

```
In [1]: from rdflib import Graph

# Create an RDF graph
g = Graph()

# Load RDF data from a file (Turtle format)
g.parse("personshape_SHACL_data.ttl", format="ttl")
```

```
Out[1]: <Graph identifier=Nefaf778d5de9d4fb5878ed2b2b0a941f3 (<class 'rdflib.graph.Graph'>) >
```

```
In [2]: query = """
PREFIX ex: <http://example.org/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?person ?name ?city ?country ?allergy
      (GROUP_CONCAT(?medication; separator=", ") AS ?medications)
WHERE {
    ?person a ex:Person ;
        ex:hasName ?name ;
        ex:hasCity ?city ;
        ex:hasCountry ?country ;
        ex:isAllergicTo ?allergy ;
        ex:takesMedication ?medication .

    FILTER(?city = dbr:Paris && ?country = dbr:France)
}
GROUP BY ?person ?name ?city ?country ?allergy
ORDER BY ?name ?allergy
"""
```

```
In [3]: # Run the SPARQL query
results = g.query(query)

# Display results
for row in results:
    print(f"Person: {row.person}, Name: {row.name}, City: {row.city}, "
          f"Country: {row.country}, Allergy: {row.allergy}, Medications: {row.medications}")
```

Person: <http://example.org/item/20>, Name: Bryan Harris, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Gluten_allergy, Medications: <http://dbpedia.org/resource/Salbutamol>

Person: <http://example.org/item/72>, Name: Jonas Morgan, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Soy_allergy, Medications: <http://dbpedia.org/resource/Insulin>, <http://dbpedia.org/resource/Metformin>

Person: <http://example.org/item/23>, Name: Jonathan Rivas, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Lactose_intolerance, Medications: <http://dbpedia.org/resource/Insulin>, <http://dbpedia.org/resource/Metformin>

Person: <http://example.org/item/1>, Name: Mark Jackson MD, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Dust_mite_allergy, Medications: <http://dbpedia.org/resource/Epinephrine>, <http://dbpedia.org/resource/Isoniazid>, <http://dbpedia.org/resource/Salbutamol>

Person: <http://example.org/item/1>, Name: Mark Jackson MD, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Peanut_allergy, Medications: <http://dbpedia.org/resource/Epinephrine>, <http://dbpedia.org/resource/Isoniazid>, <http://dbpedia.org/resource/Salbutamol>

Person: <http://example.org/item/17>, Name: Michelle Ayala, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Gluten_allergy, Medications: <http://dbpedia.org/resource/Insulin>, <http://dbpedia.org/resource/Metformin>

Person: <http://example.org/item/17>, Name: Michelle Ayala, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Soy_allergy, Medications: <http://dbpedia.org/resource/Insulin>, <http://dbpedia.org/resource/Metformin>

Person: <http://example.org/item/21>, Name: Sherry Turner, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Soy_allergy, Medications: <http://dbpedia.org/resource/Salbutamol>

Person: <http://example.org/item/81>, Name: Susan Murphy, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Gluten_allergy, Medications: <http://dbpedia.org/resource/Insulin>, <http://dbpedia.org/resource/Metformin>

Person: <http://example.org/item/47>, Name: Valerie Thornton, City: <http://dbpedia.org/resource/Paris>, Country: <http://dbpedia.org/resource/France>, Allergy: http://dbpedia.org/resource/Lactose_intolerance, Medications: <http://dbpedia.org/resource/Salbutamol>

```
In [4]: from rdflib import Graph
import pandas as pd

# Load RDF data
g = Graph()
g.parse("personshape_SHACL_data.ttl", format="ttl")

# SPARQL query (city-country + allergies and medications)
query = """
PREFIX ex: <http://example.org/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?person ?name ?city ?country ?allergy
```

```

        (GROUP_CONCAT(?medication; separator=", ") AS ?medications)
WHERE {
    ?person a ex:Person ;
        ex:hasName ?name ;
        ex:hasCity ?city ;
        ex:hasCountry ?country ;
        ex:isAllergicTo ?allergy ;
        ex:takesMedication ?medication .

        FILTER(?city = dbr:Paris && ?country = dbr:France)
}
GROUP BY ?person ?name ?city ?country ?allergy
ORDER BY ?name ?allergy
"""

# Execute query
results = g.query(query)

# Convert to pandas DataFrame
df = pd.DataFrame([row.asdict() for row in results])
df.head()

```

Out [4] :

	person	name	city	
0	http://example.org/item/20	Bryan Harris	http://dbpedia.org/resource/Paris	http://dbpedia.org/
1	http://example.org/item/72	Jonas Morgan	http://dbpedia.org/resource/Paris	http://dbpedia.org/
2	http://example.org/item/23	Jonathan Rivas	http://dbpedia.org/resource/Paris	http://dbpedia.org/
3	http://example.org/item/1	Mark Jackson MD	http://dbpedia.org/resource/Paris	http://dbpedia.org/
4	http://example.org/item/1	Mark Jackson MD	http://dbpedia.org/resource/Paris	http://dbpedia.org/

In [5] : `import matplotlib.pyplot as plt`

```

# Show as a table in Jupyter
fig, ax = plt.subplots(figsize=(10, 2))
ax.axis('tight')
ax.axis('off')
ax.table(cellText=df.values, colLabels=df.columns, loc='center', cellLoc='center')
plt.show()

```

```
In [6]: import networkx as nx

G = nx.Graph()

# Add nodes and edges
for _, row in df.iterrows():
    person = row['name']
    allergies = row['allergy']
    medications = row['medications'].split(", ")

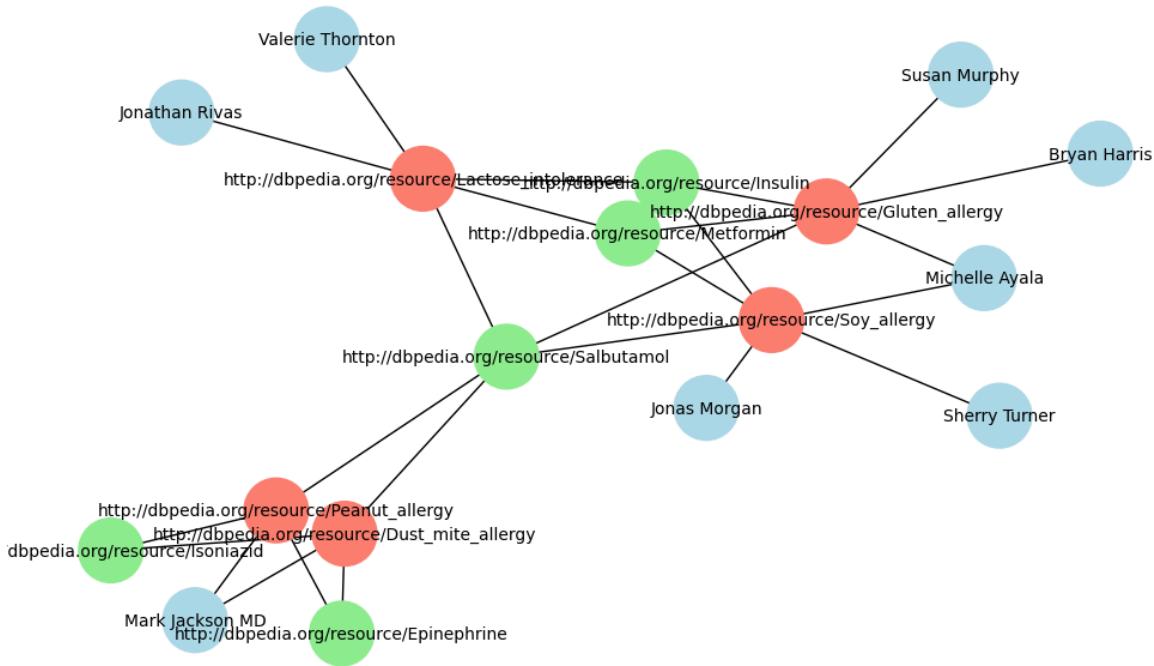
    G.add_node(person, type='person')

    # Allergy edges
    G.add_node(allergies, type='allergy')
    G.add_edge(person, allergies)

    # Medication edges
    for med in medications:
        G.add_node(med, type='medication')
        G.add_edge(allergies, med)

# Draw the graph
pos = nx.spring_layout(G, seed=42)
colors = []
for node in G.nodes(data=True):
    if node[1]['type'] == 'person':
        colors.append('lightblue')
    elif node[1]['type'] == 'allergy':
        colors.append('salmon')
    else:
        colors.append('lightgreen')

plt.figure(figsize=(10, 6))
nx.draw(G, pos, with_labels=True, node_color=colors, node_size=1500, font_si
plt.show()
```



```
In [10]: from pyvis.network import Network
from selenium import webdriver
import time

# Create the pyvis network
net = Network(height="500px", width="100%")
for _, row in df.iterrows():
    person = row['name']
    allergies = row['allergy']
    medications = row['medications'].split(", ")

    net.add_node(person, label=person, color='lightblue')
    net.add_node(allergies, label=allergies, color='salmon')
    net.add_edge(person, allergies)

    for med in medications:
        net.add_node(med, label=med, color='lightgreen')
        net.add_edge(allergies, med)

# Save interactive HTML first
html_file = "person_allergy_medications.html"
net.save_graph(html_file)

# Configure Selenium with headless Chrome
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--window-size=1200,800')
driver = webdriver.Chrome(options=options)

# Load the HTML
driver.get(f"file:///{html_file}")
time.sleep(2) # Wait for graph to render
```

```
# Take screenshot
driver.save_screenshot("person_allergy_medications.jpg")
driver.quit()
```

```
/opt/anaconda3/lib/python3.12/site-packages/selenium/webdriver/remote/webdriver.py:1013: UserWarning: name used for saved screenshot does not match file type. It should end with a '.png` extension
    return self.get_screenshot_as_file(filename)
```

```
In [17]: from rdflib import Graph, Namespace, RDF, RDFS, OWL
import pandas as pd

# Load RDF data
g = Graph()
g.parse("GAN_ProteinShape_data.ttl", format="ttl")

# Namespaces
oboInOwl = Namespace("http://www.geneontology.org/formats/oboInOwl#")
pr = Namespace("http://purl.obolibrary.org/obo/pr#")
dcterms = Namespace("http://purl.org/dc/terms/")

# Example SPARQL: Proteins associated with GO term GO:0031092
query = """
PREFIX pr: <http://purl.obolibrary.org/obo/pr#>
PREFIX oboInOwl: <http://www.geneontology.org/formats/oboInOwl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?protein ?label ?go_term
WHERE {
    ?protein a ?type ;
        rdfs:label ?label ;
        owl:someValuesFrom ?go_term .

    FILTER(CONTAINS(STR(?go_term), "GO_0031092"))
}
"""

# Execute query
results = g.query(query)

# Convert to DataFrame
df = pd.DataFrame([row.asdict() for row in results])
df.head()
```

Out[17]: —

```
In [19]: from rdflib import Graph, Namespace
import pandas as pd

# Load RDF
g = Graph()
g.parse("GAN_ProteinShape_data.ttl", format="ttl")

# Namespaces
pr = Namespace("http://purl.obolibrary.org/obo/pr#")
```

```

oboInOwl = Namespace("http://www.geneontology.org/formats/oboInOwl#")
NCBITaxon = Namespace("http://purl.obolibrary.org/obo/NCBITaxon_")

# Query proteins linked to a species, e.g., NCBITaxon_9606 (human)
query = """
PREFIX pr: <http://purl.obolibrary.org/obo/pr#>
PREFIX oboInOwl: <http://www.geneontology.org/formats/oboInOwl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?protein ?label ?species
WHERE {
    ?protein a ?type ;
              rdfs:label ?label ;
              oboInOwl:hasRelatedSynonym ?species .

    FILTER(CONTAINS(STR(?species), "NCBITaxon_9606"))
}
"""

# Execute query
results = g.query(query)

# Convert to DataFrame
df = pd.DataFrame([row.asdict() for row in results])
df.head()

```

Out[19]:

		protein	label
0	http://purl.obolibrary.org/obo/pr#000000453_1	UniProtKB:P38110	http://purl.obolibrary.org
1	http://purl.obolibrary.org/obo/pr#000000453_3	Rab11-FIP3 (human)	http://purl.obolibrary.org
2	http://purl.obolibrary.org/obo/pr#000003452_1	A protein that is a translation product of the...	http://purl.obolibrary.org
3	http://purl.obolibrary.org/obo/pr#000003452_1	A protein that is a translation product of the...	http://purl.obolibrary.org
4	http://purl.obolibrary.org/obo/pr#000003452_2	An interleukin-1 receptor type 1 that is a tra...	http://purl.obolibrary.org

In [20]:

```

import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()

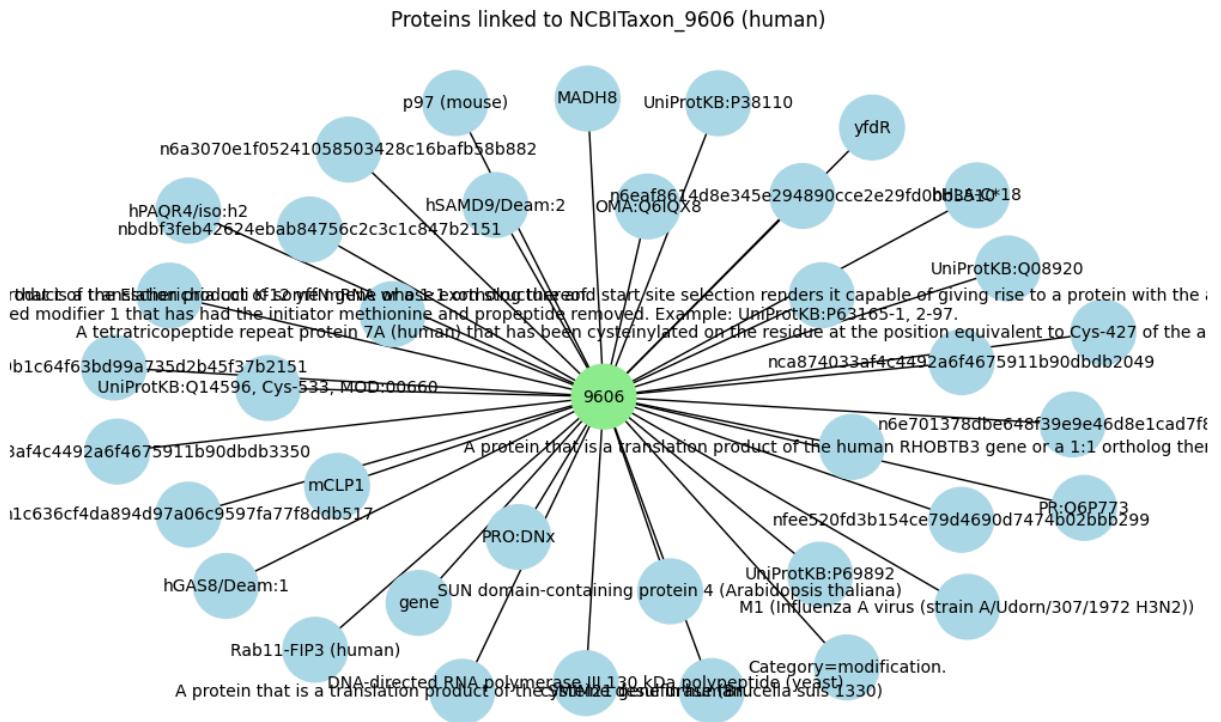
for _, row in df.iterrows():
    protein = row['label']
    species = row['species'].split("_")[-1] # clean species ID

    G.add_node(protein, type='protein')
    G.add_node(species, type='species')
    G.add_edge(protein, species)

```

```
# Draw graph
pos = nx.spring_layout(G, seed=42)
colors = ['lightblue' if node[1]['type']=='protein' else 'lightgreen' for nc

plt.figure(figsize=(10,6))
nx.draw(G, pos, with_labels=True, node_color=colors, node_size=1500, font_si
plt.title("Proteins linked to NCBITaxon_9606 (human)")
plt.show()
```



```
In [22]: from rdflib import Graph, Namespace
import pandas as pd

# Load RDF
g = Graph()
g.parse("GAN_ProteinShape_data.ttl", format="ttl")

# Namespaces
pr = Namespace("http://purl.obolibrary.org/obo/pr#")
oboInOwl = Namespace("http://www.geneontology.org/formats/oboInOwl#")
owl = Namespace("http://www.w3.org/2002/07/owl#")
rdfs = Namespace("http://www.w3.org/2000/01/rdf-schema#")

# Query proteins, GO terms, and synonyms
query = """
PREFIX pr: <http://purl.obolibrary.org/obo/pr#>
PREFIX oboInOwl: <http://www.geneontology.org/formats/oboInOwl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?protein ?protein_label ?GO ?synonym
WHERE {
    ?protein rdfs:label ?protein_label .
```

```

OPTIONAL { ?protein owl:annotatedSource ?GO . }
OPTIONAL {
    ?protein oboInOwl:hasExactSynonym ?synonym .
}
}
LIMIT 100
"""

# Execute query
results = g.query(query)

# Convert to pandas DataFrame
df = pd.DataFrame([row.asdict() for row in results])
df.head()

```

Out[22]:

		protein	protein_label	
0	http://purl.obolibrary.org/obo/pr#000000453_1	UniProtKB:P38110		http://purl.obolibrary.org/obo/pr#000000453_1
1	http://purl.obolibrary.org/obo/pr#000000453_5	UniProtKB:P38110		http://purl.obolibrary.org/obo/pr#000000453_5
2	http://purl.obolibrary.org/obo/pr#000063115_3	UniProtKB:P38110		
3	http://purl.obolibrary.org/obo/pr#000000453_2		Rab11-FIP3 (human)	http://purl.obolibrary.org/obo/pr#000000453_2
4	http://purl.obolibrary.org/obo/pr#000000453_3		Rab11-FIP3 (human)	http://purl.obolibrary.org/obo/pr#000000453_3

In [23]:

```

import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()

for _, row in df.iterrows():
    protein = row['protein_label']
    GO = row['GO'] if row['GO'] else None
    synonym = row['synonym'] if row['synonym'] else None

    G.add_node(protein, type='protein')

    if GO:
        G.add_node(GO, type='GO')
        G.add_edge(protein, GO)

    if synonym:
        G.add_node(synonym, type='synonym')
        G.add_edge(protein, synonym)

# Draw graph
pos = nx.spring_layout(G, seed=42)
colors = []
for node in G.nodes(data=True):
    if node[1]['type']=='protein':

```

```
    colors.append('lightblue')
elif node[1]['type']=='GO':
    colors.append('lightgreen')
else:
    colors.append('salmon')

plt.figure(figsize=(12,8))
nx.draw(G, pos, with_labels=True, node_color=colors, node_size=1500, font_si
plt.title("Proteins, GO terms, and Synonyms Interrelation")
plt.show()
```

Proteins, GO terms, and Synonyms Interrelation

