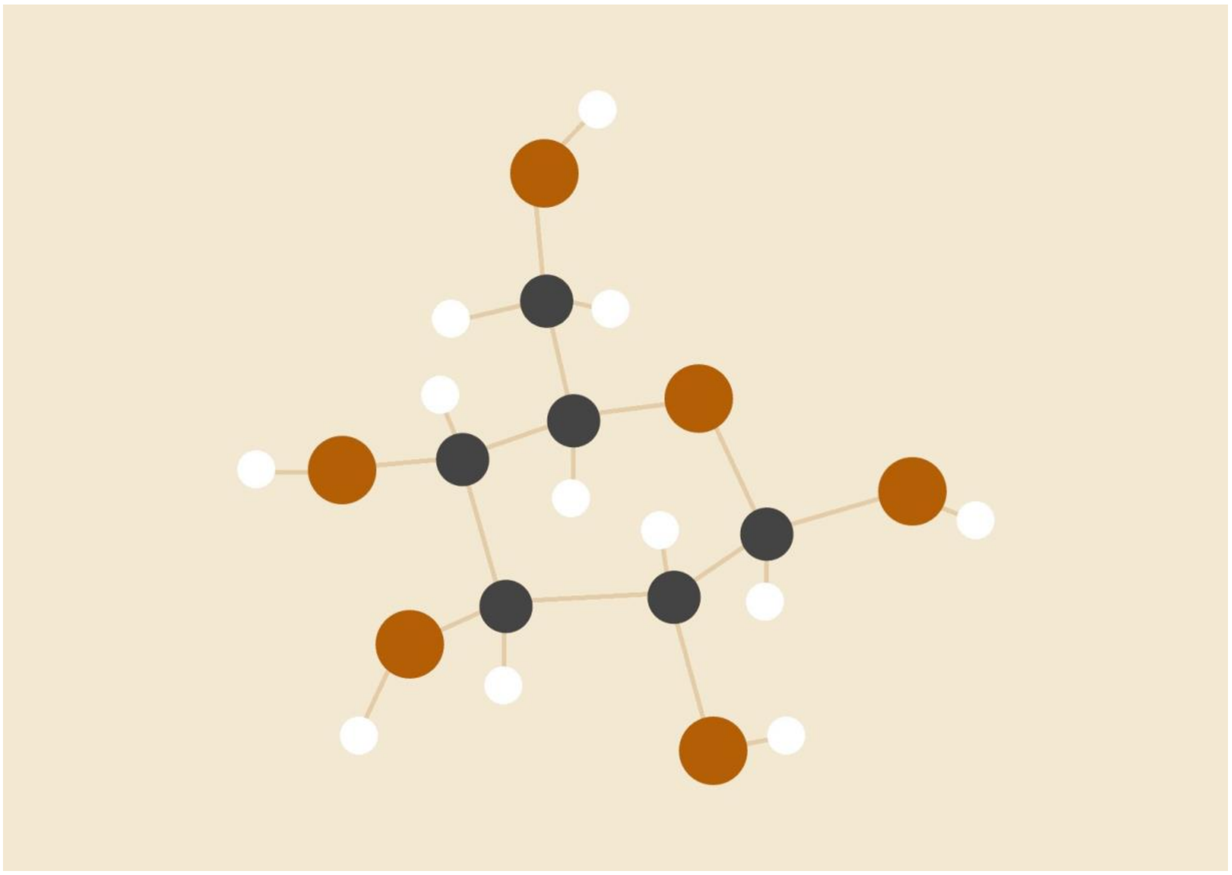
PROJECT REPORT

Comic Characters Ontology



Nadja Näf

Elina Stüssi,

Stylianos Psychias,

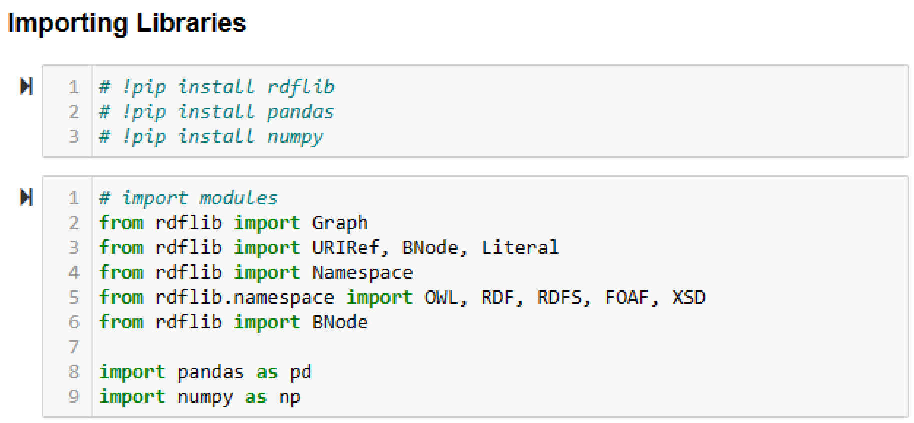
Florian Heinz

ABSTRACT

This report presents the development of an ontology for comic book characters, their details, the comics they appear in, the species they belong to, and the publishers involved. By consolidating multiple datasets and utilizing Python's rdflib library, we created a comprehensive knowledge graph that captures the relationships and properties within the comic book domain. The ontology facilitates efficient organization and querying through SPARQL, allowing users to explore and retrieve valuable information about comic book entities.

1. DATASET

An extensive code was written in Python for mapping the provided dataset to the base ontology. First the necessary libraries were imported.



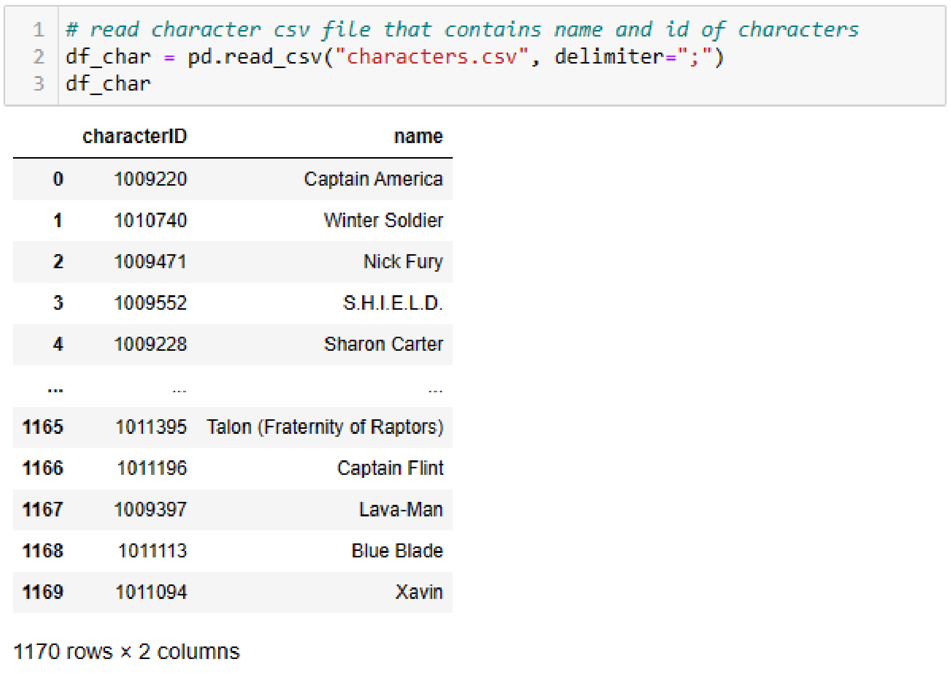
Importing Libraries

```python
# !pip install rdflib
# !pip install pandas
# !pip install numpy
```

```python
# import modules
from rdflib import Graph
from rdflib import URIRef, BNode, Literal
from rdflib import Namespace
from rdflib.namespace import OWL, RDF, RDFS, FOAF, XSD
from rdflib import BNode

import pandas as pd
import numpy as np
```

Figure 1: importing libraries

Multiple datasets namely "characters.csv" (see Figure 2), “charactersToComics.csv”(see Figure 3), “comics.csv” (see Figure 4) and “marvel_characters_info_with_external_resource.csv” (see Figure 5) were combined into a single dataset.

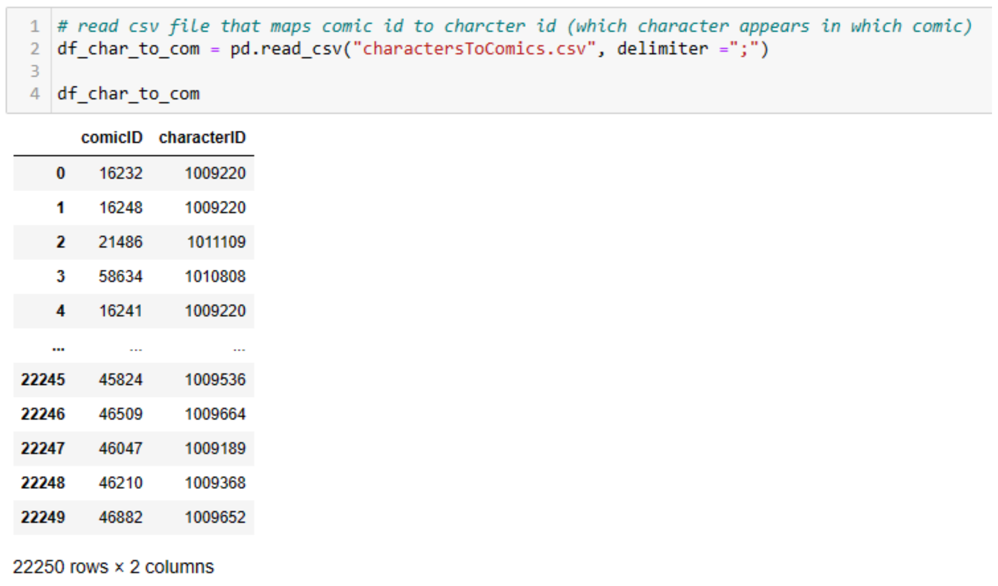


```python
# read character csv file that contains name and id of characters
df_char = pd.read_csv("characters.csv", delimiter=";")
df_char
```

	characterID	name
0	1009220	Captain America
1	1010740	Winter Soldier
2	1009471	Nick Fury
3	1009552	S.H.I.E.L.D.
4	1009228	Sharon Carter
...	...	...
1165	1011395	Talon (Fraternity of Raptors)
1166	1011196	Captain Flint
1167	1009397	Lava-Man
1168	1011113	Blue Blade
1169	1011094	Xavin

1170 rows × 2 columns

Figure 2: characters.csv

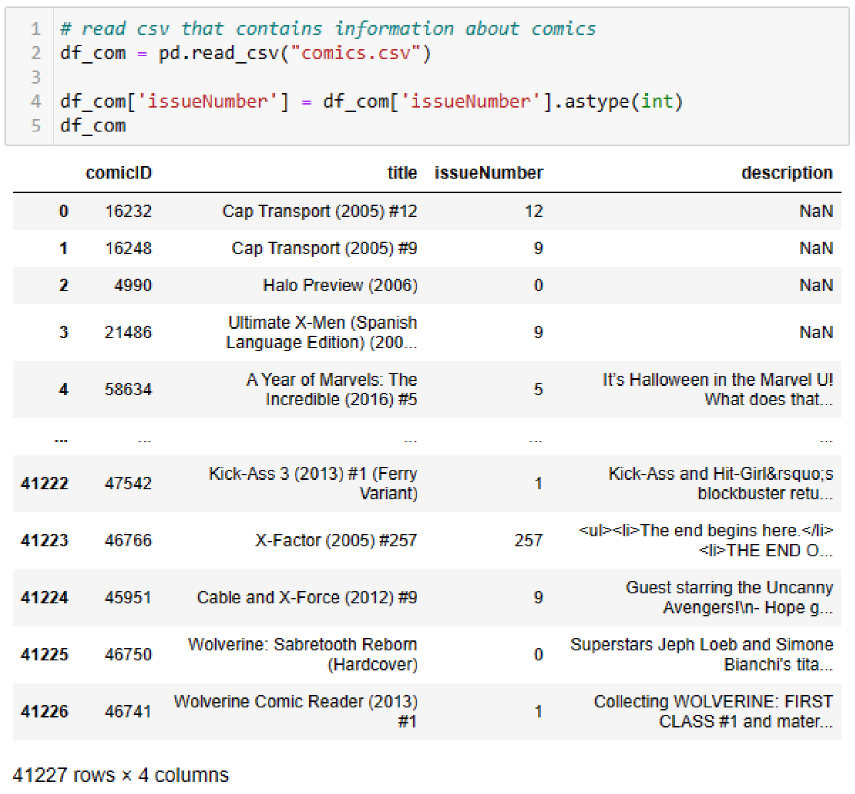


```python
# read csv file that maps comic id to charcter id (which character appears in which comic)
df_char_to_com = pd.read_csv("charactersToComics.csv", delimiter =";")

df_char_to_com
```

	comicID	characterID
0	16232	1009220
1	16248	1009220
2	21486	1011109
3	58634	1010808
4	16241	1009220
...	...	...
22245	45824	1009536
22246	46509	1009664
22247	46047	1009189
22248	46210	1009368
22249	46882	1009652

22250 rows × 2 columns

Figure 3: charactersToComics.csv



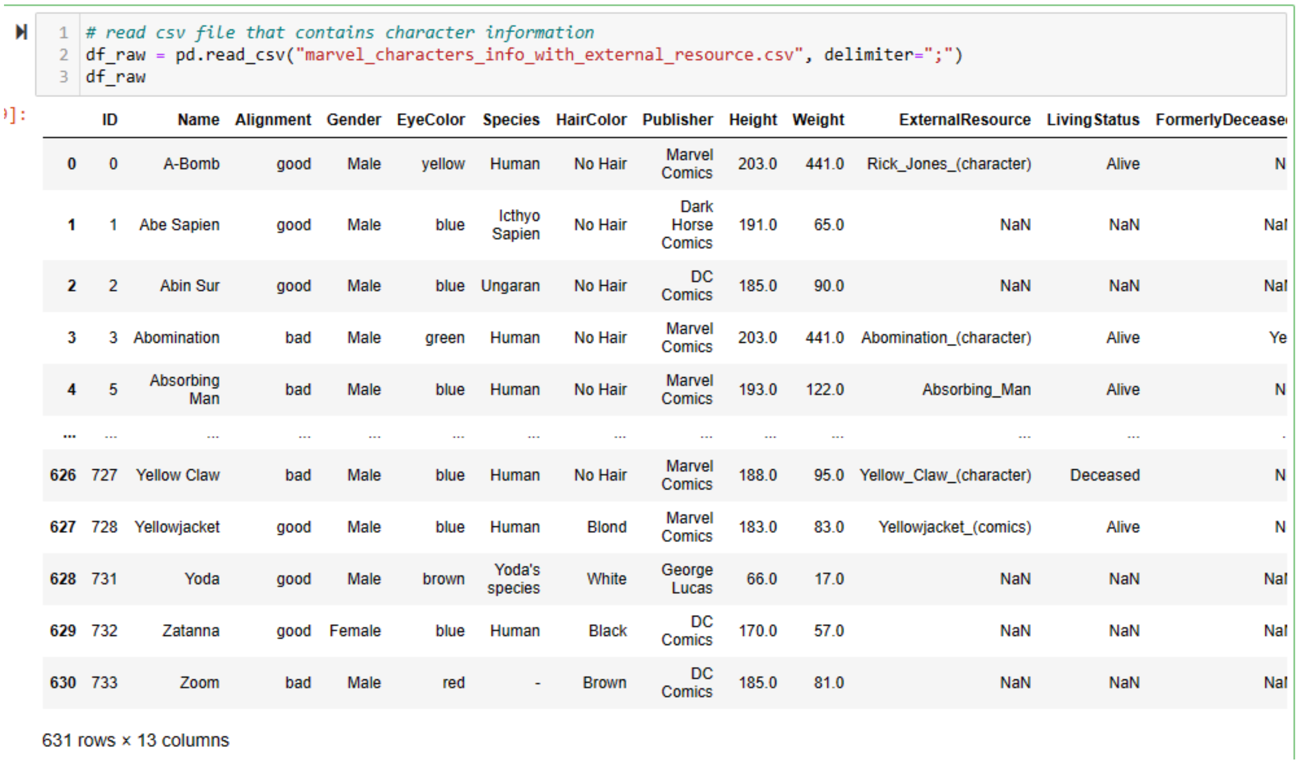
```python
# read csv that contains information about comics
df_com = pd.read_csv("comics.csv")

df_com['issueNumber'] = df_com['issueNumber'].astype(int)
df_com
```

	comicID	title	issueNumber	description
0	16232	Cap Transport (2005) #12	12	NaN
1	16248	Cap Transport (2005) #9	9	NaN
2	4990	Halo Preview (2006)	0	NaN
3	21486	Ultimate X-Men (Spanish Language Edition) (200...	9	NaN
4	58634	A Year of Marvels: The Incredible (2016) #5	5	It's Halloween in the Marvel U! What does that...
...	...	...	...	...
41222	47542	Kick-Ass 3 (2013) #1 (Ferry Variant)	1	Kick-Ass and Hit-Girl&rsquo;s blockbuster retu...
41223	46766	X-Factor (2005) #257	257	<ul><li>The end begins here.</li><li>THE END O...
41224	45951	Cable and X-Force (2012) #9	9	Guest starring the Uncanny Avengers!\n- Hope g...
41225	46750	Wolverine: Sabretooth Reborn (Hardcover)	0	Superstars Jeph Loeb and Simone Bianchi's tita...
41226	46741	Wolverine Comic Reader (2013) #1	1	Collecting WOLVERINE: FIRST CLASS #1 and mater...

41227 rows × 4 columns

Figure 4: comics.csv

Loading Dataset



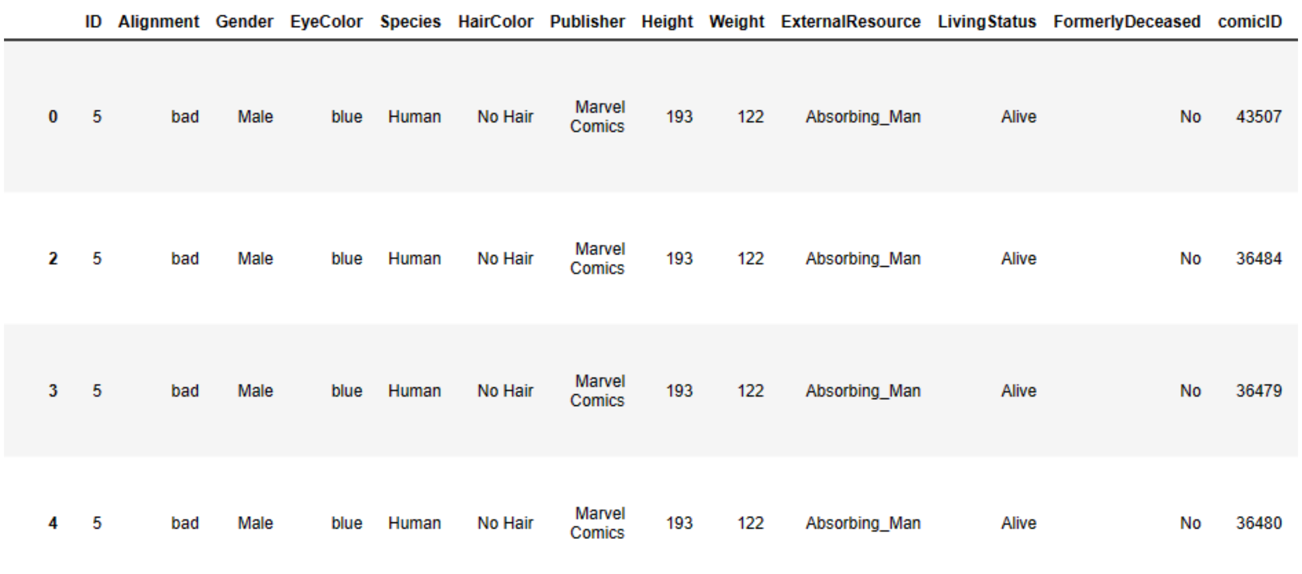
```python
# read csv file that contains character information
df_raw = pd.read_csv("marvel_characters_info_with_external_resource.csv", delimiter=";")
df_raw
```

	ID	Name	Alignment	Gender	EyeColor	Species	HairColor	Publisher	Height	Weight	ExternalResource	LivingStatus	FormerlyDecease
0	0	A-Bomb	good	Male	yellow	Human	No Hair	Marvel Comics	203.0	441.0	Rick_Jones_(character)	Alive	N
1	1	Abe Sapien	good	Male	blue	Icthyo Sapien	No Hair	Dark Horse Comics	191.0	65.0	NaN	NaN	Nal
2	2	Abin Sur	good	Male	blue	Ungaran	No Hair	DC Comics	185.0	90.0	NaN	NaN	Nal
3	3	Abomination	bad	Male	green	Human	No Hair	Marvel Comics	203.0	441.0	Abomination_(character)	Alive	Ye
4	5	Absorbing Man	bad	Male	blue	Human	No Hair	Marvel Comics	193.0	122.0	Absorbing_Man	Alive	N
...	...	...	...	...	...	...	...	...	...	...	...	...	.
626	727	Yellow Claw	bad	Male	blue	Human	No Hair	Marvel Comics	188.0	95.0	Yellow_Claw_(character)	Deceased	N
627	728	Yellowjacket	good	Male	blue	Human	Blond	Marvel Comics	183.0	83.0	Yellowjacket_(comics)	Alive	N
628	731	Yoda	good	Male	brown	Yoda's species	White	George Lucas	66.0	17.0	NaN	NaN	Nal
629	732	Zatanna	good	Female	blue	Human	Black	DC Comics	170.0	57.0	NaN	NaN	Nal
630	733	Zoom	bad	Male	red	-	Brown	DC Comics	185.0	81.0	NaN	NaN	Nal

631 rows × 13 columns

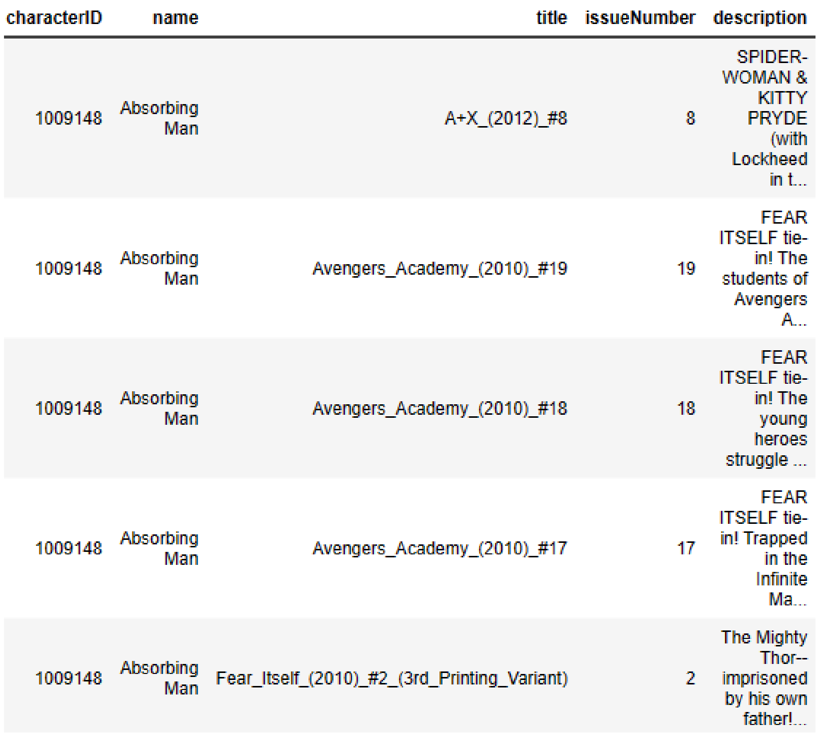
Figure 5: marvel_characters_info_with_external_resource.csv

After the datasets were joined into one dataset. Then some data cleaning was performed where NULL values were replaced, and data types were made consistent (see Figure 6-7).



	ID	Alignment	Gender	EyeColor	Species	HairColor	Publisher	Height	Weight	ExternalResource	LivingStatus	FormerlyDeceased	comicID
0	5	bad	Male	blue	Human	No Hair	Marvel Comics	193	122	Absorbing_Man	Alive	No	43507
2	5	bad	Male	blue	Human	No Hair	Marvel Comics	193	122	Absorbing_Man	Alive	No	36484
3	5	bad	Male	blue	Human	No Hair	Marvel Comics	193	122	Absorbing_Man	Alive	No	36479
4	5	bad	Male	blue	Human	No Hair	Marvel Comics	193	122	Absorbing_Man	Alive	No	36480

Figure 6: summarized and cleaned dataframe until column comicID

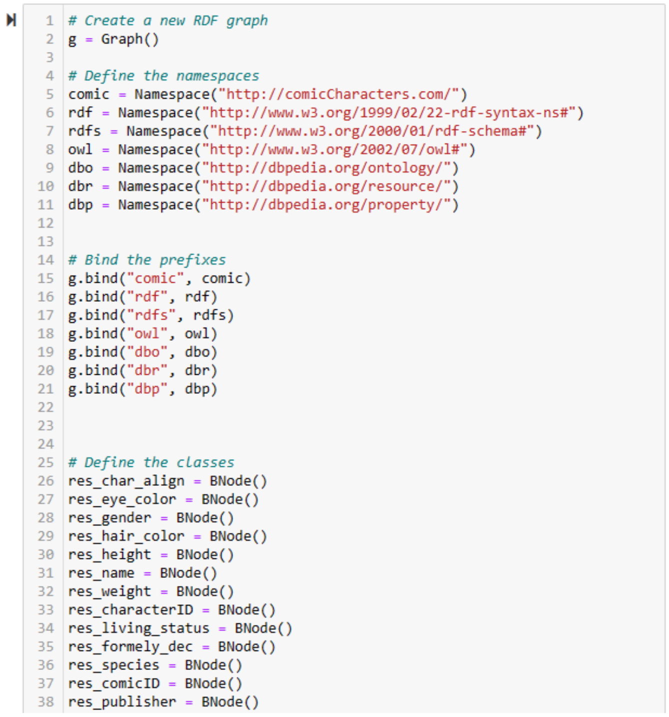


characterID	name	title	issueNumber	description
1009148	Absorbing Man	A+X_(2012)_#8	8	SPIDER-WOMAN & KITTY PRYDE (with Lockheed in t...
1009148	Absorbing Man	Avengers_Academy_(2010)_#19	19	FEAR ITSELF tie-in! The students of Avengers A...
1009148	Absorbing Man	Avengers_Academy_(2010)_#18	18	FEAR ITSELF tie-in! The young heroes struggle ...
1009148	Absorbing Man	Avengers_Academy_(2010)_#17	17	FEAR ITSELF tie-in! Trapped in the Infinite Ma...
1009148	Absorbing Man	Fear_Itself_(2010)_#2_(3rd_Printing_Variant)	2	The Mighty Thor--imprisoned by his own father!...

Figure 7: summarized and cleaned dataframe from comicID until end

2. ONTOLOGY CREATION

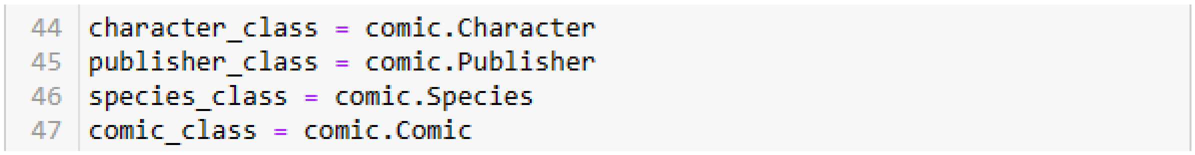
The code starts by importing the required libraries and creating a new RDF graph using the "Graph()" function. Additionally, namespaces are defined using the "Namespace()" function. Namespaces are used to define the prefixes for URIs in the ontology. In this code, several namespaces are defined which can be seen in Figure 8, line 5 – 11.



```
 1  # Create a new RDF graph
 2  g = Graph()
 3
 4  # Define the namespaces
 5  comic = Namespace("http://comicCharacters.com/")
 6  rdf = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
 7  rdfs = Namespace("http://www.w3.org/2000/01/rdf-schema#")
 8  owl = Namespace("http://www.w3.org/2002/07/owl#")
 9  dbo = Namespace("http://dbpedia.org/ontology/")
10  dbr = Namespace("http://dbpedia.org/resource/")
11  dbp = Namespace("http://dbpedia.org/property/")
12
13
14  # Bind the prefixes
15  g.bind("comic", comic)
16  g.bind("rdf", rdf)
17  g.bind("rdfs", rdfs)
18  g.bind("owl", owl)
19  g.bind("dbo", dbo)
20  g.bind("dbr", dbr)
21  g.bind("dbp", dbp)
22
23
24
25  # Define the classes
26  res_char_align = BNode()
27  res_eye_color = BNode()
28  res_gender = BNode()
29  res_hair_color = BNode()
30  res_height = BNode()
31  res_name = BNode()
32  res_weight = BNode()
33  res_characterID = BNode()
34  res_living_status = BNode()
35  res_formely_dec = BNode()
36  res_species = BNode()
37  res_comicID = BNode()
38  res_publisher = BNode()
```

Figure 8: Ontology Creation

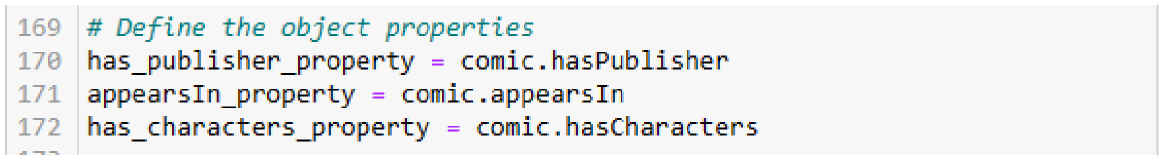
Following this, Class definitions are created using the “comic” prefix. Four classes are defined: “Character”, “Species”, “Publisher”, and “Comic” (see Figure 9).



```python
character_class = comic.Character
publisher_class = comic.Publisher
species_class = comic.Species
comic_class = comic.Comic
```

Figure 9: classes

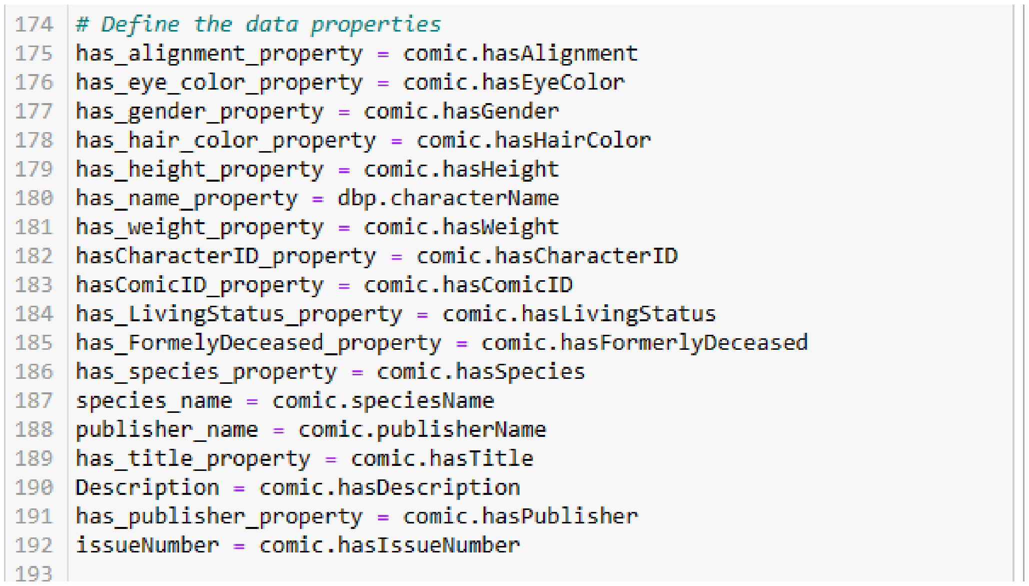
Furthermore, object properties are defined using the “comic” prefix. Three object properties are defined: “hasPublisher”, “hasCharacters”, “hasSpecies” and “appearsIn”. These properties define the relationships between entities in the ontology (see Figure 10).



```python
# Define the object properties
has_publisher_property = comic.hasPublisher
appearsIn_property = comic.appearsIn
has_characters_property = comic.hasCharacters
```

Figure 10: object properties

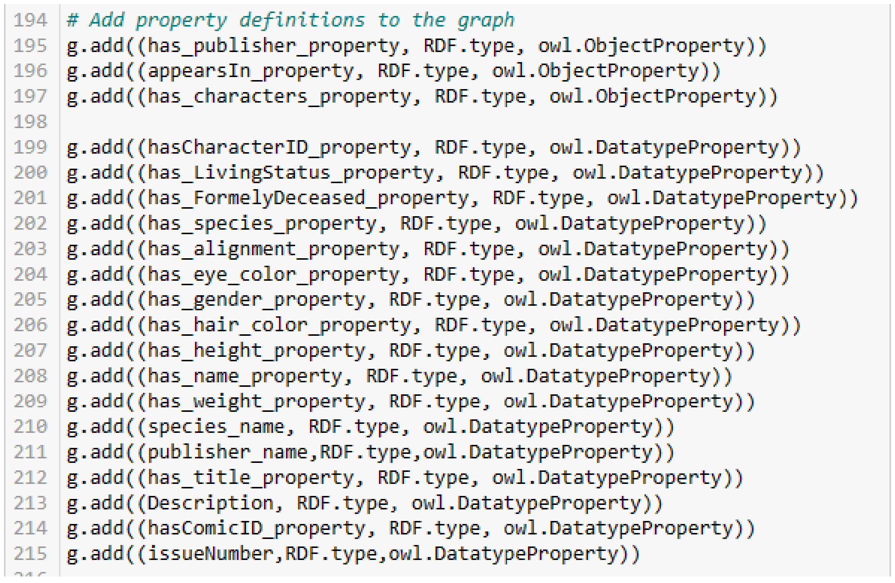
Afterwards, data properties are defined using the “comic.” prefix. Several data properties are defined, such as “has_alignment_property”, “has_eye_color_property”, ”has_gender_property”, and so on. These properties are used to define attributes or characteristics of entities in the ontology (see Figure 11).



```python
# Define the data properties
has_alignment_property = comic.hasAlignment
has_eye_color_property = comic.hasEyeColor
has_gender_property = comic.hasGender
has_hair_color_property = comic.hasHairColor
has_height_property = comic.hasHeight
has_name_property = dbp.characterName
has_weight_property = comic.hasWeight
hasCharacterID_property = comic.hasCharacterID
hasComicID_property = comic.hasComicID
has_LivingStatus_property = comic.hasLivingStatus
has_FormelyDeceased_property = comic.hasFormerlyDeceased
has_species_property = comic.hasSpecies
species_name = comic.speciesName
publisher_name = comic.publisherName
has_title_property = comic.hasTitle
Description = comic.hasDescription
has_publisher_property = comic.hasPublisher
issueNumber = comic.hasIssueNumber

```

Figure 11: data properties

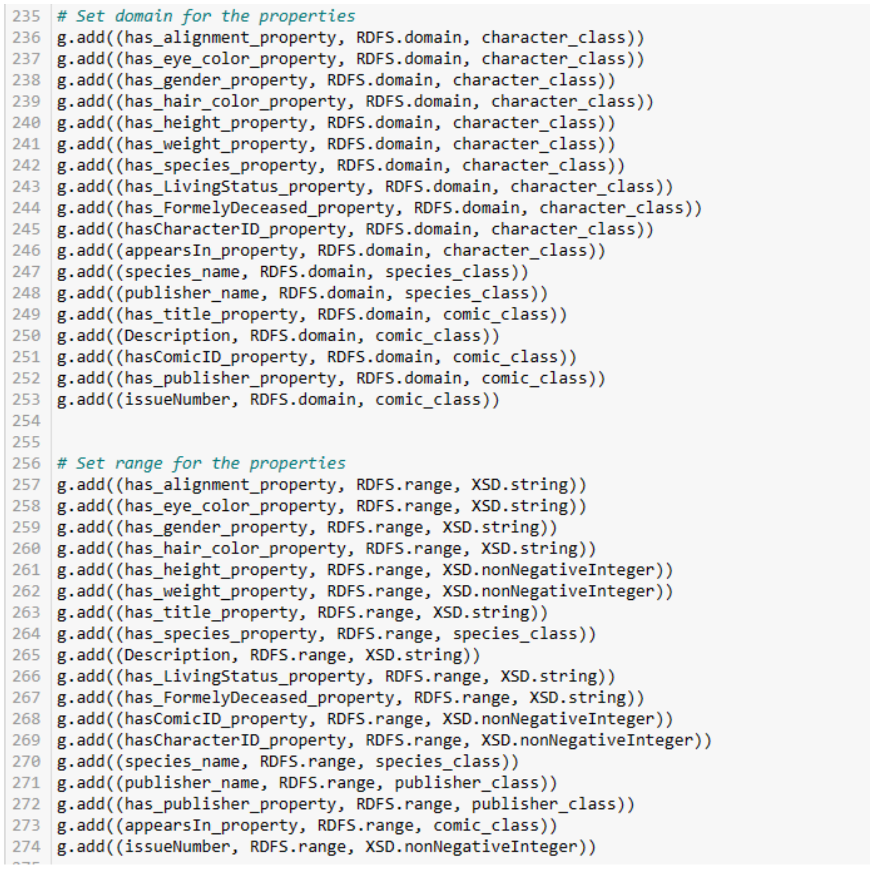
Moreover, class and property definitions are added to the graph using the “g.add()” function. Each definition is added as a triple with the subject, predicate, and object (see Figure 12).



```python
# Add property definitions to the graph
g.add((has_publisher_property, RDF.type, owl.ObjectProperty))
g.add((appearsIn_property, RDF.type, owl.ObjectProperty))
g.add((has_characters_property, RDF.type, owl.ObjectProperty))

g.add((hasCharacterID_property, RDF.type, owl.DatatypeProperty))
g.add((has_LivingStatus_property, RDF.type, owl.DatatypeProperty))
g.add((has_FormelyDeceased_property, RDF.type, owl.DatatypeProperty))
g.add((has_species_property, RDF.type, owl.DatatypeProperty))
g.add((has_alignment_property, RDF.type, owl.DatatypeProperty))
g.add((has_eye_color_property, RDF.type, owl.DatatypeProperty))
g.add((has_gender_property, RDF.type, owl.DatatypeProperty))
g.add((has_hair_color_property, RDF.type, owl.DatatypeProperty))
g.add((has_height_property, RDF.type, owl.DatatypeProperty))
g.add((has_name_property, RDF.type, owl.DatatypeProperty))
g.add((has_weight_property, RDF.type, owl.DatatypeProperty))
g.add((species_name, RDF.type, owl.DatatypeProperty))
g.add((publisher_name,RDF.type,owl.DatatypeProperty))
g.add((has_title_property, RDF.type, owl.DatatypeProperty))
g.add((Description, RDF.type, owl.DatatypeProperty))
g.add((hasComicID_property, RDF.type, owl.DatatypeProperty))
g.add((issueNumber,RDF.type,owl.DatatypeProperty))
```

Figure 12: property definition

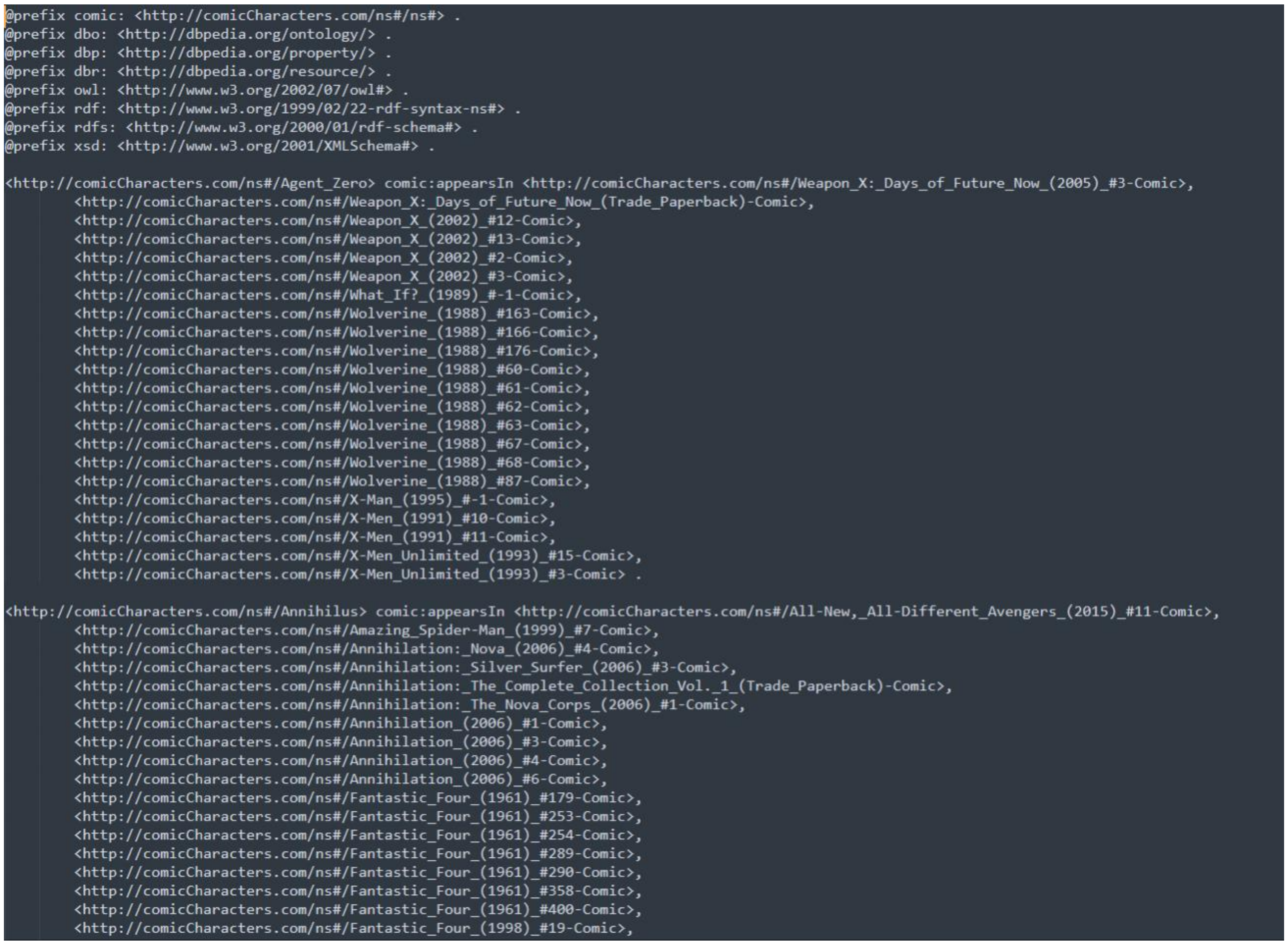
Additionally, domain and range restrictions are set for the properties using the "RDFS.domain" and "RDFS.range" predicates. These restrictions define which classes the properties can be applied to and what types of values they can have (see Figure 13).



```python
# Set domain for the properties
g.add((has_alignment_property, RDFS.domain, character_class))
g.add((has_eye_color_property, RDFS.domain, character_class))
g.add((has_gender_property, RDFS.domain, character_class))
g.add((has_hair_color_property, RDFS.domain, character_class))
g.add((has_height_property, RDFS.domain, character_class))
g.add((has_weight_property, RDFS.domain, character_class))
g.add((has_species_property, RDFS.domain, character_class))
g.add((has_LivingStatus_property, RDFS.domain, character_class))
g.add((has_FormelyDeceased_property, RDFS.domain, character_class))
g.add((hasCharacterID_property, RDFS.domain, character_class))
g.add((appearsIn_property, RDFS.domain, character_class))
g.add((species_name, RDFS.domain, species_class))
g.add((publisher_name, RDFS.domain, species_class))
g.add((has_title_property, RDFS.domain, comic_class))
g.add((Description, RDFS.domain, comic_class))
g.add((hasComicID_property, RDFS.domain, comic_class))
g.add((has_publisher_property, RDFS.domain, comic_class))
g.add((issueNumber, RDFS.domain, comic_class))


# Set range for the properties
g.add((has_alignment_property, RDFS.range, XSD.string))
g.add((has_eye_color_property, RDFS.range, XSD.string))
g.add((has_gender_property, RDFS.range, XSD.string))
g.add((has_hair_color_property, RDFS.range, XSD.string))
g.add((has_height_property, RDFS.range, XSD.nonNegativeInteger))
g.add((has_weight_property, RDFS.range, XSD.nonNegativeInteger))
g.add((has_title_property, RDFS.range, XSD.string))
g.add((has_species_property, RDFS.range, species_class))
g.add((Description, RDFS.range, XSD.string))
g.add((has_LivingStatus_property, RDFS.range, XSD.string))
g.add((has_FormelyDeceased_property, RDFS.range, XSD.string))
g.add((hasComicID_property, RDFS.range, XSD.nonNegativeInteger))
g.add((hasCharacterID_property, RDFS.range, XSD.nonNegativeInteger))
g.add((species_name, RDFS.range, species_class))
g.add((publisher_name, RDFS.range, publisher_class))
g.add((has_publisher_property, RDFS.range, publisher_class))
g.add((appearsIn_property, RDFS.range, comic_class))
g.add((issueNumber, RDFS.range, XSD.nonNegativeInteger))
```

Figure 13: domains and ranges

Finally, the graph is serialized and saved to disk as a Turtle file format (.ttl) with the name "comicChar.owl" (see Figure 14).



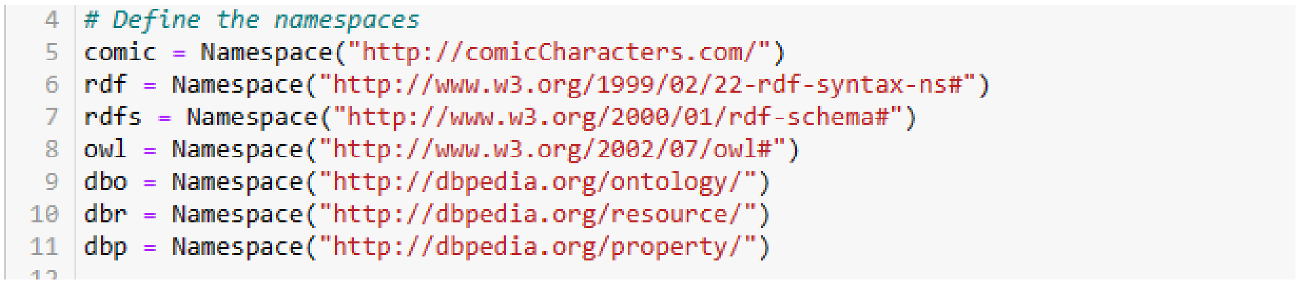
```
@prefix comic: <http://comicCharacters.com/ns#/ns#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://comicCharacters.com/ns#/Agent_Zero> comic:appearsIn <http://comicCharacters.com/ns#/Weapon_X:_Days_of_Future_Now_(2005)_#3-Comic>,
        <http://comicCharacters.com/ns#/Weapon_X:_Days_of_Future_Now_(Trade_Paperback)-Comic>,
        <http://comicCharacters.com/ns#/Weapon_X_(2002)_#12-Comic>,
        <http://comicCharacters.com/ns#/Weapon_X_(2002)_#13-Comic>,
        <http://comicCharacters.com/ns#/Weapon_X_(2002)_#2-Comic>,
        <http://comicCharacters.com/ns#/Weapon_X_(2002)_#3-Comic>,
        <http://comicCharacters.com/ns#/What_If?_(1989)_#-1-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#163-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#166-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#176-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#60-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#61-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#62-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#63-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#67-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#68-Comic>,
        <http://comicCharacters.com/ns#/Wolverine_(1988)_#87-Comic>,
        <http://comicCharacters.com/ns#/X-Man_(1995)_#-1-Comic>,
        <http://comicCharacters.com/ns#/X-Men_(1991)_#10-Comic>,
        <http://comicCharacters.com/ns#/X-Men_(1991)_#11-Comic>,
        <http://comicCharacters.com/ns#/X-Men_Unlimited_(1993)_#15-Comic>,
        <http://comicCharacters.com/ns#/X-Men_Unlimited_(1993)_#3-Comic> .

<http://comicCharacters.com/ns#/Annihilus> comic:appearsIn <http://comicCharacters.com/ns#/All-New,_All-Different_Avengers_(2015)_#11-Comic>,
        <http://comicCharacters.com/ns#/Amazing_Spider-Man_(1999)_#7-Comic>,
        <http://comicCharacters.com/ns#/Annihilation:_Nova_(2006)_#4-Comic>,
        <http://comicCharacters.com/ns#/Annihilation:_Silver_Surfer_(2006)_#3-Comic>,
        <http://comicCharacters.com/ns#/Annihilation:_The_Complete_Collection_Vol._1_(Trade_Paperback)-Comic>,
        <http://comicCharacters.com/ns#/Annihilation:_The_Nova_Corps_(2006)_#1-Comic>,
        <http://comicCharacters.com/ns#/Annihilation_(2006)_#1-Comic>,
        <http://comicCharacters.com/ns#/Annihilation_(2006)_#3-Comic>,
        <http://comicCharacters.com/ns#/Annihilation_(2006)_#4-Comic>,
        <http://comicCharacters.com/ns#/Annihilation_(2006)_#6-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#179-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#253-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#254-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#289-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#290-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#358-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1961)_#400-Comic>,
        <http://comicCharacters.com/ns#/Fantastic_Four_(1998)_#19-Comic>,
```

Figure 14: mappedOntology

3. MAPPING DATASET TO ONTOLOGY

The provided code is a function called "createTriples()" that performs ontology mapping. Let's go through the code step by step:

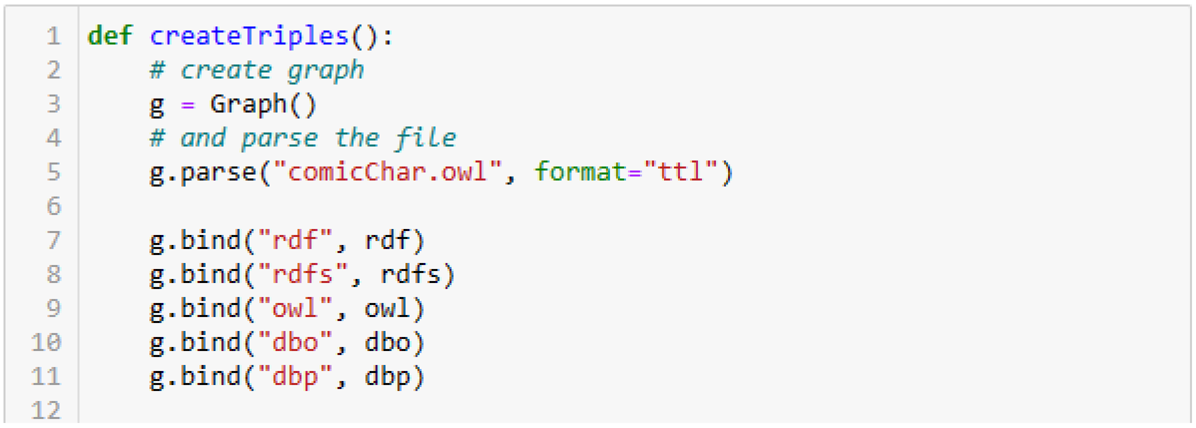
It starts by creating a new instance of the "Graph" class from the "rdflib" library. This graph will be used to represent the ontology and store the triples It parses an input ontology file named "comicChar.owl" in Turtle format and adds its content to the graph. The "g.parse()" function is used for parsing and loading the ontology into the graph. Next, it uses the previously defined namespaces with "rdflib". These namespaces are used to create compact URIs for the ontology concepts. We defined the following namespaces:



```python
# Define the namespaces
comic = Namespace("http://comicCharacters.com/")
rdf = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
rdfs = Namespace("http://www.w3.org/2000/01/rdf-schema#")
owl = Namespace("http://www.w3.org/2002/07/owl#")
dbo = Namespace("http://dbpedia.org/ontology/")
dbr = Namespace("http://dbpedia.org/resource/")
dbp = Namespace("http://dbpedia.org/property/")
```

Figure 15: namespaces

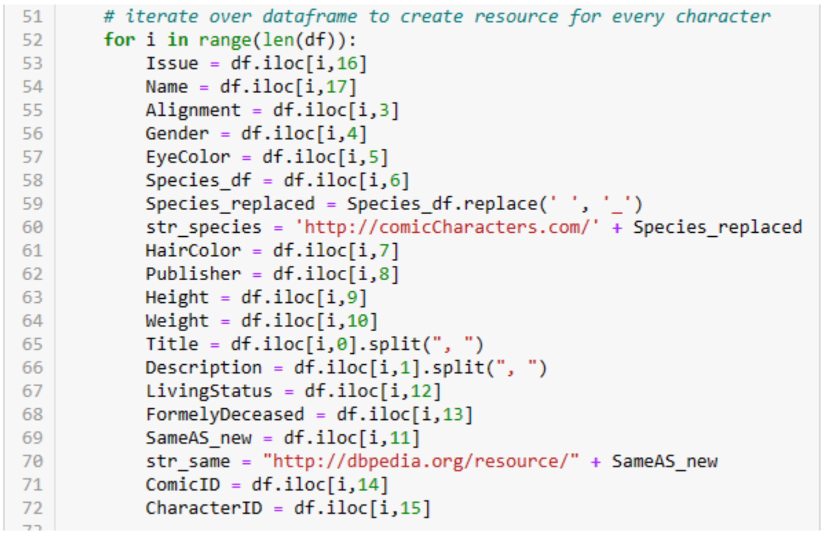
It binds the defined namespaces to their respective prefixes using the "g.bind()" function. This allows using the prefixes instead of the full URIs in the triple statements. Whenever the whole URI is used this is done to escape special characters. A loop is started to iterate over a DataFrame ("df") to extract information about comic book characters (see Figure 16).



```python
def createTriples():
    # create graph
    g = Graph()
    # and parse the file
    g.parse("comicChar.owl", format="ttl")

    g.bind("rdf", rdf)
    g.bind("rdfs", rdfs)
    g.bind("owl", owl)
    g.bind("dbo", dbo)
    g.bind("dbp", dbp)

```

Figure 16: creation of createTriples function

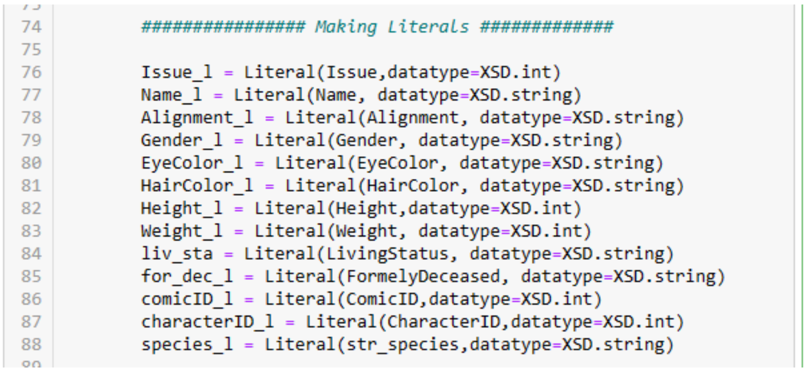
Inside the loop, various attributes of the comic book characters are extracted from the DataFrame (see Figure 17).



```
    # iterate over dataframe to create resource for every character
    for i in range(len(df)):
        Issue = df.iloc[i,16]
        Name = df.iloc[i,17]
        Alignment = df.iloc[i,3]
        Gender = df.iloc[i,4]
        EyeColor = df.iloc[i,5]
        Species_df = df.iloc[i,6]
        Species_replaced = Species_df.replace(' ', '_')
        str_species = 'http://comicCharacters.com/' + Species_replaced
        HairColor = df.iloc[i,7]
        Publisher = df.iloc[i,8]
        Height = df.iloc[i,9]
        Weight = df.iloc[i,10]
        Title = df.iloc[i,0].split(", ")
        Description = df.iloc[i,1].split(", ")
        LivingStatus = df.iloc[i,12]
        FormelyDeceased = df.iloc[i,13]
        SameAS_new = df.iloc[i,11]
        str_same = "http://dbpedia.org/resource/" + SameAS_new
        ComicID = df.iloc[i,14]
        CharacterID = df.iloc[i,15]
```

Figure 17: looping through the characters

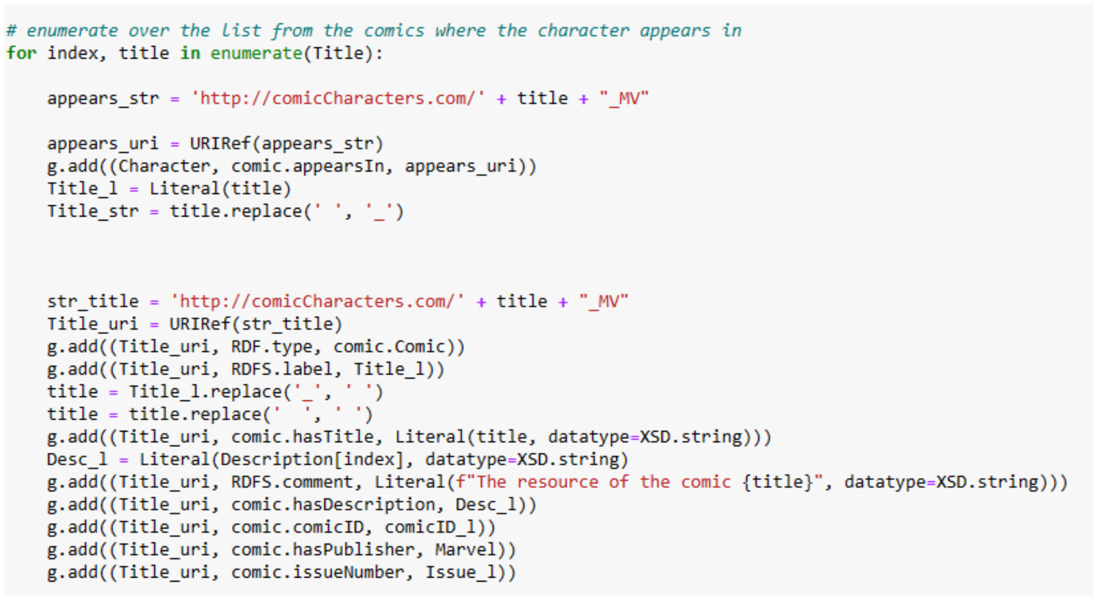
Literal values are created for each attribute using the “Literal” class from “rdflib”. These literals are used to represent attribute values in the ontology (see Figure 18).



```
        ################ Making Literals #############

        Issue_l = Literal(Issue,datatype=XSD.int)
        Name_l = Literal(Name, datatype=XSD.string)
        Alignment_l = Literal(Alignment, datatype=XSD.string)
        Gender_l = Literal(Gender, datatype=XSD.string)
        EyeColor_l = Literal(EyeColor, datatype=XSD.string)
        HairColor_l = Literal(HairColor, datatype=XSD.string)
        Height_l = Literal(Height,datatype=XSD.int)
        Weight_l = Literal(Weight, datatype=XSD.int)
        liv_sta = Literal(LivingStatus, datatype=XSD.string)
        for_dec_l = Literal(FormelyDeceased, datatype=XSD.string)
        comicID_l = Literal(ComicID,datatype=XSD.int)
        characterID_l = Literal(CharacterID,datatype=XSD.int)
        species_l = Literal(str_species,datatype=XSD.string)
```

Figure 18: making literals

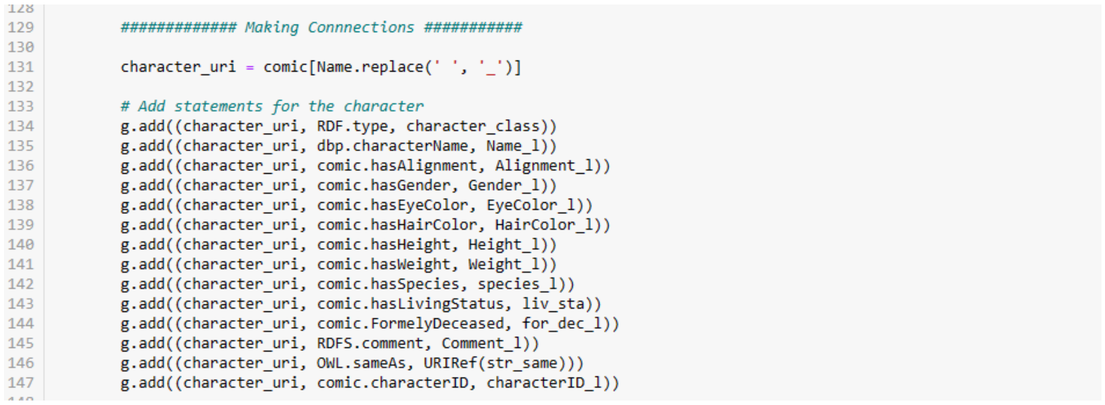
Some preprocessing is done on the Name, Publisher, and Title attributes to replace spaces with underscores and create valid URIs. Furthermore, URIs are created using the “URIRef” class from “rdflib” for the Character, Publisher, and Title (see Figure 19).



```python
# enumerate over the list from the comics where the character appears in
for index, title in enumerate(Title):

    appears_str = 'http://comicCharacters.com/' + title + "_MV"

    appears_uri = URIRef(appears_str)
    g.add((Character, comic.appearsIn, appears_uri))
    Title_l = Literal(title)
    Title_str = title.replace(' ', '_')



    str_title = 'http://comicCharacters.com/' + title + "_MV"
    Title_uri = URIRef(str_title)
    g.add((Title_uri, RDF.type, comic.Comic))
    g.add((Title_uri, RDFS.label, Title_l))
    title = Title_l.replace('_', ' ')
    title = title.replace('  ', ' ')
    g.add((Title_uri, comic.hasTitle, Literal(title, datatype=XSD.string)))
    Desc_l = Literal(Description[index], datatype=XSD.string)
    g.add((Title_uri, RDFS.comment, Literal(f"The resource of the comic {title}", datatype=XSD.string)))
    g.add((Title_uri, comic.hasDescription, Desc_l))
    g.add((Title_uri, comic.comicID, comicID_l))
    g.add((Title_uri, comic.hasPublisher, Marvel))
    g.add((Title_uri, comic.issueNumber, Issue_l))
```

Figure 19: replace spaces

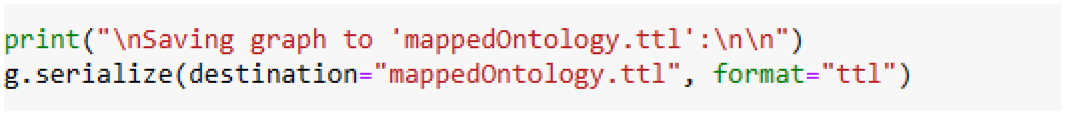
Various triple statements are then added to the graph ("g.add()") to represent the relationships and attributes of the comic book characters, the publisher, and the title (see Figure 20).



```python
        ############# Making Connnections ###########

        character_uri = comic[Name.replace(' ', '_')]

        # Add statements for the character
        g.add((character_uri, RDF.type, character_class))
        g.add((character_uri, dbp.characterName, Name_l))
        g.add((character_uri, comic.hasAlignment, Alignment_l))
        g.add((character_uri, comic.hasGender, Gender_l))
        g.add((character_uri, comic.hasEyeColor, EyeColor_l))
        g.add((character_uri, comic.hasHairColor, HairColor_l))
        g.add((character_uri, comic.hasHeight, Height_l))
        g.add((character_uri, comic.hasWeight, Weight_l))
        g.add((character_uri, comic.hasSpecies, species_l))
        g.add((character_uri, comic.hasLivingStatus, liv_sta))
        g.add((character_uri, comic.FormelyDeceased, for_dec_l))
        g.add((character_uri, RDFS.comment, Comment_l))
        g.add((character_uri, OWL.sameAs, URIRef(str_same)))
        g.add((character_uri, comic.characterID, characterID_l))
```

Figure 20: statemants for character

After the loop finishes, the graph is serialized to a file named "mappedOntology.owl" in Turtle format using the "g.serialize()" function. Additionally, a message is printed to indicate that the graph has been saved. (see Figure 21).

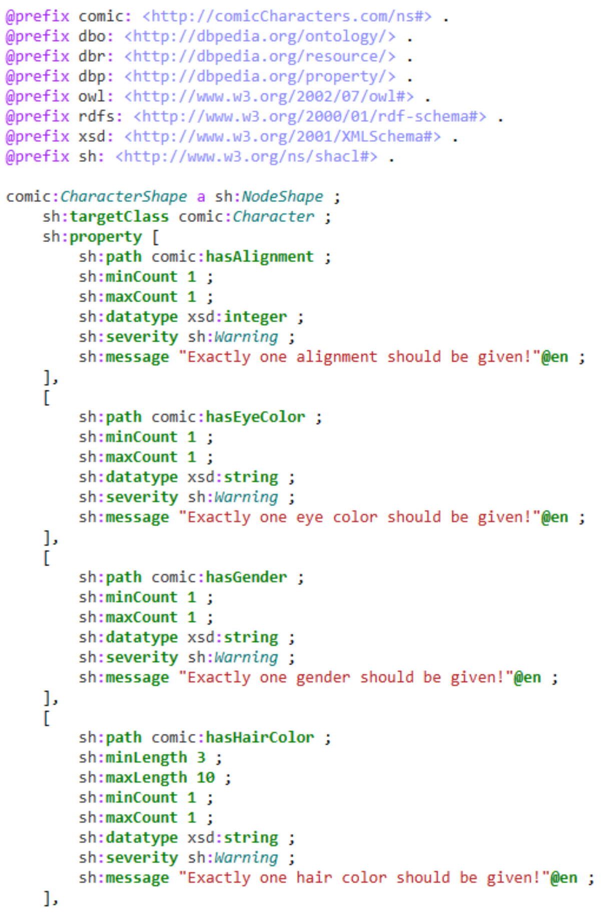


```python
print("\nSaving graph to 'mappedOntology.ttl':\n\n")
g.serialize(destination="mappedOntology.ttl", format="ttl")
```

Figure 21: serialization

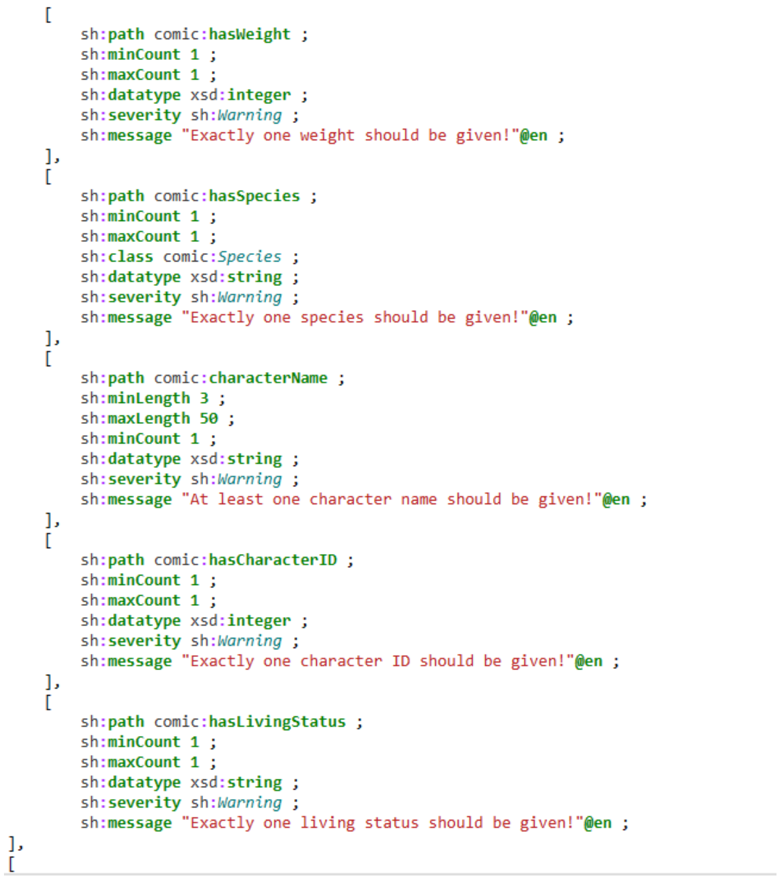
4. SHACL

In order to test the given restrictions, a Shacl file was added with noteworthy restrictions on the predicates. First and foremost, empty nodes were created to which the restrictions were added. Restrictions, as shown in Figure 22-23, include the (min/max) count, the datatype, length, and warning messages.



```
@prefix comic: <http://comicCharacters.com/ns#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

comic:CharacterShape a sh:NodeShape ;
    sh:targetClass comic:Character ;
    sh:property [
        sh:path comic:hasAlignment ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:integer ;
        sh:severity sh:Warning ;
        sh:message "Exactly one alignment should be given!"@en ;
    ],
    [
        sh:path comic:hasEyeColor ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "Exactly one eye color should be given!"@en ;
    ],
    [
        sh:path comic:hasGender ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "Exactly one gender should be given!"@en ;
    ],
    [
        sh:path comic:hasHairColor ;
        sh:minLength 3 ;
        sh:maxLength 10 ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "Exactly one hair color should be given!"@en ;
    ],
```

Figure 22: Shacl, part 1



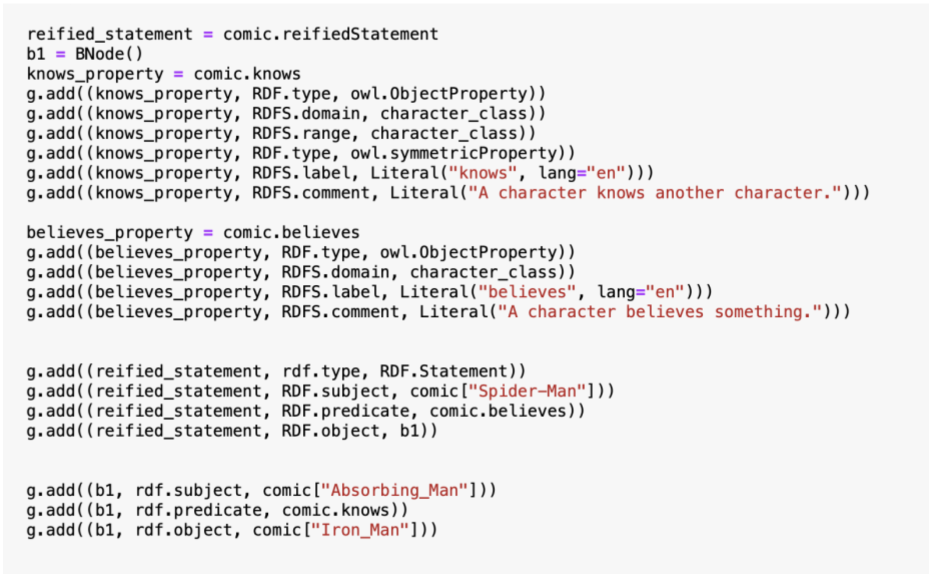
```
    [
        sh:path comic:hasWeight ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:integer ;
        sh:severity sh:Warning ;
        sh:message "Exactly one weight should be given!"@en ;
    ],
    [
        sh:path comic:hasSpecies ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:class comic:Species ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "Exactly one species should be given!"@en ;
    ],
    [
        sh:path comic:characterName ;
        sh:minLength 3 ;
        sh:maxLength 50 ;
        sh:minCount 1 ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "At least one character name should be given!"@en ;
    ],
    [
        sh:path comic:hasCharacterID ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:integer ;
        sh:severity sh:Warning ;
        sh:message "Exactly one character ID should be given!"@en ;
    ],
    [
        sh:path comic:hasLivingStatus ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:severity sh:Warning ;
        sh:message "Exactly one living status should be given!"@en ;
    ],
    [
```

Figure 23: Shacl, part 2

Loading the ontology and the Shacl file in GraphDB, the ontology could be aligned with the restrictions and thus verified. Additionally, using Protégé allowed to delete any occurring errors or areas prone to errors, i.e. warnings. Consequently, an error free ontology could be compiled.

5. RDF REIFICATION

RDF reification is a mechanism that allows for statements about statements. An RDF triple can thus give information about another RDF triple. An example of this is shown in Figure 24.

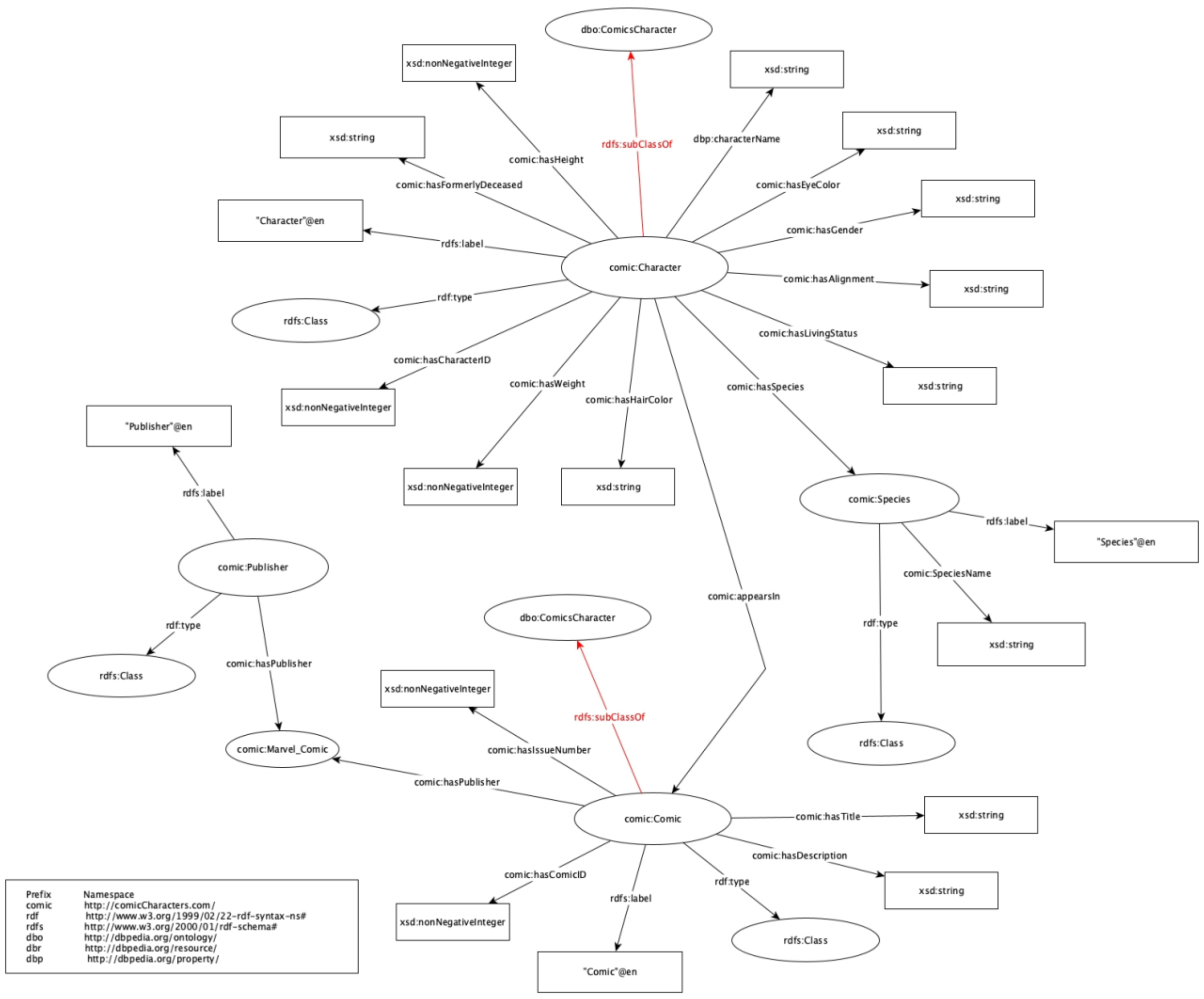


```
reified_statement = comic.reifiedStatement
b1 = BNode()
knows_property = comic.knows
g.add((knows_property, RDF.type, owl.ObjectProperty))
g.add((knows_property, RDFS.domain, character_class))
g.add((knows_property, RDFS.range, character_class))
g.add((knows_property, RDF.type, owl.symmetricProperty))
g.add((knows_property, RDFS.label, Literal("knows", lang="en")))
g.add((knows_property, RDFS.comment, Literal("A character knows another character.")))

believes_property = comic.believes
g.add((believes_property, RDF.type, owl.ObjectProperty))
g.add((believes_property, RDFS.domain, character_class))
g.add((believes_property, RDFS.label, Literal("believes", lang="en")))
g.add((believes_property, RDFS.comment, Literal("A character believes something.")))


g.add((reified_statement, rdf.type, RDF.Statement))
g.add((reified_statement, RDF.subject, comic["Spider-Man"]))
g.add((reified_statement, RDF.predicate, comic.believes))
g.add((reified_statement, RDF.object, b1))


g.add((b1, rdf.subject, comic["Absorbing_Man"]))
g.add((b1, rdf.predicate, comic.knows))
g.add((b1, rdf.object, comic["Iron_Man"]))
```

Figure 24: RDF reification

6. DESCRIPTION OF THE ONTOLOGY BY GRAPHML

The following graph (Figure 25) represents a visual overview of the classes previously mentioned. The graph not only shows the four distinct classes but all properties and respective types of the resources. Additionally, the graph visualizes how the classes connect with each other. Furthermore, inheritance of classes is highlighted in red and the “rdfs:subClassOf” predicate is used. Finally, the labels are also visualized.



Prefix	Namespace
comic	http://comicCharacters.com/
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
dbo	http://dbpedia.org/ontology/
dbr	http://dbpedia.org/resource/
dbp	http://dbpedia.org/property/

Figure 25: GraphML

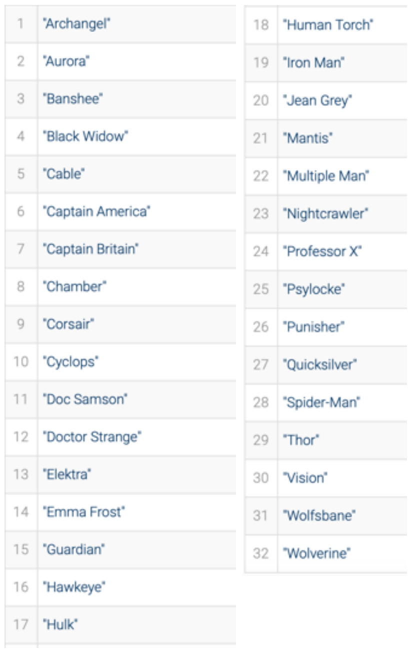
7. SPARQL QUERIES

Several Sparql queries were conducted to gain insight into the data set. When possible, visual graphs are added to illustrate the data. The following section presents questions of interest, followed by the queries conducted. Finally, the result is interpreted for each enquiry.

a. Which characters in our data have the alignment ….

…"good" and were also formerly deceased?

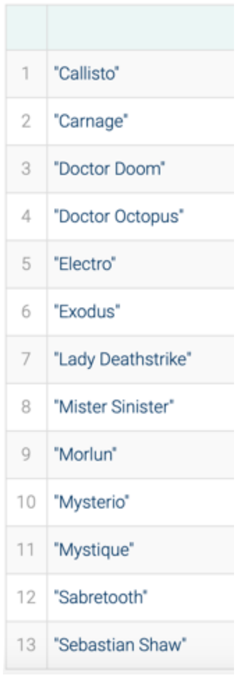
```
PREFIX comic: <http://comicCharacters.com/>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?name
WHERE {
            ?character a comic:Character .
   ?character dbp:characterName ?name .
   ?character comic:hasAlignment "good" .
   ?character comic:FormelyDeceased "Yes" .}
```



1	"Archangel"
2	"Aurora"
3	"Banshee"
4	"Black Widow"
5	"Cable"
6	"Captain America"
7	"Captain Britain"
8	"Chamber"
9	"Corsair"
10	"Cyclops"
11	"Doc Samson"
12	"Doctor Strange"
13	"Elektra"
14	"Emma Frost"
15	"Guardian"
16	"Hawkeye"
17	"Hulk"
18	"Human Torch"
19	"Iron Man"
20	"Jean Grey"
21	"Mantis"
22	"Multiple Man"
23	"Nightcrawler"
24	"Professor X"
25	"Psylocke"
26	"Punisher"
27	"Quicksilver"
28	"Spider-Man"
29	"Thor"
30	"Vision"
31	"Wolfsbane"
32	"Wolverine"

… "bad" and were also formerly deceased?

```
PREFIX comic: <http://comicCharacters.com/>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?name
WHERE {
            ?character a comic:Character .
   ?character dbp:characterName ?name .
   ?character comic:hasAlignment "bad" .
   ?character comic:FormelyDeceased "Yes" .
}
```

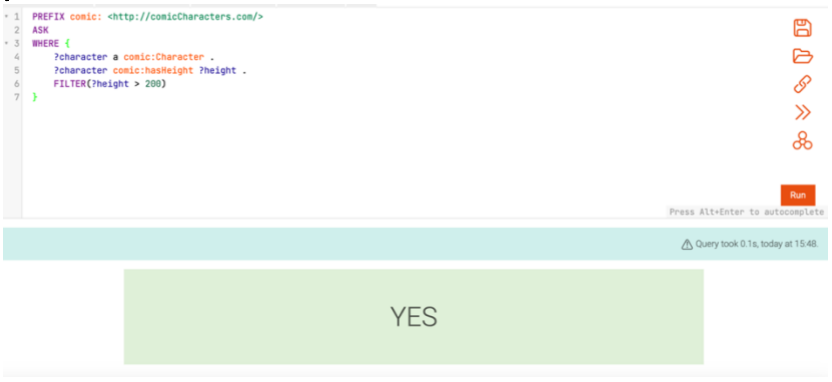


1	"Callisto"
2	"Carnage"
3	"Doctor Doom"
4	"Doctor Octopus"
5	"Electro"
6	"Exodus"
7	"Lady Deathstrike"
8	"Mister Sinister"
9	"Morlun"
10	"Mysterio"
11	"Mystique"
12	"Sabretooth"
13	"Sebastian Shaw"

This inquiry is of importance to obtain an overview over the different characters that have a “good” or “bad” alignment” and were formerly deceased. Furthermore, results show that there are more good characters that were formerly deceased. So, it’s a victory of good over evil.

b. Is there any character in our dataset that is taller than 200cm?

```
PREFIX comic: <http://comicCharacters.com/>
ASK
WHERE {
   ?character a comic:Character .
   ?character comic:hasHeight ?height .
   FILTER(?height > 200)
}
```



```
PREFIX comic: <http://comicCharacters.com/>
ASK
WHERE {
    ?character a comic:Character .
    ?character comic:hasHeight ?height .
    FILTER(?height > 200)
}
```

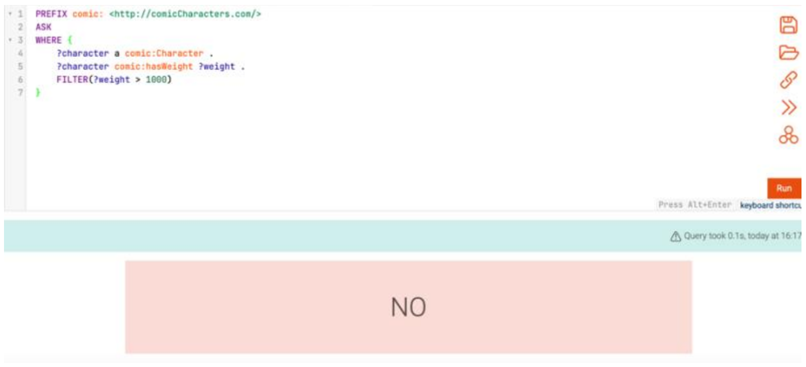
Query took 0.1s, today at 15:48.

YES

There is at least one character over 200cm. This implies that there are characters of bigger built.

c. Is there a character that is heavier than 1000 kg?

```
PREFIX comic: <http://comicCharacters.com/>

ASK

WHERE {

   ?character a comic:Character .

   ?character comic:hasWeight ?weight .

   FILTER(?weight > 1000)

}
```



```
PREFIX comic: <http://comicCharacters.com/>
ASK
WHERE {
    ?character a comic:Character .
    ?character comic:hasWeight ?weight .
    FILTER(?weight > 1000)
}
```

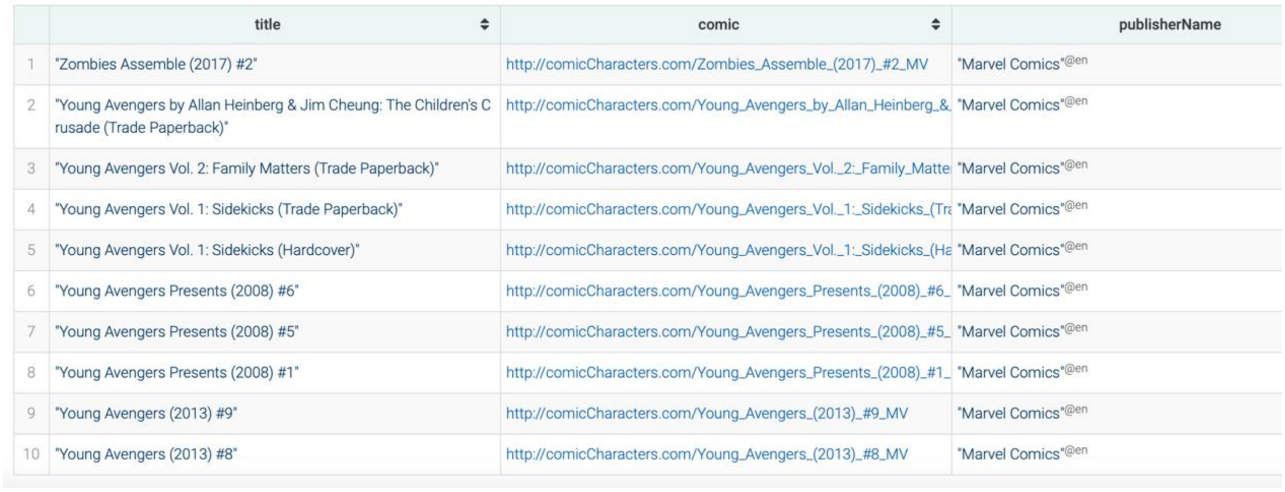
Query took 0.1s, today at 16:17

NO

There is no character over 1000kg. That tells the user that even though there are heavy characters, there is some limit.

d. What are the last 10 titles if we look at the titles alphabetically? And who is the publisher of that comic?

```
PREFIX comic: <http://comicCharacters.com/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?title ?comic ?publisherName
WHERE {
   ?comic a comic:Comic ;
            comic:hasTitle ?title ;
            comic:hasPublisher ?publisher .
   ?publisher comic:publisherName ?publisherName .
}
ORDER BY DESC(?title)
LIMIT 10
```

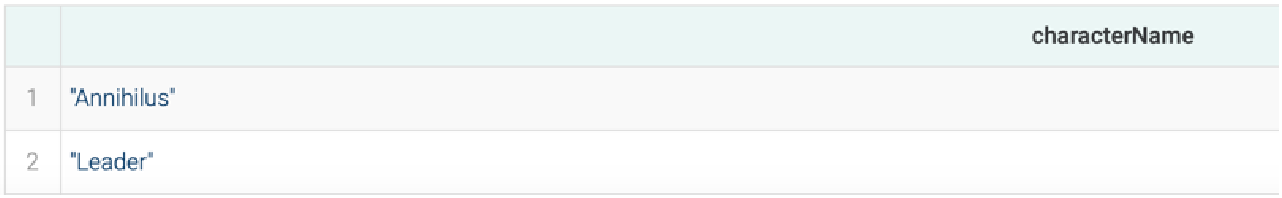


	title	comic	publisherName
1	"Zombies Assemble (2017) #2"	http://comicCharacters.com/Zombies_Assemble_(2017)_#2_MV	"Marvel Comics"@en
2	"Young Avengers by Allan Heinberg & Jim Cheung: The Children's Crusade (Trade Paperback)"	http://comicCharacters.com/Young_Avengers_by_Allan_Heinberg_&	"Marvel Comics"@en
3	"Young Avengers Vol. 2: Family Matters (Trade Paperback)"	http://comicCharacters.com/Young_Avengers_Vol._2:_Family_Matte	"Marvel Comics"@en
4	"Young Avengers Vol. 1: Sidekicks (Trade Paperback)"	http://comicCharacters.com/Young_Avengers_Vol._1:_Sidekicks_(Tra	"Marvel Comics"@en
5	"Young Avengers Vol. 1: Sidekicks (Hardcover)"	http://comicCharacters.com/Young_Avengers_Vol._1:_Sidekicks_(Ha	"Marvel Comics"@en
6	"Young Avengers Presents (2008) #6"	http://comicCharacters.com/Young_Avengers_Presents_(2008)_#6_	"Marvel Comics"@en
7	"Young Avengers Presents (2008) #5"	http://comicCharacters.com/Young_Avengers_Presents_(2008)_#5_	"Marvel Comics"@en
8	"Young Avengers Presents (2008) #1"	http://comicCharacters.com/Young_Avengers_Presents_(2008)_#1_	"Marvel Comics"@en
9	"Young Avengers (2013) #9"	http://comicCharacters.com/Young_Avengers_(2013)_#9_MV	"Marvel Comics"@en
10	"Young Avengers (2013) #8"	http://comicCharacters.com/Young_Avengers_(2013)_#8_MV	"Marvel Comics"@en

The titles of the comics are not uniform, so they do not always start with the same letter. Also, some of the comics have the issue number and year they were released attached. All the comics in our data have the same publisher, Marvel Comics. The comic URIs all end with "_MV", this was added by us to distinctly mention that they are from Marvel Comics.

e. What are the names of the characters that have no hair and green eyes?

```
PREFIX comic: <http://comicCharacters.com/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT ?characterName
WHERE {
    ?character comic:hasHairColor "No Hair"^^xsd:string ;
            comic:hasEyeColor "green"^^xsd:string .
    ?character dbp:characterName ?characterName .
}
```

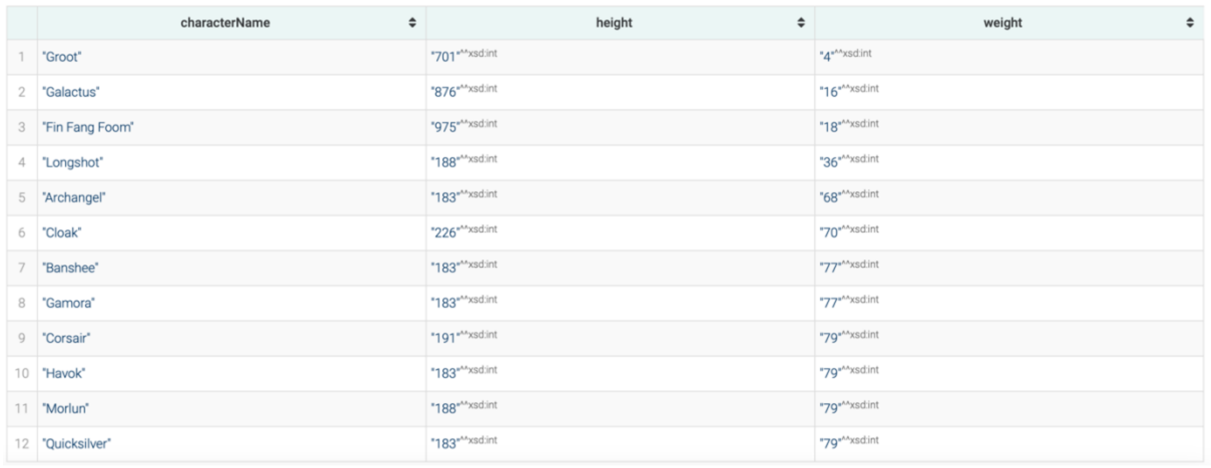


	characterName
1	"Annihilus"
2	"Leader"

This restriction only applies to two characters. This means it is not that common that a character has green eyes and no hair.

f. **What characters are over 180cm and weight less than 80kg? Oder by the weight of the characters.**

```
PREFIX comic: <http://comicCharacters.com/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?characterName ?height ?weight
    WHERE {
    ?character comic:hasHeight ?height ;
          comic:hasWeight ?weight .
    ?character dbp:characterName ?characterName
    FILTER (?height > "180"^^xsd:int && ?weight < "80"^^xsd:int)
}
ORDER BY ?weight
```

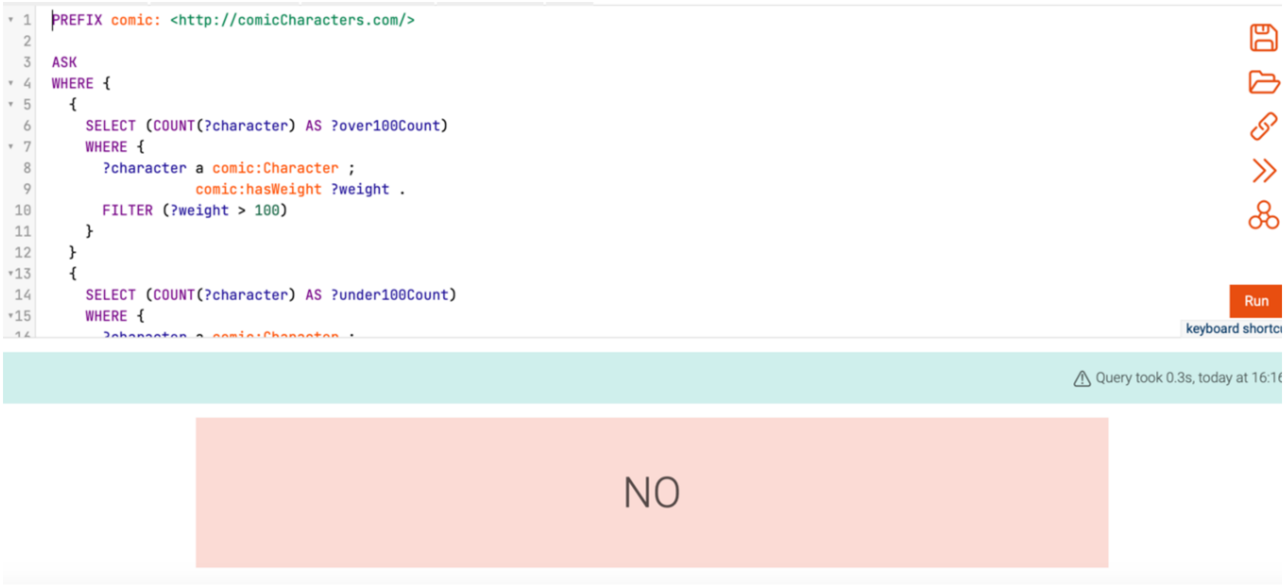


	characterName	height	weight
1	"Groot"	"701"^^xsd:int	"4"^^xsd:int
2	"Galactus"	"876"^^xsd:int	"16"^^xsd:int
3	"Fin Fang Foom"	"975"^^xsd:int	"18"^^xsd:int
4	"Longshot"	"188"^^xsd:int	"36"^^xsd:int
5	"Archangel"	"183"^^xsd:int	"68"^^xsd:int
6	"Cloak"	"226"^^xsd:int	"70"^^xsd:int
7	"Banshee"	"183"^^xsd:int	"77"^^xsd:int
8	"Gamora"	"183"^^xsd:int	"77"^^xsd:int
9	"Corsair"	"191"^^xsd:int	"79"^^xsd:int
10	"Havok"	"183"^^xsd:int	"79"^^xsd:int
11	"Morlun"	"188"^^xsd:int	"79"^^xsd:int
12	"Quicksilver"	"183"^^xsd:int	"79"^^xsd:int

The range is quite big. Groot is really tall, but at the same time weighs very little. Furthermore, the data results show that there are very few characters that fulfil the given requirements.

g. Are there more characters that weight more than 100 kg than characters that weight under 100kg?

```
PREFIX comic: <http://comicCharacters.com/>
ASK
WHERE {
 {
  SELECT (COUNT(?character) AS ?over100Count)
  WHERE {
   ?character a comic:Character ;
              comic:hasWeight ?weight .
   FILTER (?weight > 100)
  }
 }
 {
  SELECT (COUNT(?character) AS ?under100Count)
  WHERE {
   ?character a comic:Character ;
              comic:hasWeight ?weight .
   FILTER (?weight < 100)
  }
 }
 FILTER (?over100Count > ?under100Count)
}
```



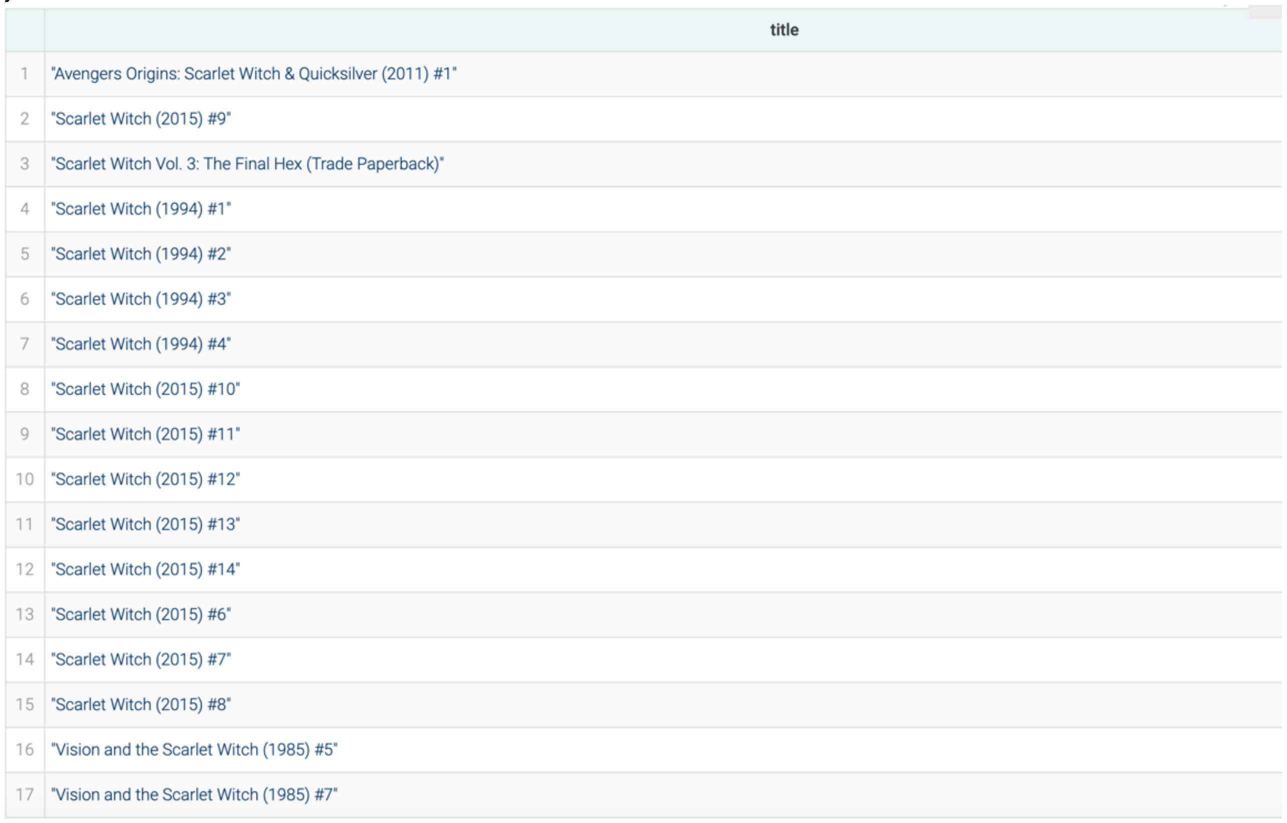
```
PREFIX comic: <http://comicCharacters.com/>

ASK
WHERE {
  {
    SELECT (COUNT(?character) AS ?over100Count)
    WHERE {
      ?character a comic:Character ;
                 comic:hasWeight ?weight .
      FILTER (?weight > 100)
    }
  }
  {
    SELECT (COUNT(?character) AS ?under100Count)
    WHERE {
```

NO

There are more characters that are under 100kg. That means that even though some of them are really big or really heavy, the majority of them seem to lie in the normal weight range for humans.

h. What comic titles contain the name of the character “Scarlet Witch”?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT ?title
WHERE {
   ?comic a comic:Comic .
   ?comic comic:hasTitle ?title .
   FILTER(CONTAINS(?title,"Scarlet Witch"))
}
```

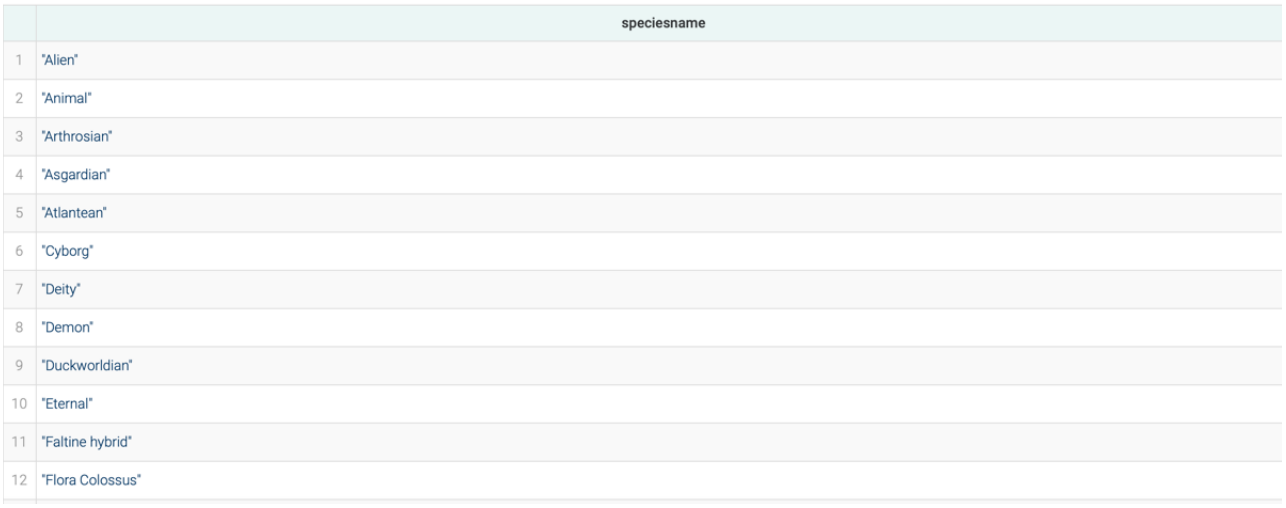
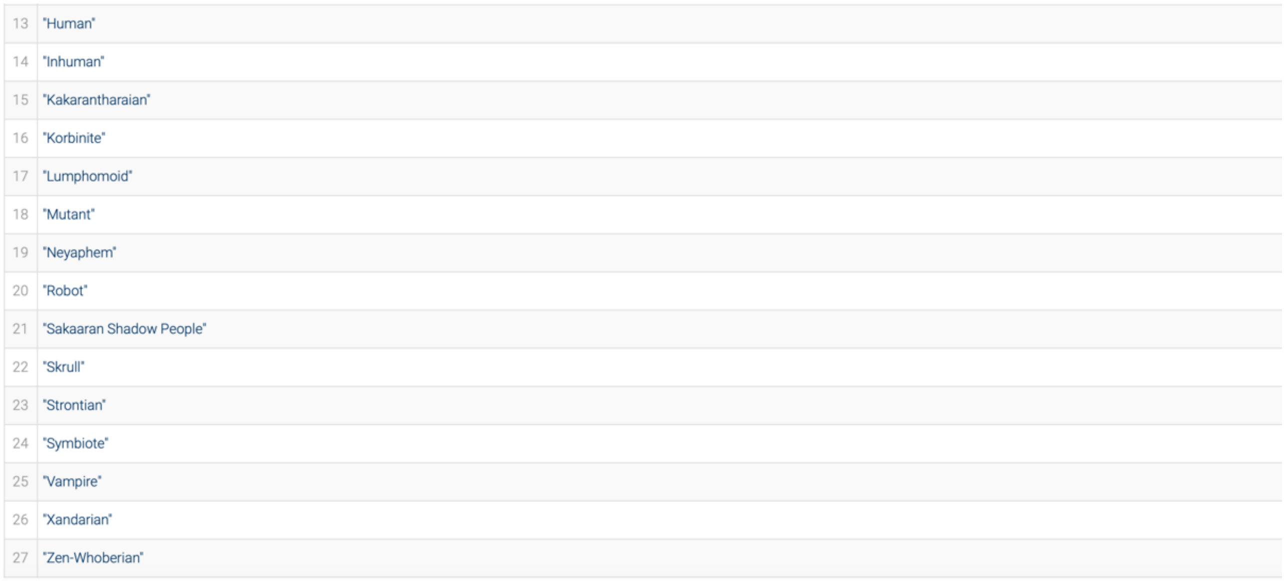


	title
1	"Avengers Origins: Scarlet Witch & Quicksilver (2011) #1"
2	"Scarlet Witch (2015) #9"
3	"Scarlet Witch Vol. 3: The Final Hex (Trade Paperback)"
4	"Scarlet Witch (1994) #1"
5	"Scarlet Witch (1994) #2"
6	"Scarlet Witch (1994) #3"
7	"Scarlet Witch (1994) #4"
8	"Scarlet Witch (2015) #10"
9	"Scarlet Witch (2015) #11"
10	"Scarlet Witch (2015) #12"
11	"Scarlet Witch (2015) #13"
12	"Scarlet Witch (2015) #14"
13	"Scarlet Witch (2015) #6"
14	"Scarlet Witch (2015) #7"
15	"Scarlet Witch (2015) #8"
16	"Vision and the Scarlet Witch (1985) #5"
17	"Vision and the Scarlet Witch (1985) #7"

Even though Scarlet Witch is not the best-known character, she appears in a variety of comic titles. In total she appears in 17 titles in our dataset.

i. What are the names of the different species that are contained in the data?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT DISTINCT ?speciesname
WHERE {
   ?species a comic:Species .
   ?species comic:SpeciesName ?speciesname .

}
```

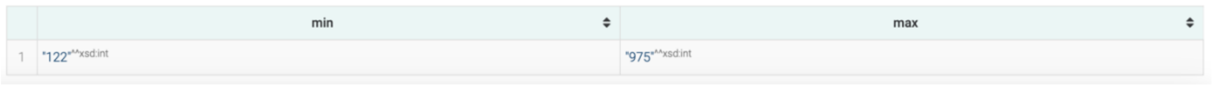



	speciesname
1	"Alien"
2	"Animal"
3	"Arthrosian"
4	"Asgardian"
5	"Atlantean"
6	"Cyborg"
7	"Deity"
8	"Demon"
9	"Duckworldian"
10	"Eternal"
11	"Faltine hybrid"
12	"Flora Colossus"
13	"Human"
14	"Inhuman"
15	"Kakarantharaian"
16	"Korbinite"
17	"Lumphomoid"
18	"Mutant"
19	"Neyaphem"
20	"Robot"
21	"Sakaaran Shadow People"
22	"Skrull"
23	"Strontian"
24	"Symbiote"
25	"Vampire"
26	"Xandarian"
27	"Zen-Whoberian"

The user can see that there is a number of different species. Some of them seem more similar to each other like for example Cyborg and Robot or Aliens and Skrull, but there are also many species that differ a lot from the others, like for example Humans and Flora Colossus.

j. What is the range of the height of the characters?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT (MIN(?height) AS ?min) (MAX(?height) AS ?max)
WHERE {
   ?char a comic:Character .
   ?char comic:hasHeight ?height .
}
```

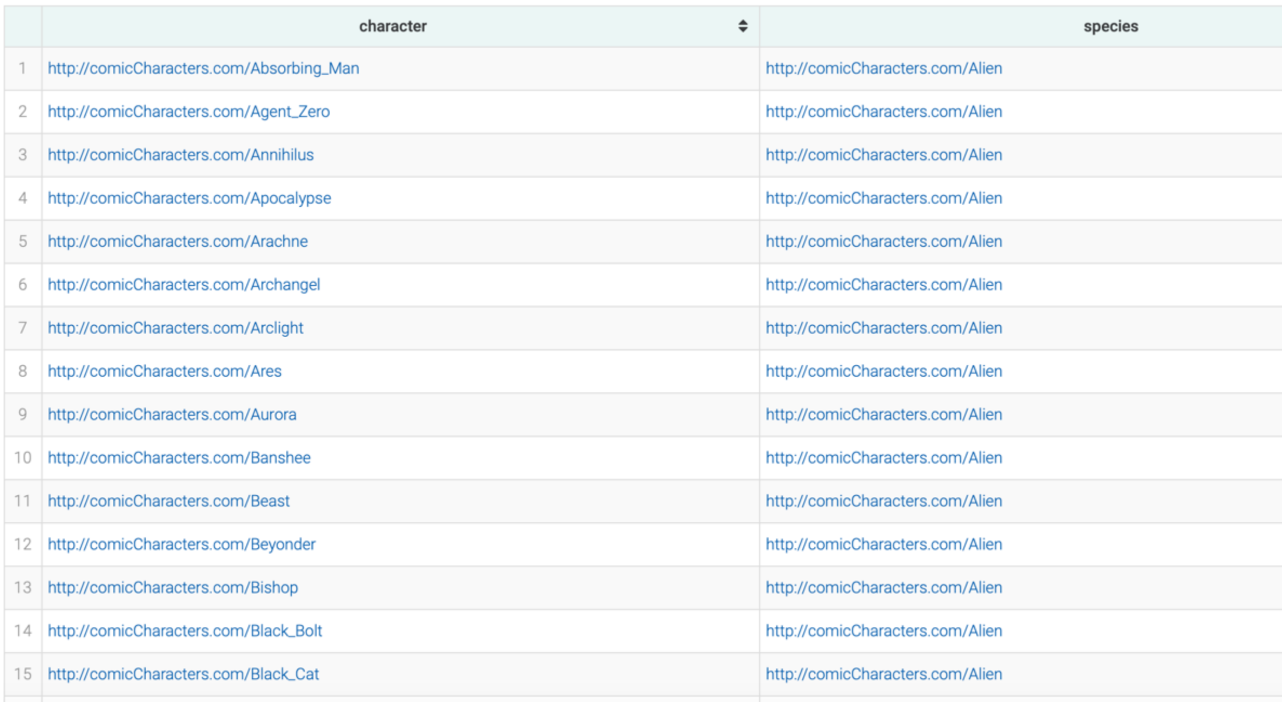


	min	max
1	"122"^^xsd:int	"975"^^xsd:int

The difference from the smallest to the tallest character is quite big, over 800cm.

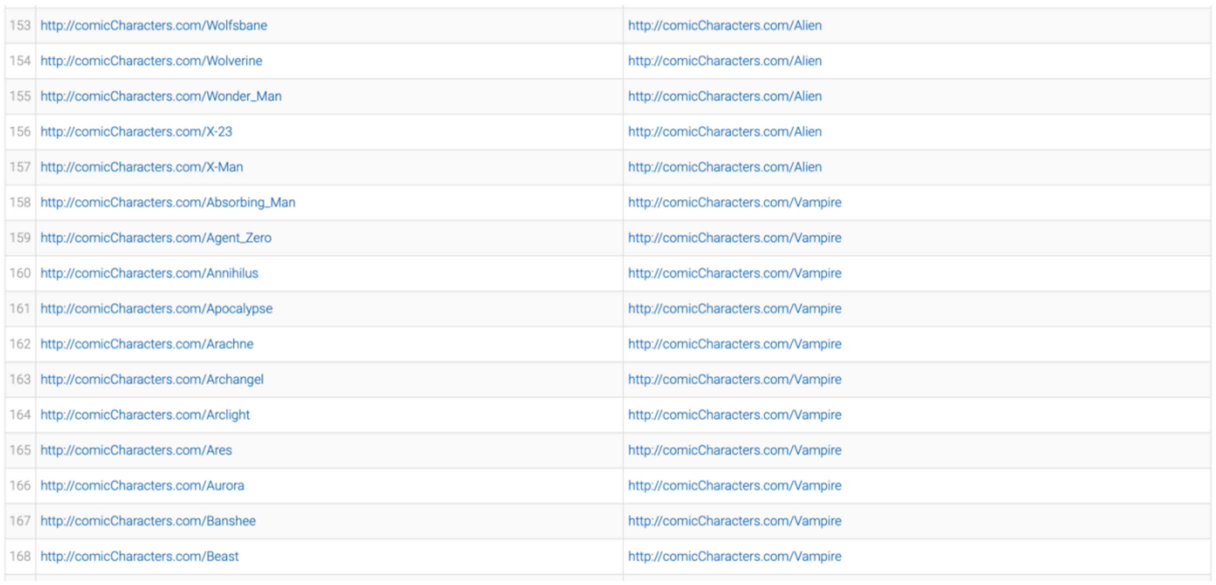
What characters are either of species Alien or Vampire? And to which species do they belong?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT ?character ?species
WHERE {
   ?character a comic:Character .
   ?species a comic:Species .
   ?species comic:SpeciesName ?name .
   {
      FILTER(CONTAINS(?name,"Alien"))
   }
   UNION {
      FILTER(CONTAINS(?name,"Vampire"))
   }
}
```



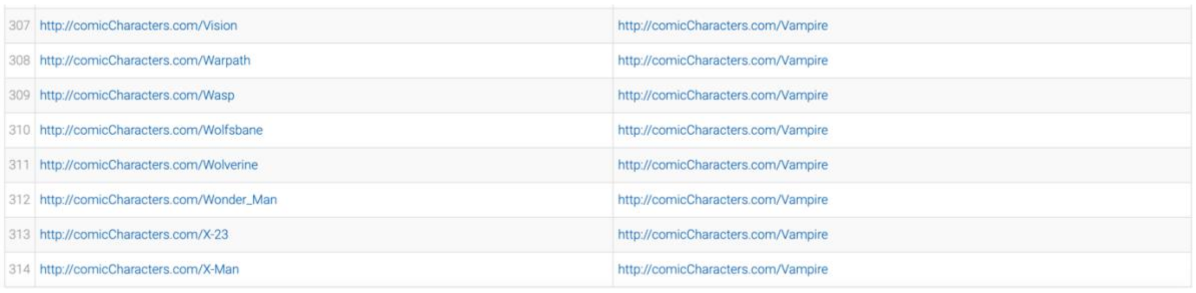
	character	species
1	http://comicCharacters.com/Absorbing_Man	http://comicCharacters.com/Alien
2	http://comicCharacters.com/Agent_Zero	http://comicCharacters.com/Alien
3	http://comicCharacters.com/Annihilus	http://comicCharacters.com/Alien
4	http://comicCharacters.com/Apocalypse	http://comicCharacters.com/Alien
5	http://comicCharacters.com/Arachne	http://comicCharacters.com/Alien
6	http://comicCharacters.com/Archangel	http://comicCharacters.com/Alien
7	http://comicCharacters.com/Arclight	http://comicCharacters.com/Alien
8	http://comicCharacters.com/Ares	http://comicCharacters.com/Alien
9	http://comicCharacters.com/Aurora	http://comicCharacters.com/Alien
10	http://comicCharacters.com/Banshee	http://comicCharacters.com/Alien
11	http://comicCharacters.com/Beast	http://comicCharacters.com/Alien
12	http://comicCharacters.com/Beyonder	http://comicCharacters.com/Alien
13	http://comicCharacters.com/Bishop	http://comicCharacters.com/Alien
14	http://comicCharacters.com/Black_Bolt	http://comicCharacters.com/Alien
15	http://comicCharacters.com/Black_Cat	http://comicCharacters.com/Alien

[…]



153	http://comicCharacters.com/Wolfsbane	http://comicCharacters.com/Alien
154	http://comicCharacters.com/Wolverine	http://comicCharacters.com/Alien
155	http://comicCharacters.com/Wonder_Man	http://comicCharacters.com/Alien
156	http://comicCharacters.com/X-23	http://comicCharacters.com/Alien
157	http://comicCharacters.com/X-Man	http://comicCharacters.com/Alien
158	http://comicCharacters.com/Absorbing_Man	http://comicCharacters.com/Vampire
159	http://comicCharacters.com/Agent_Zero	http://comicCharacters.com/Vampire
160	http://comicCharacters.com/Annihilus	http://comicCharacters.com/Vampire
161	http://comicCharacters.com/Apocalypse	http://comicCharacters.com/Vampire
162	http://comicCharacters.com/Arachne	http://comicCharacters.com/Vampire
163	http://comicCharacters.com/Archangel	http://comicCharacters.com/Vampire
164	http://comicCharacters.com/Arclight	http://comicCharacters.com/Vampire
165	http://comicCharacters.com/Ares	http://comicCharacters.com/Vampire
166	http://comicCharacters.com/Aurora	http://comicCharacters.com/Vampire
167	http://comicCharacters.com/Banshee	http://comicCharacters.com/Vampire
168	http://comicCharacters.com/Beast	http://comicCharacters.com/Vampire

[…]

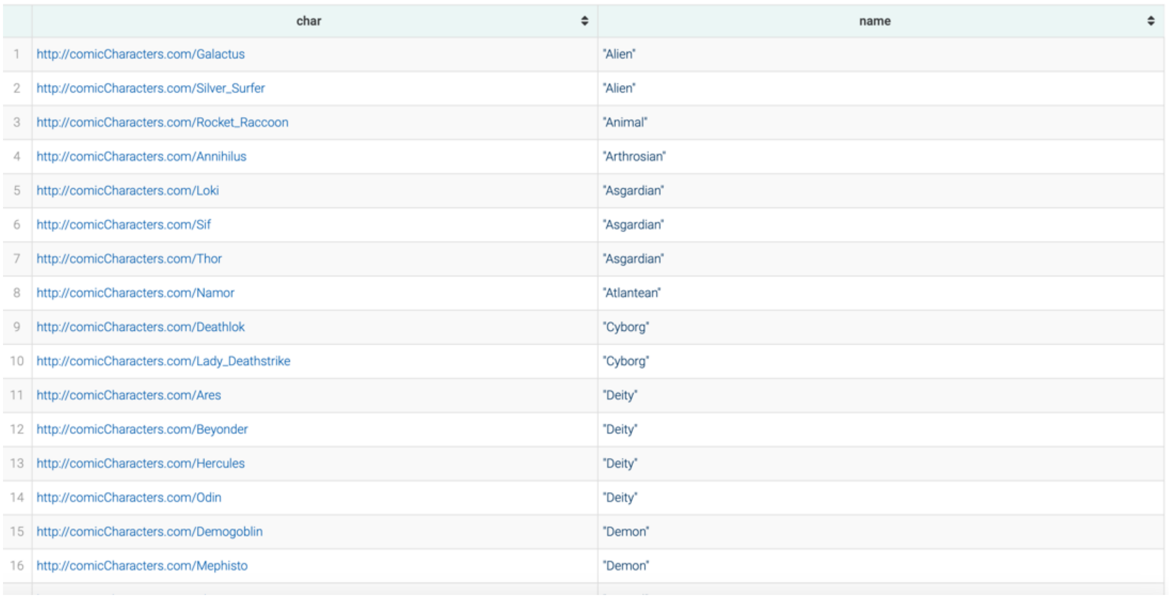


307	http://comicCharacters.com/Vision	http://comicCharacters.com/Vampire
308	http://comicCharacters.com/Warpath	http://comicCharacters.com/Vampire
309	http://comicCharacters.com/Wasp	http://comicCharacters.com/Vampire
310	http://comicCharacters.com/Wolfsbane	http://comicCharacters.com/Vampire
311	http://comicCharacters.com/Wolverine	http://comicCharacters.com/Vampire
312	http://comicCharacters.com/Wonder_Man	http://comicCharacters.com/Vampire
313	http://comicCharacters.com/X-23	http://comicCharacters.com/Vampire
314	http://comicCharacters.com/X-Man	http://comicCharacters.com/Vampire

Many characters belong to either of both species. There is the same number of vampires and aliens in the dataset.

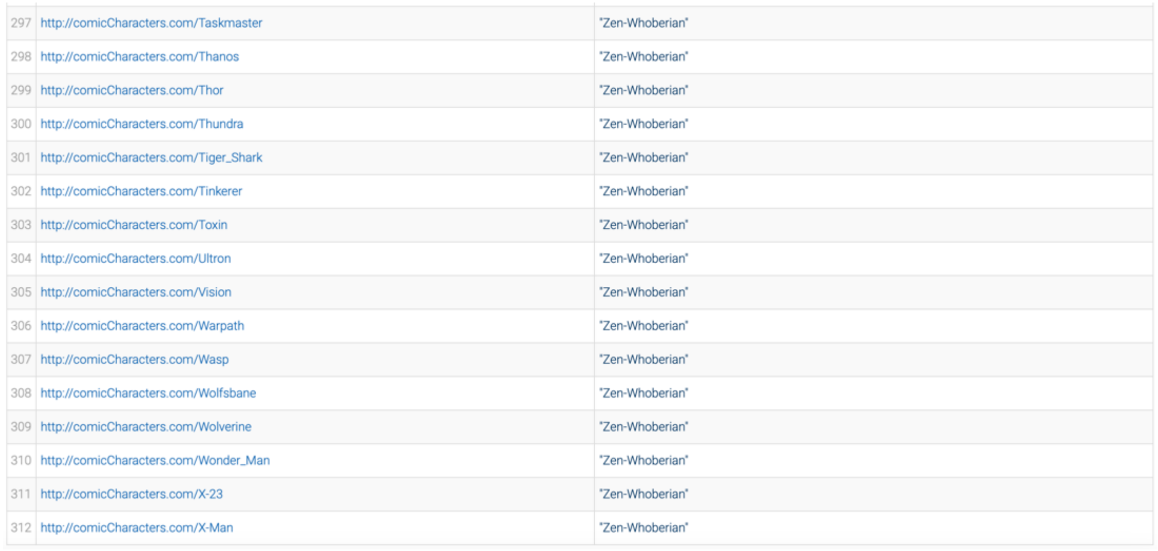
k. To what species does each character belong to?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT ?char ?name
WHERE {
   ?char a comic:Character .
   ?char comic:hasSpecies ?species .
   ?species comic:SpeciesName ?name .
}
```



	char	name
1	http://comicCharacters.com/Galactus	"Alien"
2	http://comicCharacters.com/Silver_Surfer	"Alien"
3	http://comicCharacters.com/Rocket_Raccoon	"Animal"
4	http://comicCharacters.com/Annihilus	"Arthrosian"
5	http://comicCharacters.com/Loki	"Asgardian"
6	http://comicCharacters.com/Sif	"Asgardian"
7	http://comicCharacters.com/Thor	"Asgardian"
8	http://comicCharacters.com/Namor	"Atlantean"
9	http://comicCharacters.com/Deathlok	"Cyborg"
10	http://comicCharacters.com/Lady_Deathstrike	"Cyborg"
11	http://comicCharacters.com/Ares	"Deity"
12	http://comicCharacters.com/Beyonder	"Deity"
13	http://comicCharacters.com/Hercules	"Deity"
14	http://comicCharacters.com/Odin	"Deity"
15	http://comicCharacters.com/Demogoblin	"Demon"
16	http://comicCharacters.com/Mephisto	"Demon"

[…]

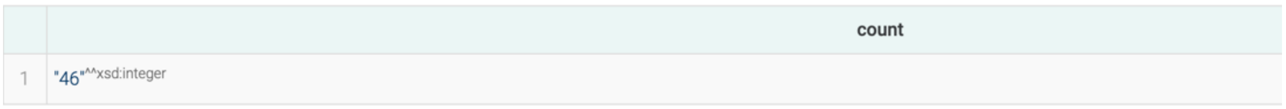


297	http://comicCharacters.com/Taskmaster	"Zen-Whoberian"
298	http://comicCharacters.com/Thanos	"Zen-Whoberian"
299	http://comicCharacters.com/Thor	"Zen-Whoberian"
300	http://comicCharacters.com/Thundra	"Zen-Whoberian"
301	http://comicCharacters.com/Tiger_Shark	"Zen-Whoberian"
302	http://comicCharacters.com/Tinkerer	"Zen-Whoberian"
303	http://comicCharacters.com/Toxin	"Zen-Whoberian"
304	http://comicCharacters.com/Ultron	"Zen-Whoberian"
305	http://comicCharacters.com/Vision	"Zen-Whoberian"
306	http://comicCharacters.com/Warpath	"Zen-Whoberian"
307	http://comicCharacters.com/Wasp	"Zen-Whoberian"
308	http://comicCharacters.com/Wolfsbane	"Zen-Whoberian"
309	http://comicCharacters.com/Wolverine	"Zen-Whoberian"
310	http://comicCharacters.com/Wonder_Man	"Zen-Whoberian"
311	http://comicCharacters.com/X-23	"Zen-Whoberian"
312	http://comicCharacters.com/X-Man	"Zen-Whoberian"

This query is for the user to get an overview. There seem to be many Zen-Whoberians in our dataset.

l. How many mutants are alive?

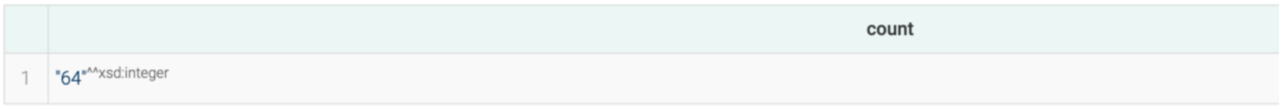
```
PREFIX comic: <http://comicCharacters.com/>
SELECT (COUNT(?char) AS ?count)
WHERE {
   ?char a comic:Character .
   ?char comic:hasLivingStatus ?alive .
   ?char comic:hasSpecies ?species .
   {
     FILTER(?species = comic:Mutant && ?alive = "Alive")
   }
}
```



	count
1	"46"^^xsd:integer

m. How many mutants are there in total?

```
PREFIX comic: <http://comicCharacters.com/>
SELECT (COUNT(?char) AS ?count)
WHERE {
   ?char a comic:Character .
   ?char comic:hasLivingStatus ?alive .
   ?char comic:hasSpecies ?species .
   {
     FILTER(?species = comic:Mutant)
   }
}
```



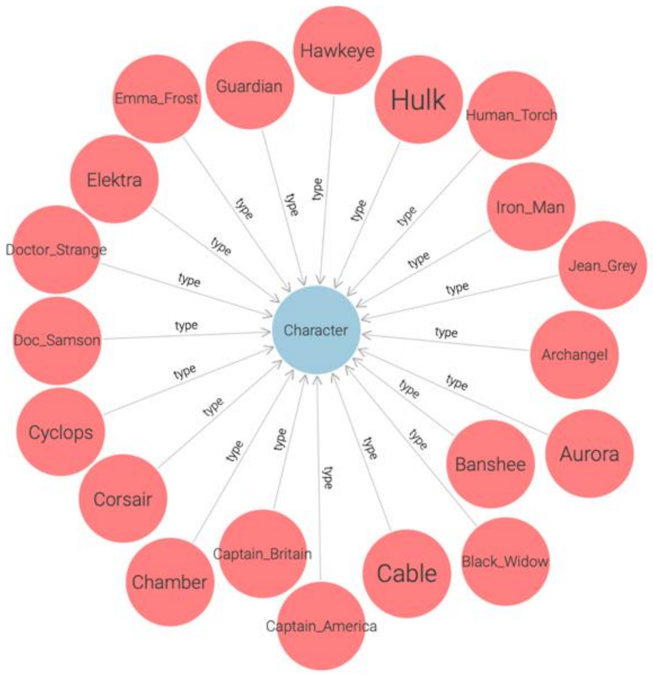
	count
1	"64"^^xsd:integer

The majority of the mutants is still alive which means that they are not necessarily the enemy that has to be defeated and killed.

8. VISUALIZATIONS

a. We want to inspect the characters that have the good alignment “good” and are formerly deceased.

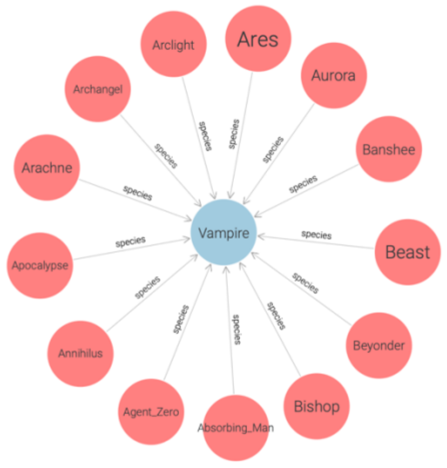
```
PREFIX comic: <http://comicCharacters.com/>
PREFIX dbp: <http://dbpedia.org/property/>
CONSTRUCT {
   ?character a comic:Character .
}
WHERE {
   ?character dbp:characterName ?name .
   ?character comic:hasAlignment "good" .
   ?character comic:FormelyDeceased "Yes" .
}
```

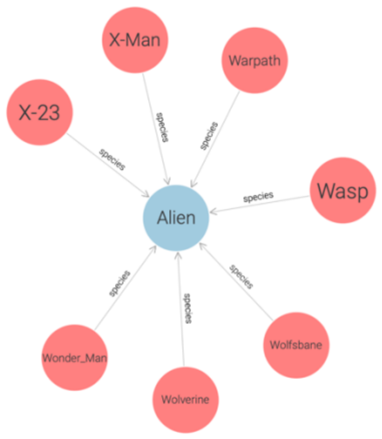


This gives the user a visual addition to the previous sparql query. It contains the same information in a different format.

b. Let’s look at s subset of the characters that are of species vampire or alien.

```
PREFIX comic: <http://comicCharacters.com/>

CONSTRUCT {
   ?character comic:hasSpecies ?species .
}
WHERE {
   ?character a comic:Character .
   ?species a comic:Species .
   ?species comic:SpeciesName ?name .
   {
      FILTER(CONTAINS(?name,"Alien"))
   }
   UNION {
      FILTER(CONTAINS(?name,"Vampire"))
   }
}
OFFSET 150
```

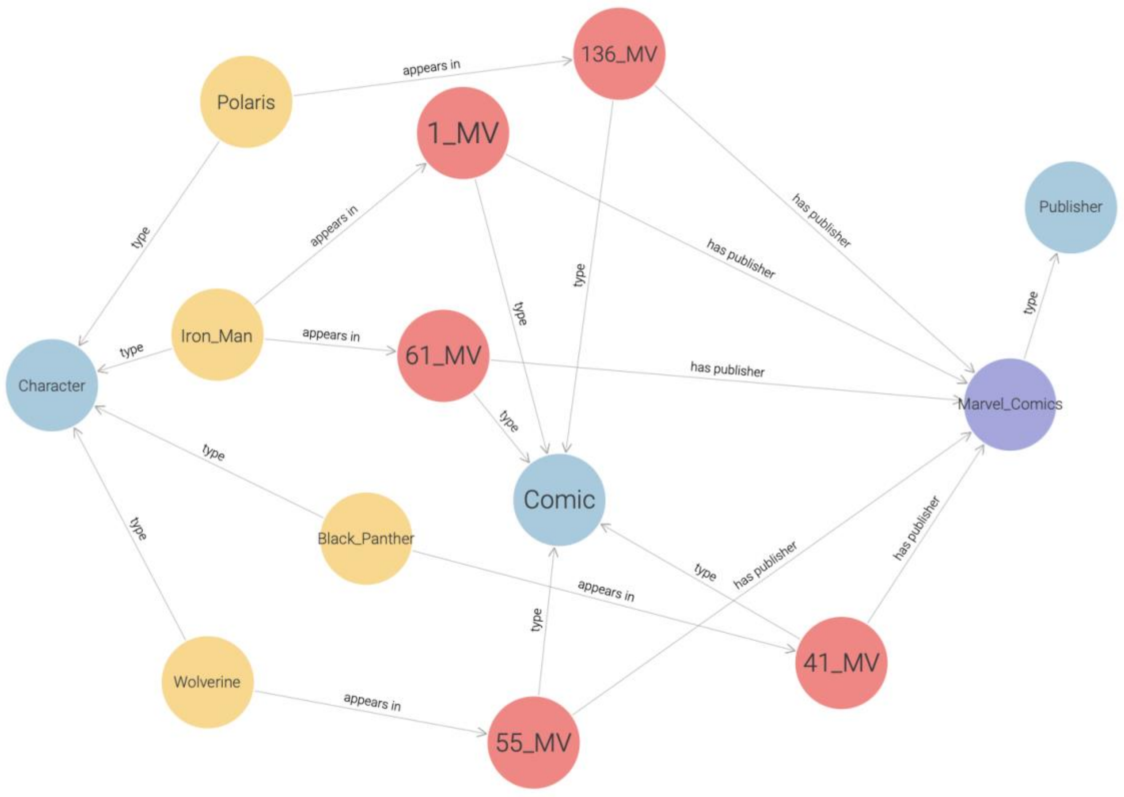




This subset conveys a wrong impression. It seems as if there are less aliens, but if we look at the query before we know that they appear equally often.

c. How are the classes comic, publisher and character connected with one another?

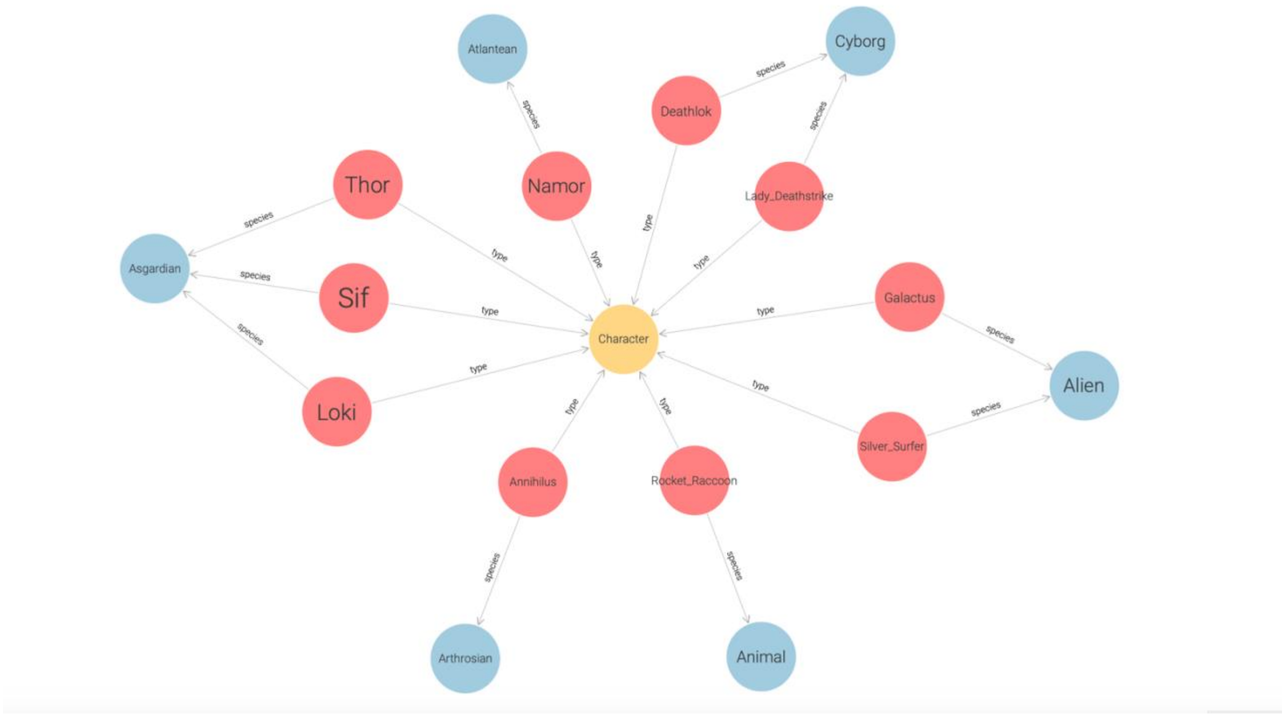
```
PREFIX comic: <http://comicCharacters.com/>
CONSTRUCT {
   ?comic a comic:Comic .
   ?character a comic:Character .
   ?publisher a comic:Publisher .
   ?comic ?haspub ?publisher .
   ?character ?appearsIn ?comic .
}
WHERE {
   ?publisher a comic:Publisher .
   ?comic a comic:Comic .
   ?comic comic:hasCharacters ?character .
   ?comic ?haspub ?publisher .
   ?character ?appearsIn ?comic .

}
ORDER BY RAND()
OFFSET 1110
LIMIT 25
```



With this visualization the user can see how the classes are interconnected. Every character appears in at least one comic. And every come has one publisher which is in our case always Marvel_comics.

d. Let's look at some characters and their species.

```
PREFIX comic: <http://comicCharacters.com/>
CONSTRUCT {
   ?char comic:hasSpecies ?species .
   ?char a comic:Character .
}
WHERE {
   ?char a comic:Character .
   ?char comic:hasSpecies ?species .
   ?species comic:SpeciesName ?name .
}
```



A character can only have one species. But more than one character can belong to the same species.

CONCLUSION

In conclusion, the creation of an ontology for comic book characters, comics, and publishers, coupled with the utilization of rdflib and SPARQL, offers a valuable resource for researchers, enthusiasts, and industry professionals. The ontology enhances the organization and accessibility of comic book information, providing a structured framework for efficient data retrieval and analysis. Further expansion and refinement of the ontology will continue to enrich the representation of the comic book domain, enabling deeper exploration and understanding of this vibrant universe.