(jsonld_file, output_ttl):
    g = rdflib.Graph()
    g.parse(jsonld_file, format='json-ld')

    # Add data properties to the graph
    data_properties = [
        "schema",
        "city",
        "coopcycle_url",
        "country",
        "facebook_url",
        "latitude",
        "longitude",
        "mail",
        "name",
        "en",
        "es",
        "fr",
        "eu",
        "url",
        "delivery_form_url",
        "instagram_url",
        "logo_src",
        "twitter_url",
        "faceboook_url"
    ]


    for prop in data_properties:
        g.add((rdflib.URIRef("http://schema.org/" + prop), rdflib.RDF.type,
rdflib.OWL.DatatypeProperty))

    # Save the Turtle data to a file
    with open(output_ttl, 'w') as f:
        f.write(g.serialize(format='turtle'))

# Usage example
generate_ttl('coopcycle.jsonld', 'converted_data.ttl')
```

We ran into a problem, because all the properties were imported as annotations, but after adding specific rules to our turtle file, we finally managed to import all the delivery services:



Schema.org LocalBusiness Type

schema:LocalBusiness

Instances

A2ROO
Beefast
Bicientrega
Biclooo
Bikexpress
Botxoriders
Cat-Kurierkollektiv
CoinCide

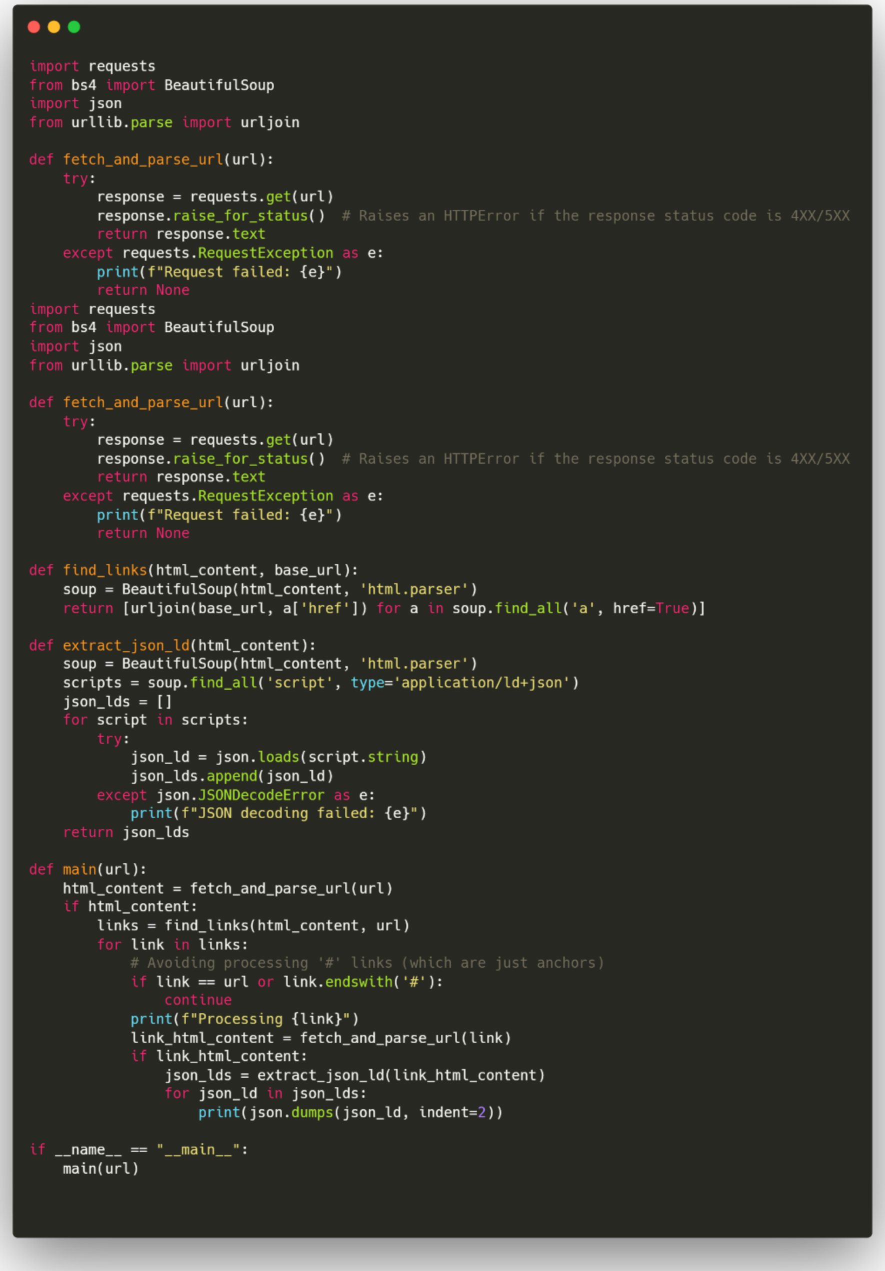
Properties

schema:longitude "5.04148"^^schema:DataType
schema:country "fr"^^schema:DataType
schema:city "Dijon"^^schema:DataType
schema:mail "contact@a2roo-dijon.fr"^^schema:DataType
schema:latitude "47.322047"^^schema:DataType
schema:name "A2ROO"^^schema:DataType
schema:coopcycle_url "https://a2roo.coopcycle.org"^^schema:DataType
schema:facebook_url "https://www.facebook.com/a2roo.dijon/"^^schema:

Restaurants

For restaurants we made a custom HTML parser with BeatifulSoup that goes through all coopcycle_url of the json file. It then extracts the data by parsing json-ld. The URL parsed are constructed as following:

{coopcycle_url} + ”/fr/shops?type=restaurant

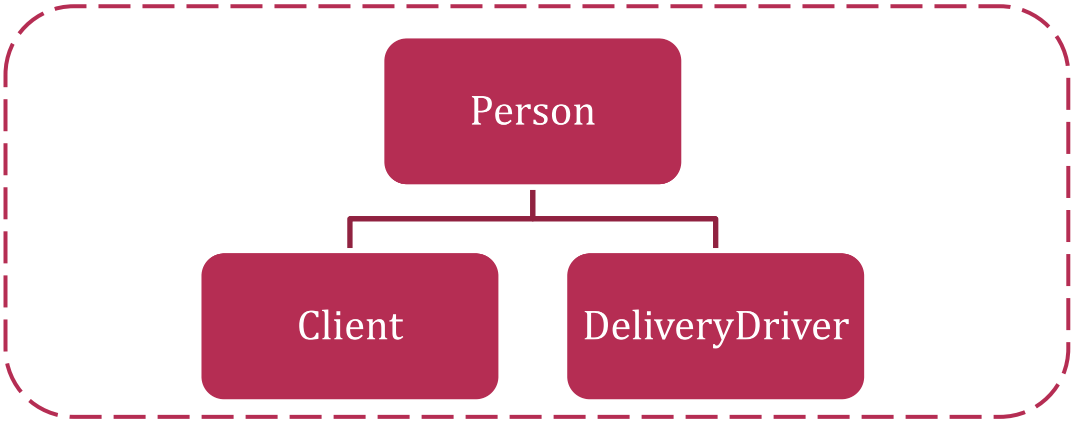


```
import requests
from bs4 import BeautifulSoup
import json
from urllib.parse import urljoin

def fetch_and_parse_url(url):
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raises an HTTPError if the response status code is 4XX/5XX
        return response.text
    except requests.RequestException as e:
        print(f"Request failed: {e}")
        return None
import requests
from bs4 import BeautifulSoup
import json
from urllib.parse import urljoin

def fetch_and_parse_url(url):
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raises an HTTPError if the response status code is 4XX/5XX
        return response.text
    except requests.RequestException as e:
        print(f"Request failed: {e}")
        return None

def find_links(html_content, base_url):
    soup = BeautifulSoup(html_content, 'html.parser')
    return [urljoin(base_url, a['href']) for a in soup.find_all('a', href=True)]

def extract_json_ld(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    scripts = soup.find_all('script', type='application/ld+json')
    json_lds = []
    for script in scripts:
        try:
            json_ld = json.loads(script.string)
            json_lds.append(json_ld)
        except json.JSONDecodeError as e:
            print(f"JSON decoding failed: {e}")
    return json_lds

def main(url):
    html_content = fetch_and_parse_url(url)
    if html_content:
        links = find_links(html_content, url)
        for link in links:
            # Avoiding processing '#' links (which are just anchors)
            if link == url or link.endswith('#'):
                continue
            print(f"Processing {link}")
            link_html_content = fetch_and_parse_url(link)
            if link_html_content:
                json_lds = extract_json_ld(link_html_content)
                for json_ld in json_lds:
                    print(json.dumps(json_ld, indent=2))

if __name__ == "__main__":
    main(url)
```

Food

Persons

Persons have been populated manually. We have two types of persons in our ontology:

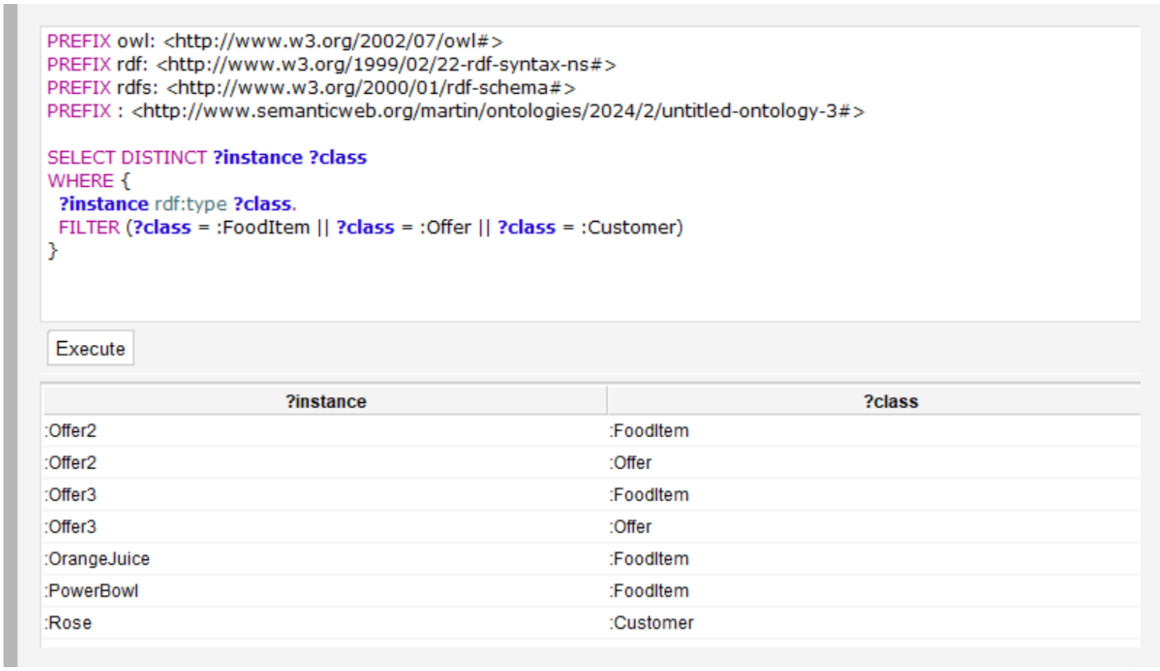


PART 3 | QUERRIES

SIMPLE	Advanced
5	5

Simple Queries

1 List the instances of the class food products, offers, and customers:

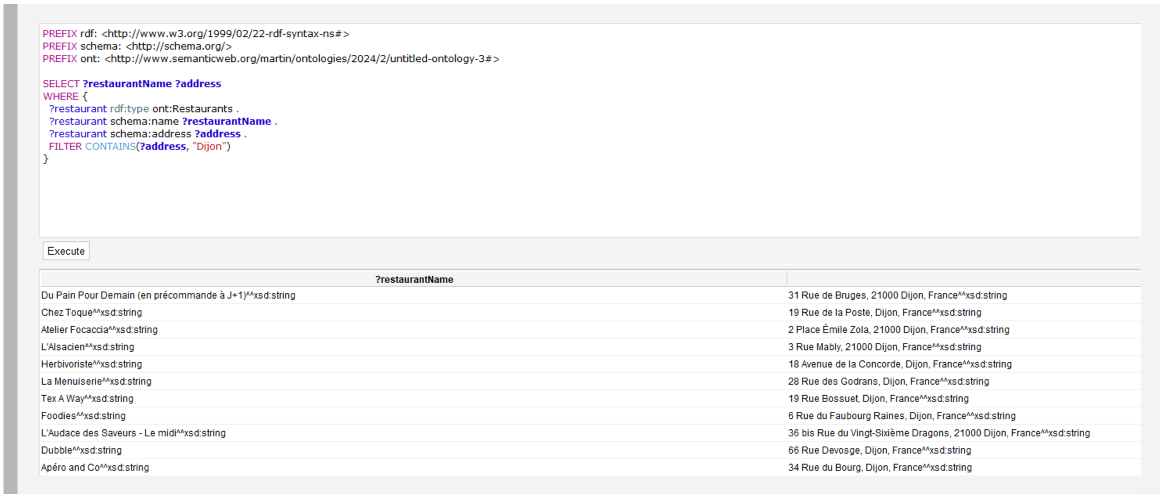


```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/martin/ontologies/2024/2/untitled-ontology-3#>

SELECT DISTINCT ?instance ?class
WHERE {
 ?instance rdf:type ?class.
 FILTER (?class = :FoodItem || ?class = :Offer || ?class = :Customer)
}
```

?instance	?class
:Offer2	:FoodItem
:Offer2	:Offer
:Offer3	:FoodItem
:Offer3	:Offer
:OrangeJuice	:FoodItem
:PowerBowl	:FoodItem
:Rose	:Customer

2 **List the name of all Paris restaurants.**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>
PREFIX ont:
<http://www.semanticweb.org/martin/ontologies/2024/2/untitled-
ontology-3#>

SELECT ?restaurantName ?address
WHERE {
  ?restaurant rdf:type ont:Restaurants .
  ?restaurant schema:name ?restaurantName .
  ?restaurant schema:address ?address .
  FILTER CONTAINS(?address, "Dijon")
}
```

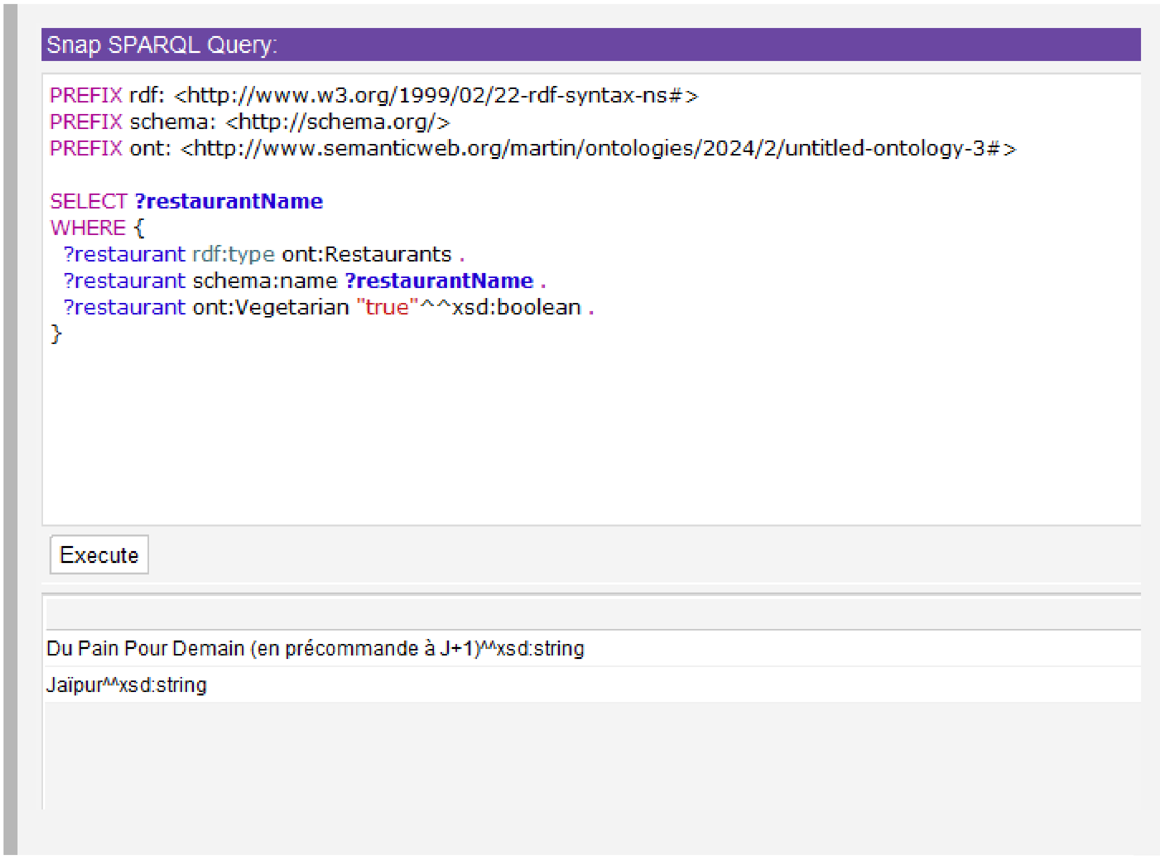


```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>
PREFIX ont: <http://www.semanticweb.org/martin/ontologies/2024/2/untitled-ontology-3#>

SELECT ?restaurantName ?address
WHERE {
  ?restaurant rdf:type ont:Restaurants .
  ?restaurant schema:name ?restaurantName .
  ?restaurant schema:address ?address .
  FILTER CONTAINS(?address, "Dijon")
}
```

?restaurantName	
Du Pain Pour Demain (en précommande à J+1)^^xsd:string	31 Rue de Bruges, 21000 Dijon, France^^xsd:string
Chez Toque^^xsd:string	19 Rue de la Poste, Dijon, France^^xsd:string
Atelier Focaccia^^xsd:string	2 Place Émile Zola, 21000 Dijon, France^^xsd:string
L'Alsacien^^xsd:string	3 Rue Mably, 21000 Dijon, France^^xsd:string
Herbivoriste^^xsd:string	18 Avenue de la Concorde, Dijon, France^^xsd:string
La Menuiserie^^xsd:string	28 Rue des Godrans, Dijon, France^^xsd:string
Tex A Way^^xsd:string	19 Rue Bossuet, Dijon, France^^xsd:string
Foodies^^xsd:string	6 Rue du Faubourg Raines, Dijon, France^^xsd:string
L'Audace des Saveurs - Le midi^^xsd:string	36 bis Rue du Vingt-Sixième Dragons, 21000 Dijon, France^^xsd:string
Dubble^^xsd:string	66 Rue Devosge, Dijon, France^^xsd:string
Apéro and Co^^xsd:string	34 Rue du Bourg, Dijon, France^^xsd:string

3 **List the name of all vegetarian restaurants, for each one, display their delivery services.**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>
PREFIX ont:
<http://www.semanticweb.org/martin/ontologies/2024/2/untitled-
ontology-3#>

SELECT ?restaurantName
WHERE {
  ?restaurant rdf:type ont:Restaurants .
  ?restaurant schema:name ?restaurantName .
  ?restaurant ont:Vegetarian "true"^^xsd:boolean .
}
```

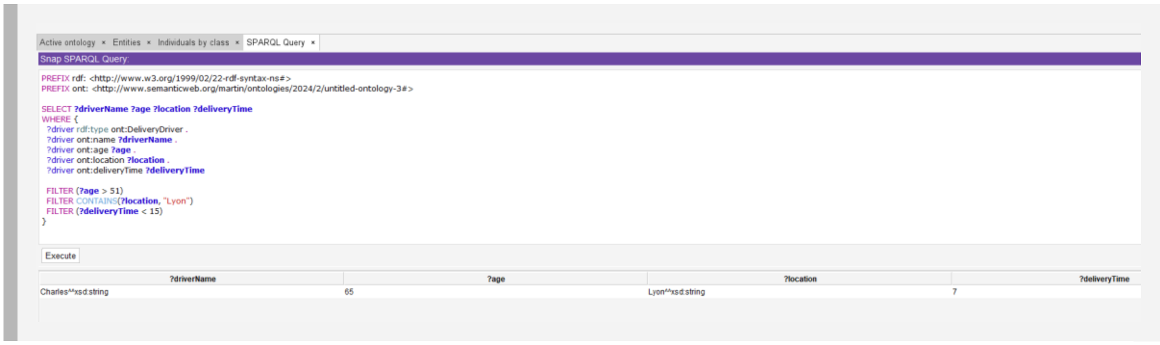


```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>
PREFIX ont: <http://www.semanticweb.org/martin/ontologies/2024/2/untitled-ontology-3#>

SELECT ?restaurantName
WHERE {
  ?restaurant rdf:type ont:Restaurants .
  ?restaurant schema:name ?restaurantName .
  ?restaurant ont:Vegetarian "true"^^xsd:boolean .
}
```

```
Du Pain Pour Demain (en précommande à J+1)^^xsd:string
Jaïpur^^xsd:string
```

4 **List the name of deliverymen older than 51 years and they can deliver in Lyon**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont:
<http://www.semanticweb.org/martin/ontologies/2024/2/untitled-
ontology-3#>

SELECT ?driverName ?age ?location ?deliveryTime
WHERE {
  ?driver rdf:type ont:DeliveryDriver .
  ?driver ont:name ?driverName .
  ?driver ont:age ?age .
  ?driver ont:location ?location .
  ?driver ont:deliveryTime ?deliveryTime

  FILTER (?age > 51)
  FILTER CONTAINS(?location, "Lyon")
  FILTER (?deliveryTime < 15)
}
```

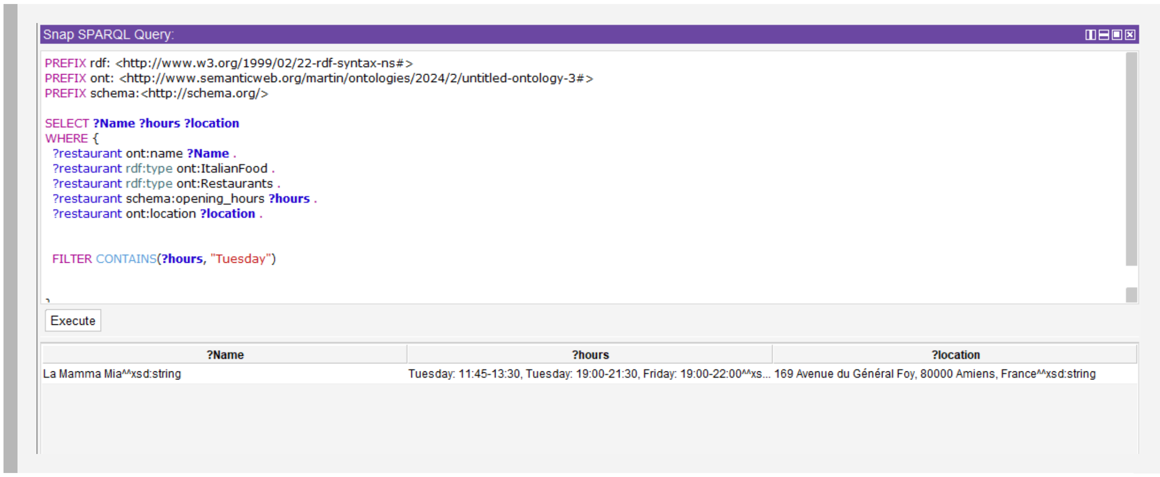


```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont: <http://www.semanticweb.org/martin/ontologies/2024/2/untitled-ontology-3#>

SELECT ?driverName ?age ?location ?deliveryTime
WHERE {
 ?driver rdf:type ont:DeliveryDriver .
 ?driver ont:name ?driverName .
 ?driver ont:age ?age .
 ?driver ont:location ?location .
 ?driver ont:deliveryTime ?deliveryTime

 FILTER (?age > 51)
 FILTER CONTAINS(?location, "Lyon")
 FILTER (?deliveryTime < 15)
}
```

?driverName	?age	?location	?deliveryTime
Charles^^xsd:string	65	Lyon^^xsd:string	7

5 **List of restaurants that serve Italian food for a specific day and where and until when.**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont:
<http://www.semanticweb.org/martin/ontologies/2024/2/untitled-
ontology-3#>
PREFIX schema:<http://schema.org/>

SELECT ?Name ?hours ?location
WHERE {
  ?restaurant ont:name ?Name .
  ?restaurant rdf:type ont:ItalianFood .
  ?restaurant rdf:type ont:Restaurants .
  ?restaurant schema:opening_hours ?hours .
  ?restaurant ont:location ?location .


  FILTER CONTAINS(?hours, "Tuesday")


}
```



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont: <http://www.semanticweb.org/martin/ontologies/2024/2/untitled-ontology-3#>
PREFIX schema:<http://schema.org/>

SELECT ?Name ?hours ?location
WHERE {
 ?restaurant ont:name ?Name .
 ?restaurant rdf:type ont:ItalianFood .
 ?restaurant rdf:type ont:Restaurants .
 ?restaurant schema:opening_hours ?hours .
 ?restaurant ont:location ?location .


 FILTER CONTAINS(?hours, "Tuesday")


}
```

?Name	?hours	?location
La Mamma Mia^^xsd:string	Tuesday: 11:45-13:30, Tuesday: 19:00-21:30, Friday: 19:00-22:00^^xs...	169 Avenue du Général Foy, 80000 Amiens, France^^xsd:string

Advanced Queries

Query	Description
Optional Graph Patterns	Retrieves names, email addresses, and addresses of persons if available.
Alternatives and Conjunctions	Retrieves names of restaurants serving Italian or French cuisine in Paris with express delivery service.
CONSTRUCT Query Form	Constructs triples for restaurants in Paris serving Italian cuisine with express delivery service.
ASK Query Form	Checks if there are any restaurants serving Italian cuisine with express delivery service.
DESCRIBE Query Form	Describes restaurants in Paris serving Italian cuisine based on the ontology.

1 **Retrieves names, email addresses, and addresses of persons if available**

```
SELECT ? Name ?email ?address
WHERE {
  ?person rdf:type :Person;
          :name ? Name.
  OPTIONAL {
    ?person :hasEmail ?email.
  }
  OPTIONAL {
    ?person :hasLocation ?location.
  }
}
```

2 **Retrieves names of restaurants that have an email and a phone number.**

```
SELECT ?restaurantName ?email ?PhoneNumber

WHERE {

  ?restaurant rdf:type :Restaurant;

              (:hasEmail ?email) && (hasPhoneNumber ?phoneNumber);

     ?restaurant :name ?restaurantName.

}
```

3 **Constructs triples for restaurants in Paris serving Italian cuisine with express delivery service.**

```
CONSTRUCT {

  ?restaurant :hasDeliveryService "Express".

}

WHERE {

  ?restaurant rdf:type :Restaurant;

              (:locatedInCity "Paris") && (:servesFood :Italian).

}
```

4 **Checks if there are any restaurants serving Italian cuisine with express delivery service.**

```
ASK {
  ?restaurant rdf:type :Restaurant;
            (:servesFood :Italian) && (:hasDeliveryService "Express").
}
```

5 **Describes restaurants in Paris serving Italian cuisine based on the ontology.**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>

DESCRIBE ?restaurant
WHERE {
  ?restaurant rdf:type schema:Restaurant .
  ?restaurant schema:address ?address .
  ?restaurant schema:servesFood ?food .

  ?food rdf:type schema:ItalianFood .

  FILTER CONTAINS(?address, "Paris")
}
```