Assignment 2

by Lukas Hofstetter - 51820745

⚠️⚠️⚠️ One-click-coopy version hosted on: ⚠️⚠️⚠️

```
https://github.com/hofslu/ISS-A2_Hofstetter_51820745_Report/blob/main
/A2_Hofstetter_51820745_Report.md
```

Task 1

SELECT queries – please implement one of these queries:

Q1: Return all films

```
PREFIX : <http://semantics.id/ns/example/film#>
SELECT ?film WHERE {
    ?film a :Film .
}
```

	film
1	http://semantics.id/ns/example#film_1
2	http://semantics.id/ns/example#film_2
3	http://semantics.id/ns/example#film_3
4	http://semantics.id/ns/example#film_4
5	http://semantics.id/ns/example#film_5

Q2: Return all films with their title

ASK queries – please implement one of these queries:

Q3: Is there a film named “Dune”?

```
PREFIX : <http://semantics.id/ns/example/film#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
ASK WHERE {
    ?film a :Film ;
          rdfs:label "Dune" .
```

```
}
```

YES

Q4: Is there a film named "Dune" released before 1984?

DESCRIBE queries – please implement one of these queries:

Q5: Give me all information about "Dune" movie released in 1984

```
PREFIX : <http://semantics.id/ns/example/film#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
DESCRIBE ?film WHERE {
    ?film a :Film ;
          rdfs:label "Dune" ;
          :releaseYear "1984"^^xsd:integer .
}
```

	subject	predicate	object
1	http://semantics.id/ns/example#film_5	rdf:type	:Film
2	http://semantics.id/ns/example#film_5	rdf:type	:Artwork
3	http://semantics.id/ns/example#film_5	rdf:type	owl:NamedIndividual
4	http://semantics.id/ns/example#film_5	rdfs:label	"Dune"
5	http://semantics.id/ns/example#film_5	:hasActor	:kyle_maclachlan
6	http://semantics.id/ns/example#film_5	:hasPerformer	:kyle_maclachlan
7	http://semantics.id/ns/example#film_5	:hasGenre	:genre_science_fiction
8	http://semantics.id/ns/example#film_5	:releaseYear	"1984"^^xsd:integer

Q6: Give me all information about actors which are playing in "Dune" released in 1984

SPARQL CONSTRUCT queries – please implement one of these queries and additionally build a query of your own with code Q9

Q7: Return all writers and the film studios for which they have worked

```
PREFIX ex: <http://semantics.id/ns/example/film#>
```

```
CONSTRUCT {
    ?writer ex:hasWorkedFor ?studio .
} WHERE {
    ?film ex:hasScriptWriter ?writer .
    ?film ex:hasFilmStudio ?studio .
}
```

	subject	predicate	object
1	http://semantics.id/ns/example#writer_1	:hasWorkedFor	http://semantics.id/ns/example#WaltDisneyPictures
2	http://semantics.id/ns/example#writer_2	:hasWorkedFor	http://semantics.id/ns/example#WaltDisneyPictures
3	http://semantics.id/ns/example#julius_avery	:hasWorkedFor	http://semantics.id/ns/example#EntertainmentOne
4	http://semantics.id/ns/example#writer_3	:hasWorkedFor	http://semantics.id/ns/example#EntertainmentOne
5	http://semantics.id/ns/example#writer_4	:hasWorkedFor	http://semantics.id/ns/example#WaltDisneyPictures
6	http://semantics.id/ns/example#writer_5	:hasWorkedFor	http://semantics.id/ns/example#WaltDisneyPictures

Q8: Return all actors and directors who know each other from movies

Q9: Constructing playedInGenre Relationship

```
PREFIX ex: <http://semantics.id/ns/example/film#>

CONSTRUCT {
    ?actor ex:playedInGenre ?genre .
} WHERE {
    ?film ex:hasActor ?actor ;
          ex:hasGenre ?genre .
}
```

	subject	predicate	object
1	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_animation
2	http://semantics.id/ns/example#hayley_atwell	:playedInGenre	:genre_animation
3	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_family
4	http://semantics.id/ns/example#hayley_atwell	:playedInGenre	:genre_family
5	http://semantics.id/ns/example#alicia_vikander	:playedInGenre	:genre_action
6	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_action
7	http://semantics.id/ns/example#alicia_vikander	:playedInGenre	:genre_drama
8	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_drama
9	http://semantics.id/ns/example#dan_stevens	:playedInGenre	:genre_family
10	http://semantics.id/ns/example#emma_watson	:playedInGenre	:genre_family
11	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_family
12	http://semantics.id/ns/example#dan_stevens	:playedInGenre	:genre_romance
13	http://semantics.id/ns/example#emma_watson	:playedInGenre	:genre_romance
14	http://semantics.id/ns/example#ewan_mcgregor	:playedInGenre	:genre_romance
15	:zendaya	:playedInGenre	:genre_science_fiction
16	:kyle_maclachlan	:playedInGenre	:genre_science_fiction

FILTER queries – please implement one of these queries and additionally build a query of your own with code Q13

Q10: Select film studios that were established after 1960

```
PREFIX ex: <http://semantics.id/ns/example/film#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?studio ?date WHERE {
    ?studio a ex:FilmStudio ;
            ex:establishedDate ?date .
    FILTER (?date >= "1960-12-31"^^xsd:date)
}
```

	studio	date
1	http://semantics.id/ns/example#EntertainmentOne	"1973-01-01"^^xsd:date
2	http://semantics.id/ns/example#WaltDisneyPictures	"1985-06-21"^^xsd:date

Q11: Select unique name of actors who play in movies between 2014 and 2020

Q12: Select name of actors who play in movies released after 2016 and with title containing string “Beauty”

Q13 Actors in Family Genre Films Born After 1980

```
PREFIX ex: <http://semantics.id/ns/example/film#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?actorName ?birthdate WHERE {
    ?film ex:hasActor ?actor ;
          ex:hasGenre ex:genre_family .
    ?actor ex:fullName ?actorName ;
           ex:dateOfBirth ?birthdate .
    FILTER (?birthdate > "1980-01-01"^^xsd:date)
}
```

	actorName	birthdate
1	"Hayley Atwell"	"1982-04-05"^^xsd:date
2	"Dan Stevens"	"1982-10-10"^^xsd:date
3	"Emma Watson"	"1990-04-15"^^xsd:date

ORDER and GROUP queries – please implement one of these queries and additionally build a query of your own with code Q16

Q14: Select name of actors who plays in movies ordered by birthdate (ascending)

```
PREFIX ex: <http://semantics.id/ns/example/film#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?actorName ?actor ?birthdate WHERE {
    ?film ex:hasActor ?actor .
    ?actor a ex:Actor ;
           ex:dateOfBirth ?birthdate ;
           ex:fullName ?actorName .
}
ORDER BY ?birthdate
```

	actorName	actor	birthdate
1	"Kyle Merritt MacLachlan"	:kyle_maclachlan	"1959-02-22"^^xsd:date
2	"Ewan McGregor"	http://semantics.id/ns/example#ewan_mcgregor	"1971-03-31"^^xsd:date
3	"Hayley Atwell"	http://semantics.id/ns/example#hayley_atwell	"1982-04-05"^^xsd:date
4	"Dan Stevens"	http://semantics.id/ns/example#dan_stevens	"1982-10-10"^^xsd:date
5	"Emma Watson"	http://semantics.id/ns/example#emma_watson	"1990-04-15"^^xsd:date
6	"Zendaya Maree Stoermer Coleman"	:zendaya	"1996-09-01"^^xsd:date

Q15: Count the average number of ScriptWriters involved in each movie!

Q16 Count Films per Actor

```
PREFIX ex: <http://semantics.id/ns/example/film#>

SELECT ?actor (COUNT(?film) AS ?numberOfFilms) WHERE {
    ?film ex:hasActor ?actor .
} GROUP BY ?actor
ORDER BY DESC(?numberOfFilms)
```

	actor	numberOfFilms
1	http://semantics.id/ns/example#ewan_mcgregor	"3"^^xsd:integer
2	http://semantics.id/ns/example#hayley_atwell	"1"^^xsd:integer
3	http://semantics.id/ns/example#alicia_vikander	"1"^^xsd:integer
4	http://semantics.id/ns/example#dan_stevens	"1"^^xsd:integer
5	http://semantics.id/ns/example#emma_watson	"1"^^xsd:integer
6	:zendaya	"1"^^xsd:integer
7	:kyle_maclachlan	"1"^^xsd:integer

UNION queries – please implement this query without utilizing inference mechanisms:

Q17: List all actors and crews together with the title of movies that they are involved in, ordered by their name.

```
PREFIX ex: <http://semantics.id/ns/example/film#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?personName ?movieTitle WHERE {
    {
        ?film ex:hasActor ?person .
        ?person ex:fullName ?personName .
    }
    UNION
    {
        ?film ex:hasCrew ?person .
        ?person ex:fullName ?personName .
    }
    ?film rdfs:label ?movieTitle .
}
ORDER BY ?personName
```

	personName	movieTitle
1	"Alicia Vikander"	"Son of a Gun"
2	"Dan Stevens"	"Beauty and the Beast"
3	"Emma Watson"	"Beauty and the Beast"
4	"Ewan McGregor"	"Christopher Robin"
5	"Ewan McGregor"	"Son of a Gun"
6	"Ewan McGregor"	"Beauty and the Beast"
7	"Hayley Atwell"	"Christopher Robin"
8	"Kyle Merritt MacLachlan"	"Dune"
9	"Zendaya Maree Stoermer Coleman"	"Dune"

Task 2

dpedia Querries

http://dbpedia.org/sparql

Q18: How many South Korean movies are listed in DBPedia?

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT (COUNT(?movie) AS ?numberOfMovies) WHERE {
    ?movie a dbo:Film ;
           dbo:country dbr:South_Korea .
}
```

numberOfMovies
548

Q19: Find all movies released after year 2000.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?movie ?date
WHERE {
  ?movie a dbo:Film ;
         dbo:releaseDate ?date .
  FILTER (YEAR(?date) > 2000)
}
ORDER BY ?date
-- ORDER BY DESC(?date)
LIMIT 10
```

movie	date
http://dbpedia.org/resource/Great_Hotels	2001-01-01
http://dbpedia.org/resource/Konsento	2001-01-01
http://dbpedia.org/resource/Sleepless_(2001_film)	2001-01-05
http://dbpedia.org/resource/Hannah_and_Her_Brothers__Hana_a_jej_bratia__1	2001-01-11
http://dbpedia.org/resource/Rentun_Ruusu	2001-01-12

movie	date
http://dbpedia.org/resource/100_Years_(film)	2115-11-18
http://dbpedia.org/resource/Force_(film_series)	2024-08-15
http://dbpedia.org/resource/The_Nun_2	2023-09-08
http://dbpedia.org/resource/Gadar_2	2023-08-15
http://dbpedia.org/resource/Carry_On_Jatta_(film_series)	2023-06-29

Q20: Find all movies directed by Steven Spielberg where Tom Hanks is not playing

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?movie WHERE {
    ?movie a dbo:Film ;
           dbo:director dbr:Steven_Spielberg .
    MINUS { ?movie dbo:starring dbr:Tom_Hanks }
}
```

movie
http://dbpedia.org/resource/Schindler's_List
http://dbpedia.org/resource/Hook_(film)
http://dbpedia.org/resource/Close_Encounters_of_the_Third_Kind
http://dbpedia.org/resource/Empire_of_the_Sun_(film)
http://dbpedia.org/resource/Minority_Report_(film)
http://dbpedia.org/resource/Munich_(2005_film)
http://dbpedia.org/resource/The_Lost_World:_Jurassic_Park
http://dbpedia.org/resource/The_Sugarland_Express
http://dbpedia.org/resource/The_Terminal
http://dbpedia.org/resource/Lincoln_(film)

Additional Querries

Additionally, please propose and implement 2 complex queries of your own including (any combination of) FILTER, ORDER, GROUP, UNION. Name these Q21 and Q22

Q21 Count of Philosophers by Era (with More Than One Philosopher)

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?era (COUNT(?philosopher) AS ?numberOfPhilosophers) WHERE {
```

```
    ?philosopher a dbo:Philosopher ;
                 dbo:era ?era .
} GROUP BY ?era
HAVING (COUNT(?philosopher) > 1)
ORDER BY DESC(?numberOfPhilosophers)
```

era	numberOfPhilosophers
http://dbpedia.org/resource/Contemporary_philosophy	869
http://dbpedia.org/resource/20th-century_philosophy	583
http://dbpedia.org/resource/21st-century_philosophy	158
http://dbpedia.org/resource/19th-century_philosophy	143
http://dbpedia.org/resource/Medieval_philosophy	54
http://dbpedia.org/resource/18th-century_philosophy	43
http://dbpedia.org/resource/Ancient_philosophy	39
http://dbpedia.org/resource/Modern_philosophy	35
http://dbpedia.org/resource/Hellenistic_philosophy	27
http://dbpedia.org/resource/17th-century_philosophy	26

Q22 Analysis of International Collaboration in Space Missions Post-2000

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?missionType ?missionCategory (YEAR(SAMPLE(?missionDate)) AS
?missionYear) (COUNT(DISTINCT ?astronautNationality) AS
?numberOfCountriesOrAgencies) ?mission WHERE {
    {
        ?astronaut a dbo:Astronaut .
        ?mission a dbo:SpaceMission ;
                 dbp:missionType ?missionType ;
                 dbo:launchDate ?missionDate .
        ?astronaut dbo:mission ?mission .
        OPTIONAL { ?astronaut dbo:nationality ?nation . }
        OPTIONAL { ?astronaut dbo:employer ?agency . }
        BIND(COALESCE(?nation, ?agency) AS ?astronautNationality)
        FILTER (YEAR(?missionDate) > 2000)
        BIND("Crewed" AS ?missionCategory)
    }
    UNION
    {
        ?mission a dbo:SpaceMission ;
                 dbp:missionType ?missionType ;
                 dbo:launchDate ?missionDate .
        FILTER NOT EXISTS { ?mission dbo:crew ?crew }
        FILTER (YEAR(?missionDate) > 2000)
```

```
        BIND("Uncrewed" AS ?missionCategory)
    }
}
GROUP BY ?missionType ?mission ?missionCategory
HAVING (COUNT(DISTINCT ?astronautNationality) >= 0)
ORDER BY DESC(?missionYear)
```

missionType	missionCategory	missionYear	numberOfCountriesOrAgencies	mission
"Orbiter and Atmospheric probe"@en	Uncrewed	2029	0	http://dbpedia.org/resource/DAVINCI
"Atmospheric probe"@en	Uncrewed	2027	0	http://dbpedia.org/resource/Saturn_Atmospheric_Entry_Probe
"Heliosphere research"@en	Uncrewed	2025	0	http://dbpedia.org/resource/Interstellar_Mapping_and_Acceleration_Probe
"Reconnaissance, flyby of outer planets"@en	Uncrewed	2025	0	http://dbpedia.org/resource/Trident_(spacecraft)
"Space telescope"@en	Uncrewed	2025	0	http://dbpedia.org/resource/Spektr-UV
"Earth weather forecasting"@en	Uncrewed	2024	0	http://dbpedia.org/resource/GOES-U
"Europa reconnaissance"@en	Uncrewed	2024	0	http://dbpedia.org/resource/Europa_Clipper
http://dbpedia.org/resource/Planetary_science	Uncrewed	2023	0	http://dbpedia.org/resource/Jupiter_Icy_Moons_Explorer
"ISS crew transport"@en	Crewed	2023	1	http://dbpedia.org/resource/SpaceX_Crew-6
"ISS crew transport"@en	Uncrewed	2023	0	http://dbpedia.org/resource/SpaceX_Crew-6
"Asteroid orbiter"@en	Uncrewed	2023	0	http://dbpedia.org/resource/Psyche_(spacecraft)
http://dbpedia.org/resource/Earth_observation_satellite	Uncrewed	2023	0	http://dbpedia.org/resource/IMECE
"ISS resupply"@en	Uncrewed	2022	0	http://dbpedia.org/resource/Progress_MS-20
http://dbpedia.org/resource/Earth_observation	Uncrewed	2022	0	http://dbpedia.org/resource/EOS_02
http://dbpedia.org/resource/Earth_observation_satellite	Uncrewed	2022	0	http://dbpedia.org/resource/Time-Resolved_Observations_of_Precipitation_structure_and_storm_Intensity_with_a_Constellatic
"Lunar orbiter"@en	Uncrewed	2022	0	http://dbpedia.org/resource/Danuri
"Communications"@en	Uncrewed	2022	0	http://dbpedia.org/resource/HADES_(satellite)
"Crewed mission to ISS"@en	Crewed	2022	2	http://dbpedia.org/resource/Soyuz_MS-21
"ISS crew transport"@en	Crewed	2022	0	http://dbpedia.org/resource/SpaceX_Crew-5

Task 3

Q23: RDFS Domain Entailment

In Q23, we explore the concept of RDFS Domain Entailment. This entailment pattern is based on the idea that if a property has a defined domain, any resource that has this property implies that the resource is an instance of the domain class. For instance, if a property ex:hasPerformer is defined with the domain ex:Artwork, then any subject with this property can be inferred to be an instance of ex:Artwork. This query demonstrates how inference can extend the understanding of data, revealing implicit class memberships that are not explicitly stated but are logically consistent with the ontology's structure.

```
PREFIX ex: <http://semantics.id/ns/example/film#>

SELECT ?subject WHERE {
    ?subject ex:hasPerformer ?crew .
}
```

Without Inference Enabled: The query will return ?artwork that has a hasPerformer relationship without considering the class hierarchy. It means if some instances are not explicitly typed as Artwork but have

a hasPerformer relationship, they may not be returned.

	subject
1	:Belle

With Inference Enabled: This query will return all ?artwork that is related to ?performer through ex:hasPerformer. The inference will recognize that any ?artwork with a hasPerformer relationship is an instance of Artwork or its subclasses, even if it's not explicitly typed as such.

	subject
1	http://semantics.id/ns/example#film_1
2	http://semantics.id/ns/example#film_1
3	http://semantics.id/ns/example#film_2
4	http://semantics.id/ns/example#film_2
5	http://semantics.id/ns/example#film_3
6	http://semantics.id/ns/example#film_3
7	http://semantics.id/ns/example#film_3
8	:Belle
9	http://semantics.id/ns/example#film_4
10	http://semantics.id/ns/example#film_5

Q24: RDFS Sub-Property Entailment

Q24 is centered around the RDFS Sub-Property Entailment. In this pattern, if a property (say ex:hasLeadActor) is defined as a sub-property of another property (ex:hasActor), then all instances of the sub-property are also inferred to be instances of the super-property. This allows us to query for the super-property and retrieve results that include its sub-properties. The query showcases how this entailment enriches query results by capturing relationships that are not explicitly stated but are inherent in the ontology's property hierarchy

Therefor first a SubProperty needed to be created:

```
PREFIX ex: <http://semantics.id/ns/example/film#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

INSERT DATA {
    ex:hasLeadActor rdfs:subPropertyOf ex:hasActor .
}
```

and an instance added:

```
PREFIX ex: <http://semantics.id/ns/example/film#>

INSERT DATA {
    ex:film_1 ex:hasLeadActor ex:zendaya .
}
```

to check with:

```
PREFIX ex: <http://semantics.id/ns/example/film#>

SELECT ?film ?actor WHERE {
    ?film ex:hasActor ?actor .
}
```

Without Inference Enabled: The query will not include :zendaya for #film_1.

	film	actor
1	http://semantics.id/ns/example#film_1	http://semantics.id/ns/example#ewan_mcgregor
2	http://semantics.id/ns/example#film_1	http://semantics.id/ns/example#hayley_atwell
3	http://semantics.id/ns/example#film_2	http://semantics.id/ns/example#alicia_vikander
4	http://semantics.id/ns/example#film_2	http://semantics.id/ns/example#ewan_mcgregor
5	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#dan_stevens
6	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#emma_watson
7	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#ewan_mcgregor
8	http://semantics.id/ns/example#film_4	ex:zendaya
9	http://semantics.id/ns/example#film_5	ex:kyle_maclachlan

With Inference Enabled: The query returns #film_1 and :zendaya among other films and actors, showing the entailment from ex:hasLeadActor to ex:hasActor.

	film	actor
1	http://semantics.id/ns/example#film_1	http://semantics.id/ns/example#ewan_mcgregor
2	http://semantics.id/ns/example#film_1	http://semantics.id/ns/example#hayley_atwell
3	http://semantics.id/ns/example#film_2	http://semantics.id/ns/example#alicia_vikander
4	http://semantics.id/ns/example#film_2	http://semantics.id/ns/example#ewan_mcgregor
5	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#dan_stevens
6	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#emma_watson
7	http://semantics.id/ns/example#film_3	http://semantics.id/ns/example#ewan_mcgregor
8	http://semantics.id/ns/example#film_4	ex:zendaya
9	http://semantics.id/ns/example#film_5	ex:kyle_maclachlan
10	ex:film_1	ex:zendaya

Q25: OWL Class Hierarchy Entailment

Q25 focuses on demonstrating OWL Class Hierarchy Entailment. This entails that instances of a

subclass are also considered instances of its superclass. For instance, if ex:Actor is a subclass of ex:Performer, then all instances of ex:Actor are also inferred to be instances of ex:Performer. Our query illustrates this principle by selecting instances of a superclass and revealing how, with inference, it includes instances of its subclasses. This showcases the power of OWL reasoning in revealing implicit instance relationships based on class hierarchies.

```
PREFIX ex: <http://semantics.id/ns/example/film#>

SELECT ?instance WHERE {
    ?instance a ex:Performer .
}
```

Without Inference Enabled: The query will return only the direct instances of ex:Performer and not the instances of its subclasses (such as ex:Actor).

	instance
1	:alan_menken

With Inference Enabled: This query will return instances of ex:Performer and all of its subclasses (like ex:Actor). It illustrates how class hierarchy reasoning includes instances of subclasses under the superclass.

	instance
1	http://semantics.id/ns/example#alicia_vikander
2	http://semantics.id/ns/example#dan_stevens
3	http://semantics.id/ns/example#emma_watson
4	http://semantics.id/ns/example#ewan_mcgregor
5	http://semantics.id/ns/example#hayley_atwell
6	:zendaya
7	:kyle_maclachlan
8	:alan_menken