Aim Shams University

Faculty of Engineering – ICHEP

Computer Engineering and Software Systems

CSE488: Ontologies and The Semantic Web

Major Task

Github: https://github.com/NoorhanHatem/OntologiesMT

Prepared By:

Noorhan Hatem Ibrahim [19P5821]

Serag Eldin Mohamed [19P1183]

Sohaila Mohamed Anwar [18P717]

Khaled Abdelmoniem Mohamed [14P8145]

Submitted:

Monday, May 13, 2024

PROBLEM DESCRIPTION

We set out to create an ontology which models movies. A movie has one or several directors, writers and actors. It also has a title, one or several genres, a year, a country and a language. To define the Genre of a movie, possible choices are: Thriller, Crime, Action, Drama, Romance, Horror, Sci-Fi or Comedy. Actors, directors and writers are persons. Persons have a gender: male or female, a name, an age and a nationality.

CLASSES/ENTITIES

I. Movie Class

The movie class represents the films in the ontology. It acts as the domain of the following datatype properties:

- movie_title
- movie_genre
- movie_country
- movie_language
- movie_year

It acts as the domain of the following object properties:

- hasActor
- hasDirector
- hasWriter

It acts as the range of the following object properties:

- isActorOf
- isWriterOf
- isDirectorOf

II. Person Class

The person class represents all humans in the ontology. It has three subclasses which further classify all people in the ontology into:

- Actor Class
- Director Class
- Writer Class

It acts as the domain of the following datatype properties:

- person_name
- person_gender
- person_age

- person_nationality

It acts as the domain of the following object properties:

- isActorOf
- isWriterOf
- isDirectorOf

It acts as the range of the following object properties:

- hasActor
- hasDirector
- hasWriter

PROPERTIES/RELATIONS

I. Object Properties

- isActorOf: lists the movies a person starred in.
    - Domain: person
    - Range: movie
- isDirectorOf: lists the movies a person directed.
    - Domain: person
    - Range: movie
- isWriterOf: lists the movies a person wrote.
    - Domain: person
    - Range: movie
- hasActor: lists the persons who starred in a movie.
    - Domain: movie
    - Range: person
- hasDirector: lists the persons who directed a movie.
    - Domain: movie
    - Range: person
- hasWriter: lists the persons who wrote a movie.
    - Domain: movie
    - Range: person

II. Datatype Properties

- movie_title
- movie_genre
- movie_country
- movie_language
- movie_year

- person_name
- person_gender
- person_age
- person_nationality

LOGIC

I. Axioms and Restrictions

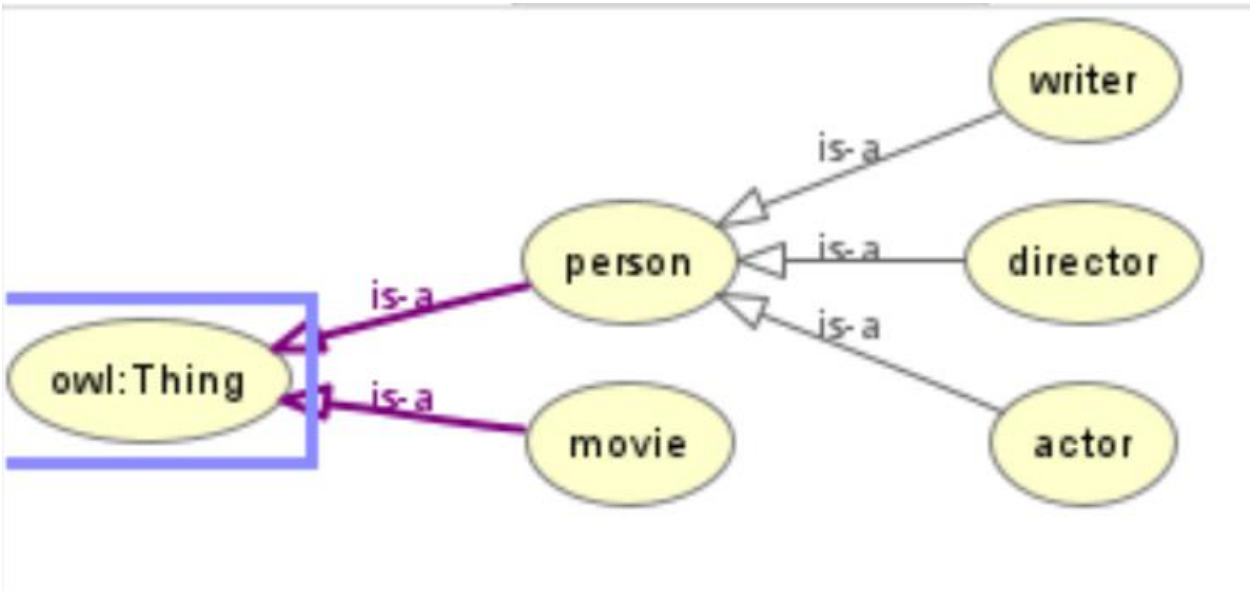
- A person can have multiple names to account for stage names etc.
- A person can have multiple nationalities.
- A person can only have a single entity for age and gender.
- A person's gender can only be male or female.
- A person can be either an actor, director, or writer; or a combination of two of them or of all three.
- The age of a dead person will be listed as their age when they died.
- A movie can have multiple countries to account for multi-national films.
- A movie can have multiple languages to account for multi-lingual movies.
- A movie can have multiple titles to account for films that have various names.
- A movie can have multiple genres to account for complex films.
- A movie can only have a single entry for "year".
- A movie's genre can be listed as one of more of the following 8 types only: Thriller, Crime, Drama, Action, Romance, Sci-Fi, Horror, Comedy.
- The movie and person classes are disjoint.

II. Inference Rules

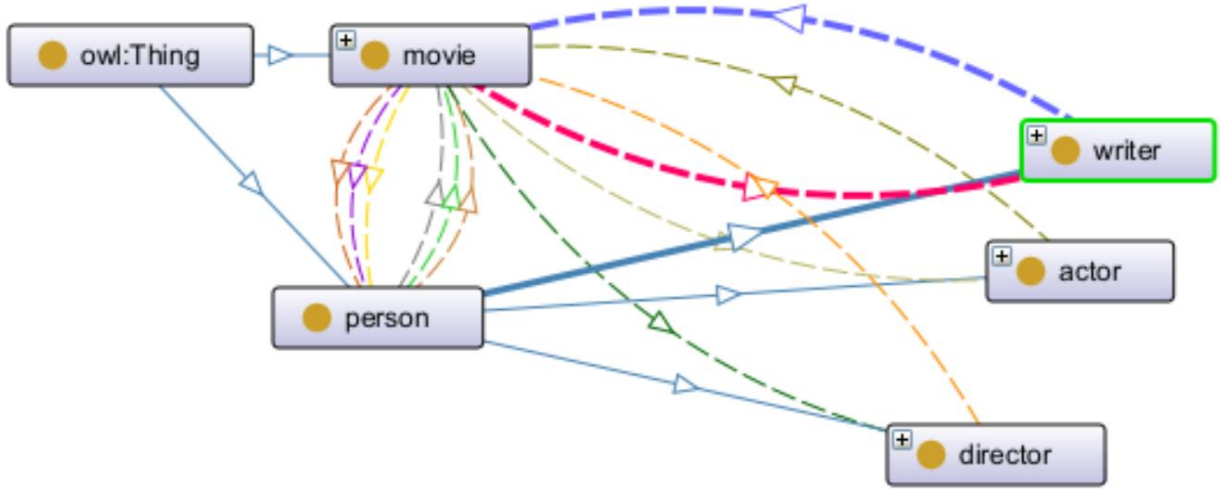
- A person cannot be a movie.
- If a person "p" isActorOf a movie "m", then that movie "m" hasActor person "p".
- If a person "p" isWriterOf a movie "m", then that movie "m" hasWriter person "p".
- If a person "p" isDirectorOf a movie "m", then that movie "m" hasDirector person "p".

VISUALIZING THE ONTOLOGY

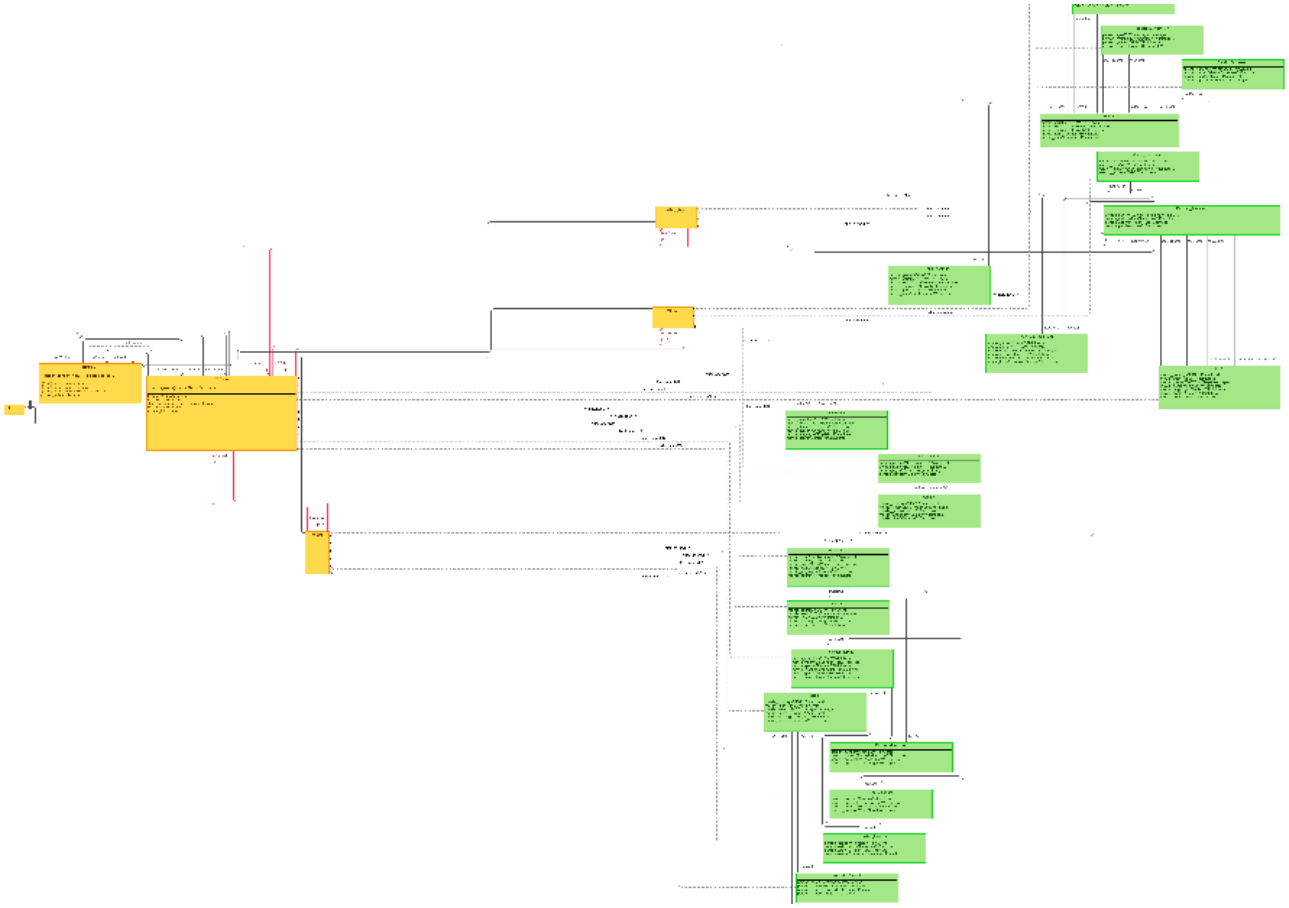
I. Class Hierarchy Graph



II. Class Relationship Graph



III. Property Graph



Note: this diagram's .png will be uploaded to our github (as property graph.png) since it was way too big to show in a word doc.

TEST SPARQL QUERIES AND THEIR OUTPUT

List the name of all Thriller movies. For each one, display its director.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?directorName
WHERE {
  ?movie rdf:type :movie ;
         :movie_genre "Thriller" ;
         :movie_title ?movieTitle .
  ?movie :hasDirector ?director .
  ?director :person_name ?directorName
.
}
```

Movie Title	Director Name
Kill Bill	Quentin Tarantino
The Irishman	Martin Scorsese

List the male actors in the movie in a specific film (Titanic).

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName
WHERE {
  ?movie rdf:type :movie ;
         :movie_title "Titanic" ;
         :hasActor ?actor .
  ?actor rdf:type :actor ;
         :person_gender "Male" ;
         :person_name ?actorName .
}
```

Actor
Leonardo Dicaprio

List all the movies written by a specific writer (Neil Gaiman).

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle
WHERE {
  ?writer rdf:type :writer ;
          :person_name "Neil Gaiman" .
  ?movie rdf:type :movie ;
         :hasWriter ?writer ;
         :movie_title ?movieTitle .
}
```

Movies
Coraline

A query with ORDER BY (age).

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName ?age
WHERE {
  ?actor rdf:type :actor ;
         :person_name ?actorName ;
         :person_age ?age .
}
ORDER BY DESC(?age)
```

Actor	Age
Quentin Tarantino	61
Robert Downey Jr	59
Leonardo Dicaprio	49
Kate Winslet	48
Rachel McAdams	45
Zoe Saldana	45
Jesse Eisenberg	40
Jennifer Lawrence	33

A query with FILTER (released after the year 2010).

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?releaseYear
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle ;
         :movie_year ?releaseYear .
  FILTER (?releaseYear > 2010)
}
```

Title	Release Year
Hunger Games	2012
Now You See Me	2013
The Irishman	2019

SNAPSHOT OF THE INTERFACE (GUI)

MANIPULATING THE ONTOLOGY (PART IV of the Project Description Doc)

I. Create a program that loads the ontology and displays all the Persons without using queries, without inference.

```
from rdflib import Graph, URIRef, RDF

g = Graph()
g.parse("movie5.owl", format='ttl')

print("Persons:")
for s, p, o in g.triples((None, RDF.type, None)):
    if o.endswith("actor") or o.endswith("director") or o.endswith("writer"):
```

```
        print(s)
Persons:
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jennifer_Lawrence
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jesse_Eisenberg
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Kate_Winslet
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Leonardo_Dicaprio
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Rachel_McAdams
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Robert_Downey_Jr
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Zoe_Saldana
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jesse_Eisenberg
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Neil_Gaiman
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Stan_Lee
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jon_Faverau
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Martin_Scorsese
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Robert_Downey_Jr
```

II. Create a program that loads the ontology and displays all the Persons using a query, without inference.

```
from rdflib import Graph, URIRef

g = Graph()
g.parse("movie5.owl", format="turtle")

query = """
    SELECT DISTINCT ?individual
    WHERE {
        {?individual a :writer .}
        UNION
        {?individual a :actor .}
        UNION
        {?individual a :director .}
    }
"""

results = g.query(query)

print("Persons")
for row in results:
    individual = row["individual"]
    print(individual)
```

```
Persons
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jesse_Eisenberg
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Neil_Gaiman
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Stan_Lee
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jennifer_Lawrence
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Kate_Winslet
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Leonardo_Dicaprio
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Rachel_McAdams
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Robert_Downey_Jr
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Zoe_Saldana
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jon_Faverau
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Martin_Scorsese
```

III. Create a program that loads the ontology and displays all the Actors without queries, using inference.

```
from rdflib import Graph, RDF
from owlrl import DeductiveClosure, OWLRL_Semantics

g = Graph()
g.parse("movie5.owl", format='ttl')

DeductiveClosure(OWLRL_Semantics).expand(g)

print("Persons:")
for s, p, o in g.triples((None, RDF.type, None)):
    if o.endswith("actor"):
        print(s)
```

```
Persons:
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jennifer_Lawrence
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Jesse_Eisenberg
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Kate_Winslet
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Leonardo_Dicaprio
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Rachel_McAdams
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Robert_Downey_Jr
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Zoe_Saldana
```

V. Create a program that displays all persons that are actors and directors. Do this using a rule that defines a new class ActorDirector.

```
from rdflib import Graph, URIRef
```

```python
g = Graph()
g.parse("movie5.owl", format="turtle")

query = """
    SELECT DISTINCT ?ActorDirector
    WHERE {

        {?ActorDirector a :actor .}

        {?ActorDirector a :director .}
    }
"""

results = g.query(query)

print("ActorDirector:")
for row in results:
    ActorDirector = row["ActorDirector"]
    print(ActorDirector)
```

```
ActorDirector:
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Quentin_Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/Robert_Downey_Jr
```

VI. Specify 3 different rules and implement them in a program.

```python
from rdflib import Graph, URIRef

g = Graph()
g.parse("movie5.owl", format="turtle")

# Query for movies made in america
query1 = """
    SELECT DISTINCT ?Movies
    WHERE {
        {?Movies a :movie .}

        {?Movies :movie_country "USA" .}
    }
"""
```

```
# Query for Action movies

query2 = """
    SELECT DISTINCT ?Movies
    WHERE {
        {?Movies a :movie .}

        {?Movies :movie_genre "Action" .}
    }
"""
# Query for movies involving quentin tarantino

query3 = """
    SELECT DISTINCT ?Movies
    WHERE {
        {?Movies a :movie .}
        {?Movies :hasActor :Quentin_Tarantino .}
        UNION
        {?Movies :hasDirector :Quentin_Tarantino .}
        UNION
        {?Movies :hasWriter :Quentin_Tarantino}
    }
"""

results1 = g.query(query1)
results2 = g.query(query2)
results3 = g.query(query3)

print("Movies made in America")
for row in results1:
    movie = row["Movies"]
    print(movie)

print("Action movies")
for row in results2:
    movie = row["Movies"]
    print(movie)

print("Movies involving Quentin Tarantino")
for row in results3:
    movie = row["Movies"]
    print(movie)
```

```
Movies made in America
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/avatar
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/coraline
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/hunger_games
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/ironman
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/kill_bill
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/now_you_see_me
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/pulp_fiction
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/the_irishman
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/the_notebook
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/titanic
Action movies
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/hunger_games
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/ironman
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/kill_bill
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/pulp_fiction
Movies involving Quentin Tarantino
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/kill_bill
http://www.semanticweb.org/noorh/ontologies/2024/4/untitled-ontology-6/pulp_fiction
```

MAIN 10 SPARQL QUERIES (PART III of the Project Description Doc)

I. List the instances of the class Actor

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actor
WHERE {
  ?actor rdf:type :actor .
}
```

II. List the instances of the class writer

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?writer
WHERE {
  ?writer rdf:type :writer .
}
```

III. List the instances of the class director

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>
```

```
SELECT ?director
WHERE {
  ?director rdf:type :director .
}
```

IV. List the name of all Thriller movies. For each one, display its director.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?directorName
WHERE {
  ?movie rdf:type :movie ;
         :movie_genre "Thriller" ;
         :movie_title ?movieTitle .
  ?movie :hasDirector ?director .
  ?director :person_name ?directorName .
}
```

V. List the name of all Crime Thriller movies.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle
WHERE {
  ?movie rdf:type :movie ;
         :movie_genre "Crime" , "Thriller" ;
         :movie_title ?movieTitle .
}
```

VI. List the male actors in the movie in specific film.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName
WHERE {
  ?movie rdf:type :movie ;
         :movie_title "Titanic" ;
         :hasActor ?actor .
  ?actor rdf:type :actor ;
         :person_gender "Male" ;
         :person_name ?actorName .
}
```

VII. How many movies have both "Action" and "Thriller" as genres?

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT (COUNT(?movie) AS ?count)
WHERE {
  ?movie rdf:type :movie ;
         :movie_genre "Action" , "Thriller" .
}
```

VIII. List all the movies written by a specific writer.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle
WHERE {
  ?writer rdf:type :writer ;
          :person_name "Neil Gaiman" .
  ?movie rdf:type :movie ;
         :hasWriter ?writer ;
         :movie_title ?movieTitle .
}
```

IX. Find movies with a certain language.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle
WHERE {
  ?movie rdf:type :movie ;
         :movie_language "English" ;
         :movie_title ?movieTitle .
}
```

X. List the name of Actors older than 51 years.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName
WHERE {
  ?actor rdf:type :actor ;
         :person_age ?age ;
```

```
        :person_name ?actorName .
  FILTER (?age > 51)
}
```

PROPOSED EXTRA 10 QUERIES (PART III of the Project Description Doc)

I. A query that contains at least 2 Optional Graph Patterns.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?directorName ?writerName
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle .
  OPTIONAL {
    ?movie :hasDirector ?director .
    ?director :person_name ?directorName .
  }
  OPTIONAL {
    ?movie :hasWriter ?writer .
    ?writer :person_name ?writerName .
  }
}
```

II. A query that contains at least 2 alternatives and conjunctions.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?actorName ?directorName
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle .
  {
    ?movie :hasActor ?actor .
    ?actor :person_name ?actorName .
  }
  UNION
  {
    ?movie :hasDirector ?director .
    ?director :person_name ?directorName .
  }
}
```

III. A query that contains a CONSTRUCT query form.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

CONSTRUCT {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle ;
         :hasActor ?actor ;
         :hasDirector ?director .
  ?actor rdf:type :actor ;
         :person_name ?actorName .
  ?director rdf:type :director ;
            :person_name ?directorName .
}
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle .
  OPTIONAL {
    ?movie :hasActor ?actor .
    ?actor :person_name ?actorName .
  }
  OPTIONAL {
    ?movie :hasDirector ?director .
    ?director :person_name ?directorName .
  }
}
```

IV. A query that contains an ASK query form.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

ASK {
  ?movie rdf:type :movie ;
         :movie_title "Avatar" .
}
```

V. A query that contains a DESCRIBE query form.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

DESCRIBE ?movie
WHERE {
  ?movie rdf:type :movie ;
         :movie_title "Avatar" .
}
```

VI. A query with GROUP BY and HAVING.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?genre (COUNT(?movie) AS ?movieCount)
WHERE {
  ?movie rdf:type :movie ;
         :movie_genre ?genre .
}
GROUP BY ?genre
HAVING (COUNT(?movie) > 2)
```

VII. A query with ORDER BY.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName ?age
WHERE {
  ?actor rdf:type :actor ;
         :person_name ?actorName ;
         :person_age ?age .
}
ORDER BY DESC(?age)
```

VIII. A query with LIMIT and OFFSET.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?actorName
WHERE {
  ?actor rdf:type :actor ;
         :person_name ?actorName .
}
LIMIT 5
OFFSET 2
```

IX. A query with FILTER.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle ?releaseYear
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle ;
         :movie_year ?releaseYear .
```

```
  FILTER (?releaseYear > 2010)
}
```

X. A subquery.

```
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/noorh/ontologies/2024/4/untitled-
ontology-6/>

SELECT ?movieTitle
WHERE {
  ?movie rdf:type :movie ;
         :movie_title ?movieTitle ;
         :hasDirector ?director .
  {
    SELECT ?director
    WHERE {
      ?director rdf:type :director .
    }
  }
}
```