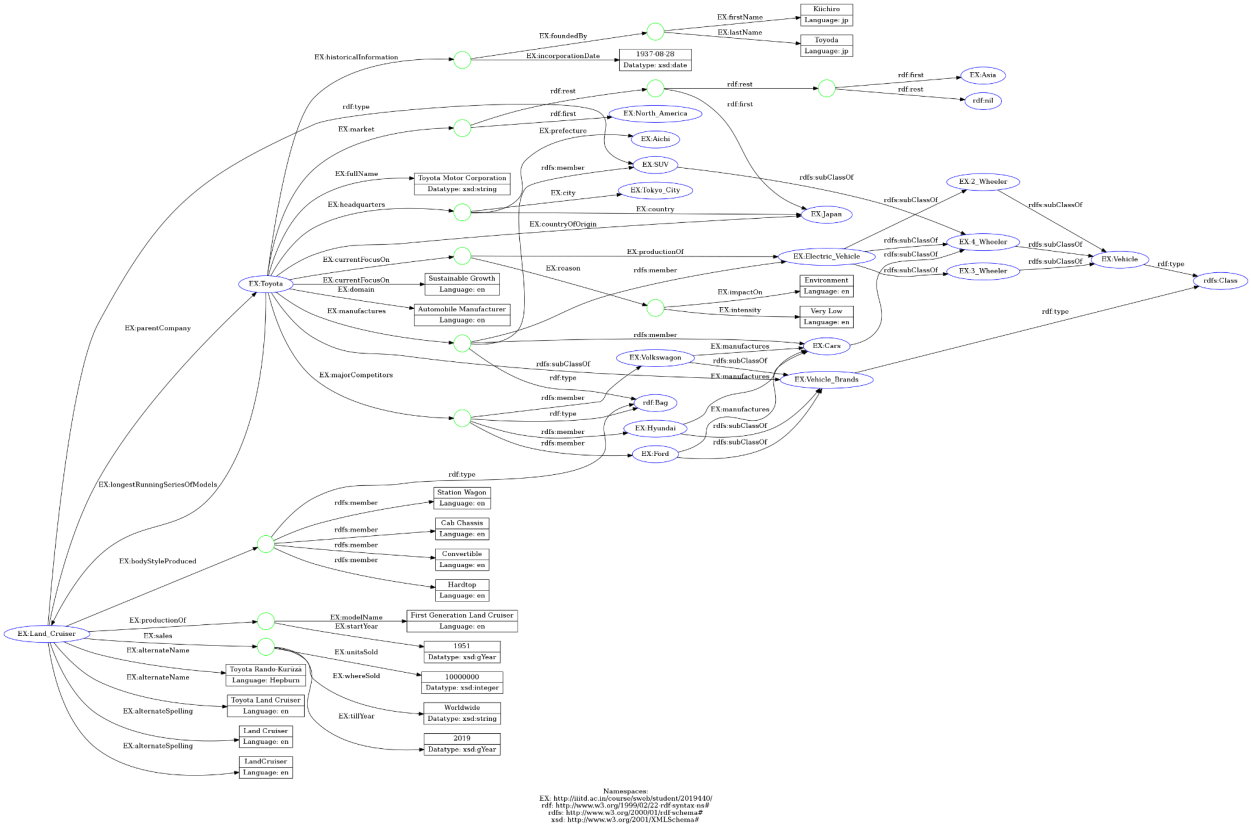
CSE 632 | Semantic Web

Assignment 1

Question 1 | RDF

Part a:

Graph:



[Link](#) to the graph

Code:

```
from rdflib import Graph, URIRef, Literal, BNode,
Namespace, XSD, collection, RDF, Bag, RDFS

# Defining the custom namespace
EX =
Namespace("http://iiitd.ac.in/course/sweb/student/2019440
/")

# Initialising the graph and binding the namespace with
it
g = Graph()
g.bind("EX", EX)

# Creating the required resources and blank nodes
LandCruiser = URIRef(EX.term("Land_Cruiser"))
Vehicle = URIRef(EX.term("Vehicle"))
VehicleBrands = URIRef(EX.term("Vehicle_Brands"))
FourWheeler = URIRef(EX.term("4_Wheeler"))
ThreeWheeler = URIRef(EX.term("3_Wheeler"))
TwoWheeler = URIRef(EX.term("2_Wheeler"))
ElectricVehicle = URIRef(EX.term("Electric_Vehicle"))
Cars = URIRef(EX.term("Cars"))
Suv = URIRef(EX.term("SUV"))
Toyota = URIRef(EX.term("Toyota"))
Japan = URIRef(EX.term("Japan"))
NorthAmerica = URIRef(EX.term("North_America"))
Asia = URIRef(EX.term("Asia"))
```

```
TokyoCity = URIRef(EX.term("Tokyo_City"))
Aichi = URIRef(EX.term("Aichi"))
Volkswagon = URIRef(EX.term("Volkswagon"))
Ford = URIRef(EX.term("Ford"))
Hyundai = URIRef(EX.term("Hyundai"))
EVReason = BNode()
LandCruiserSales = BNode()
LandCruiserFirstGenProduction = BNode()
ToyotaHeadquarters = BNode()
ToyotaHistory = BNode()
ToyotaFounder = BNode()
ToyotaFocussedProduction = BNode()
market1 = BNode()
market2 = BNode()
market3 = BNode()
bodyStyleBag = BNode()
competitorsBag = BNode()
manufactureBag = BNode()


# adding the triples to the graph
g.add((Vehicle, RDF.type, RDFS.Class))
g.add((VehicleBrands, RDF.type, RDFS.Class))
g.add((FourWheeler, RDFS.subClassOf, Vehicle))
g.add((ThreeWheeler, RDFS.subClassOf, Vehicle))
g.add((TwoWheeler, RDFS.subClassOf, Vehicle))
g.add((ElectricVehicle, RDFS.subClassOf, FourWheeler))
g.add((ElectricVehicle, RDFS.subClassOf, ThreeWheeler))
```

```
g.add((ElectricVehicle, RDFS.subClassOf, TwoWheeler))
g.add((Cars, RDFS.subClassOf, FourWheeler))
g.add((Suv, RDFS.subClassOf, FourWheeler))
g.add((LandCruiser, RDF.type, Suv))
g.add((LandCruiser, EX.alternateSpelling,
Literal("LandCruiser", lang="en")))
g.add((LandCruiser, EX.alternateSpelling, Literal("Land
Cruiser", lang="en")))
g.add((LandCruiser, EX.alternateName, Literal(
   "Toyota Rando-Kurūzā", lang="Hepburn")))
g.add((LandCruiser, EX.alternateName, Literal("Toyota
Land Cruiser", lang="en")))
g.add((LandCruiser, EX.parentCompany, Toyota))
g.add((Toyota, EX.fullName, Literal("Toyota Motor
Corporation", datatype=XSD.string)))
g.add((Toyota, EX.countryOfOrigin, Japan))
g.add((Toyota, EX.domain, Literal("Automobile
Manufacturer", lang="en")))
g.add((LandCruiser, EX.sales, LandCruiserSales))
g.add((LandCruiserSales, EX.tillYear, Literal(2019,
datatype=XSD.gYear)))
g.add((LandCruiserSales, EX.unitsSold, Literal(10000000,
datatype=XSD.integer)))
g.add((LandCruiserSales, EX.whereSold,
Literal("Worldwide", datatype=XSD.string)))
g.add((LandCruiser, EX.productionOf,
LandCruiserFirstGenProduction))
g.add((LandCruiserFirstGenProduction, EX.startYear,
```

```
      Literal(1951, datatype=XSD.gYear)))
g.add((LandCruiserFirstGenProduction, EX.modelName,
      Literal("First Generation Land Cruiser",
lang="en")))
g.add((LandCruiser, EX.bodyStyleProduced, bodyStyleBag))
g.add((bodyStyleBag, RDF.type, RDF.Bag))
g.add((bodyStyleBag, RDFS.member, Literal("Hardtop",
lang="en")))
g.add((bodyStyleBag, RDFS.member, Literal("Station
Wagon", lang="en")))
g.add((bodyStyleBag, RDFS.member, Literal("Cab Chassis",
lang="en")))
g.add((bodyStyleBag, RDFS.member, Literal("Convertible",
lang="en")))
# using bags
g.add((Toyota, EX.majorCompetitors, competitorsBag))
g.add((competitorsBag, RDF.type, RDF.Bag))
g.add((competitorsBag, RDFS.member, Ford))
g.add((competitorsBag, RDFS.member, Hyundai))
g.add((competitorsBag, RDFS.member, Volkswagon))
g.add((Toyota, EX.manufactures, manufactureBag))
g.add((manufactureBag, RDF.type, RDF.Bag))
g.add((manufactureBag, RDFS.member, Suv))
g.add((manufactureBag, RDFS.member, Cars))
g.add((manufactureBag, RDFS.member, ElectricVehicle))
g.add((Toyota, RDFS.subClassOf, VehicleBrands))
g.add((Ford, RDFS.subClassOf, VehicleBrands))
g.add((Hyundai, RDFS.subClassOf, VehicleBrands))
```

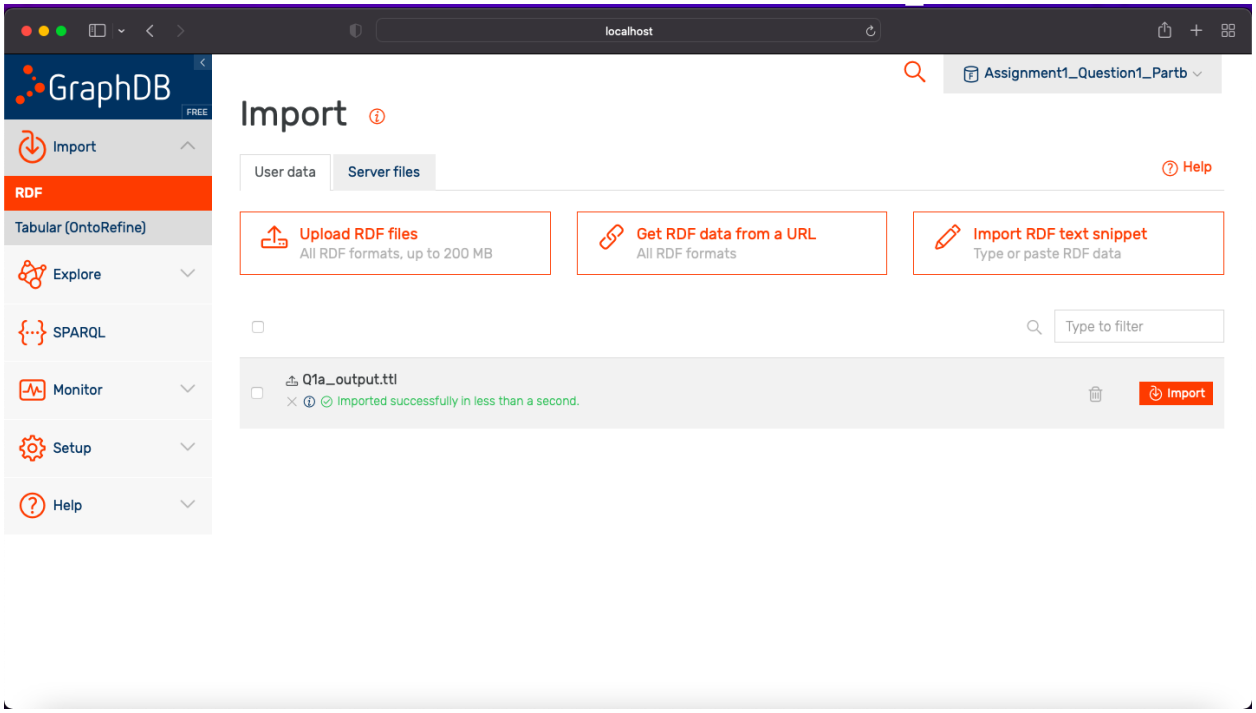
```
g.add((Volkswagon, RDFS.subClassOf, VehicleBrands))
g.add((Ford, EX.manufactures, Cars))
g.add((Hyundai, EX.manufactures, Cars))
g.add((Volkswagon, EX.manufactures, Cars))
g.add((Toyota, EX.headquarters, ToyotaHeadquarters))
g.add((ToyotaHeadquarters, EX.city, TokyoCity))
g.add((ToyotaHeadquarters, EX.prefecture, Aichi))
g.add((ToyotaHeadquarters, EX.country, Japan))
g.add((Toyota, EX.historicalInformation, ToyotaHistory))
g.add((ToyotaHistory, EX.foundedBy, ToyotaFounder))
g.add((ToyotaFounder, EX.firstName, Literal("Kiichiro",
lang="jp")))
g.add((ToyotaFounder, EX.lastName, Literal("Toyoda",
lang="jp")))
g.add((ToyotaHistory, EX.incorporationDate,
      Literal('1937-08-28', datatype=XSD.date)))
g.add((Toyota, EX.longestRunningSeriesOfModels,
LandCruiser))
g.add((Toyota, EX.currentFocusOn, Literal('Sustainable
Growth', lang="en")))
g.add((Toyota, EX.currentFocusOn,
ToyotaFocussedProduction))
g.add((ToyotaFocussedProduction, EX.productionOf,
ElectricVehicle))
g.add((ToyotaFocussedProduction, EX.reason, EVReason))
g.add((EVReason, EX.impactOn, Literal("Environment",
lang="en")))
```

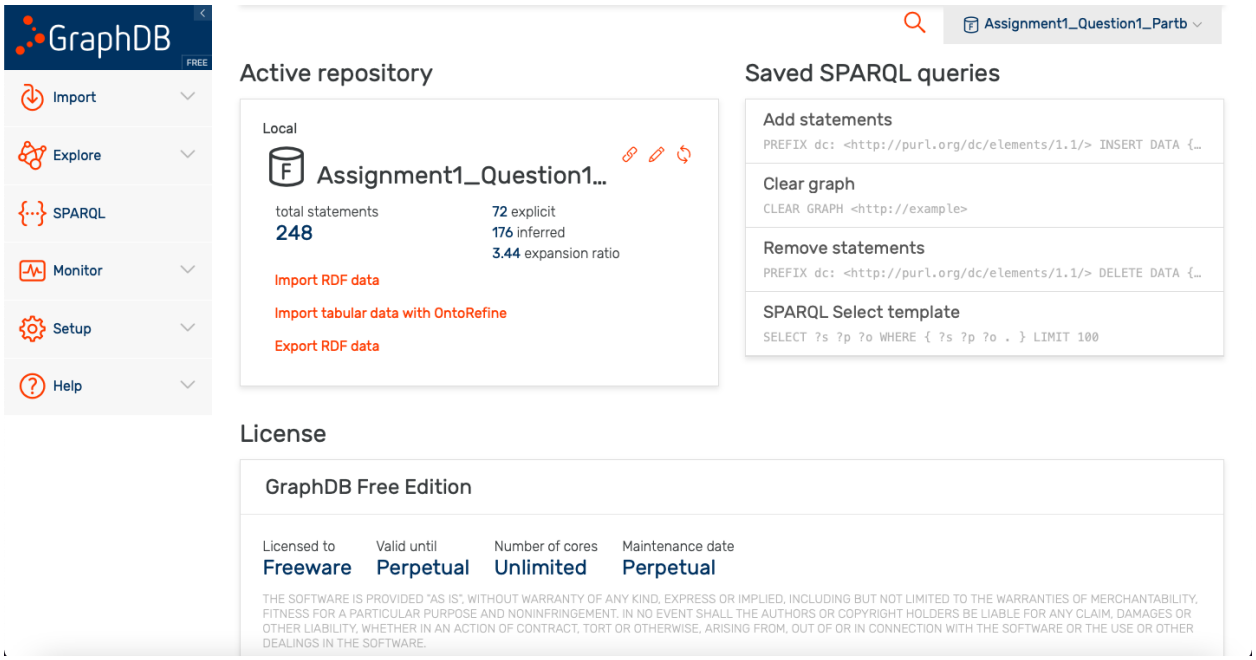
```
g.add((EVReason, EX.intensity, Literal("Very Low",
lang="en")))
# using list
g.add((Toyota, EX.market, market1))
g.add((market1, RDF.first, NorthAmerica))
g.add((market1, RDF.rest, market2))
g.add((market2, RDF.first, Japan))
g.add((market2, RDF.rest, market3))
g.add((market3, RDF.first, Asia))
g.add((market3, RDF.rest, RDF.nil))

# generating the output in .ttl format
g.serialize(destination='Q1a_output.ttl',
format='turtle')
```

Code Documentation:

- Custom namespace was defined
- The graph was initialized and the namespace was bound with it
- Required resources and blank nodes were created
- The relevant triples were added to the graph
- Graph was then serialized and the outputs were generated in .ttl format

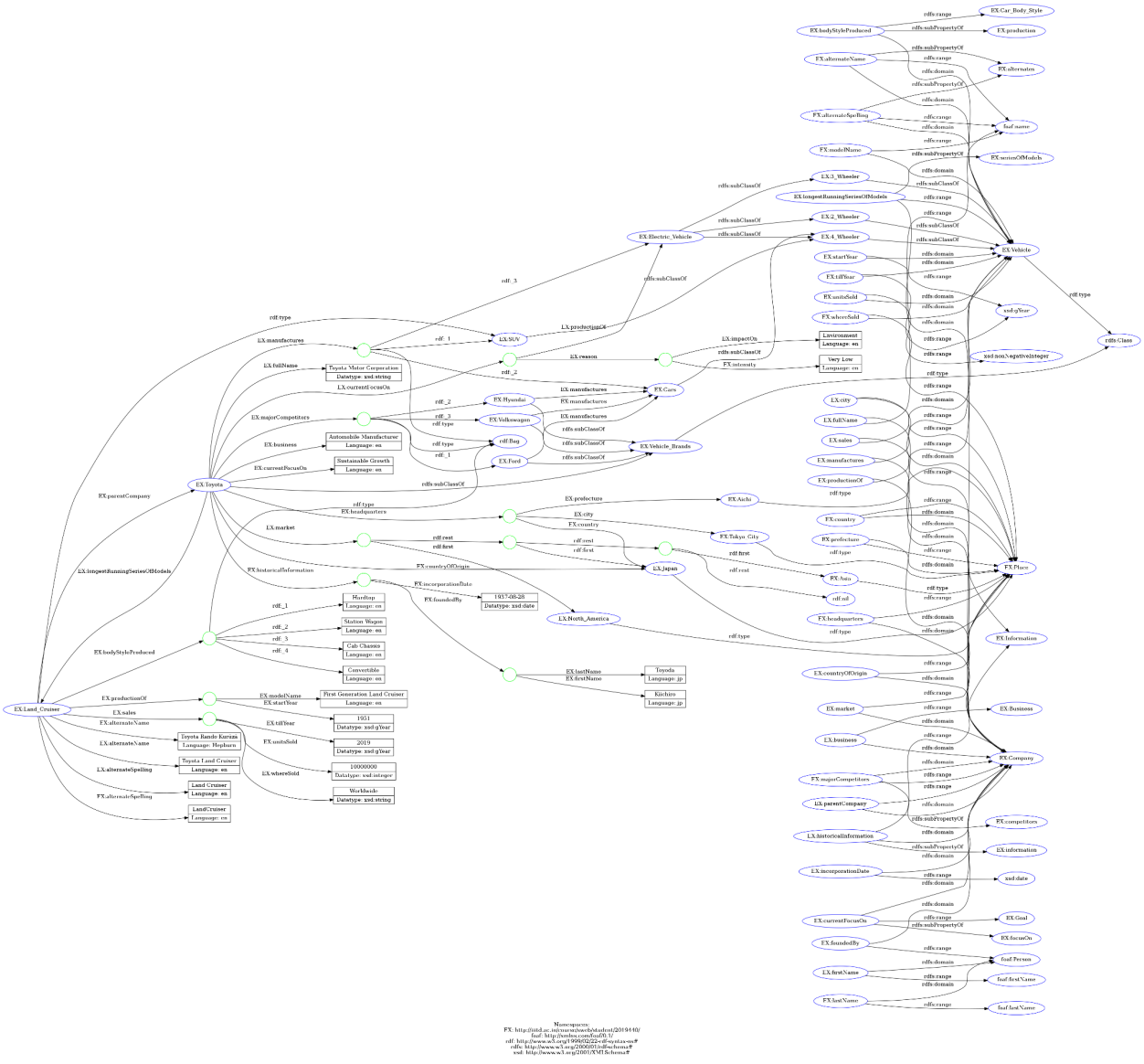
Part b:


localhost
GraphDB
FREE
Import
RDF
Tabular (OntoRefine)
Explore
SPARQL
Monitor
Setup
Help
Assignment1_Question1_Partb
Import
User data
Server files
Help
Upload RDF files
All RDF formats, up to 200 MB
Get RDF data from a URL
All RDF formats
Import RDF text snippet
Type or paste RDF data
Type to filter
Q1a_output.ttl
Imported successfully in less than a second.
Import


GraphDB
FREE
Import
Explore
SPARQL
Monitor
Setup
Help
Assignment1_Question1_Partb
Active repository
Local
Assignment1_Question1...
total statements
248
72 explicit
176 inferred
3.44 expansion ratio
Import RDF data
Import tabular data with OntoRefine
Export RDF data
Saved SPARQL queries
Add statements
PREFIX dc: <http://purl.org/dc/elements/1.1/> INSERT DATA {...
Clear graph
CLEAR GRAPH <http://example>
Remove statements
PREFIX dc: <http://purl.org/dc/elements/1.1/> DELETE DATA {...
SPARQL Select template
SELECT ?s ?p ?o WHERE { ?s ?p ?o . } LIMIT 100
License
GraphDB Free Edition
Licensed to
Freeware
Valid until
Perpetual
Number of cores
Unlimited
Maintenance date
Perpetual
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Part c:

Graph:



[Link](#) to the graph

Code:

```
from rdflib import Graph, URIRef, Literal, BNode,
Namespace, XSD, collection, RDF, Bag, RDFS, FOAF

# Defining the custom namespace
EX =
Namespace("http://iiitd.ac.in/course/sweb/student/2019440
/")

# Initialising the graph and binding the namespace with
it
g = Graph()
g.bind("EX", EX)

# Creating the required resources and blank nodes
LandCruiser = URIRef(EX.term("Land_Cruiser"))
Vehicle = URIRef(EX.term("Vehicle"))
VehicleBrands = URIRef(EX.term("Vehicle_Brands"))
FourWheeler = URIRef(EX.term("4_Wheeler"))
ThreeWheeler = URIRef(EX.term("3_Wheeler"))
TwoWheeler = URIRef(EX.term("2_Wheeler"))
ElectricVehicle = URIRef(EX.term("Electric_Vehicle"))
Cars = URIRef(EX.term("Cars"))
Suv = URIRef(EX.term("SUV"))
Toyota = URIRef(EX.term("Toyota"))
Japan = URIRef(EX.term("Japan"))
NorthAmerica = URIRef(EX.term("North_America"))
Asia = URIRef(EX.term("Asia"))
```

```
TokyoCity = URIRef(EX.term("Tokyo_City"))
Aichi = URIRef(EX.term("Aichi"))
Volkswagon = URIRef(EX.term("Volkswagon"))
Ford = URIRef(EX.term("Ford"))
Hyundai = URIRef(EX.term("Hyundai"))
Company = URIRef(EX.term("Company"))
Information = URIRef(EX.term("Information"))
Place = URIRef(EX.term("Place"))
Goal = URIRef(EX.term("Goal"))
Business = URIRef(EX.term("Business"))
CarBodyStyle = URIRef(EX.term("Car_Body_Style"))
EVReason = BNode()
LandCruiserSales = BNode()
LandCruiserFirstGenProduction = BNode()
ToyotaHeadquarters = BNode()
ToyotaHistory = BNode()
ToyotaFounder = BNode()
ToyotaFocussedProduction = BNode()
market1 = BNode()
market2 = BNode()
market3 = BNode()
bodyStyleBag = BNode()
competitorsBag = BNode()
manufactureBag = BNode()

# adding the triples to the graph
g.add((Vehicle, RDF.type, RDFS.Class))
g.add((VehicleBrands, RDF.type, RDFS.Class))
```

```
g.add((FourWheeler, RDFS.subClassOf, Vehicle))
g.add((ThreeWheeler, RDFS.subClassOf, Vehicle))
g.add((TwoWheeler, RDFS.subClassOf, Vehicle))
g.add((ElectricVehicle, RDFS.subClassOf, FourWheeler))
g.add((ElectricVehicle, RDFS.subClassOf, ThreeWheeler))
g.add((ElectricVehicle, RDFS.subClassOf, TwoWheeler))
g.add((Cars, RDFS.subClassOf, FourWheeler))
g.add((Suv, RDFS.subClassOf, FourWheeler))
g.add((LandCruiser, RDF.type, Suv))
g.add((LandCruiser, EX.alternateSpelling,
Literal("LandCruiser", lang="en")))
g.add((LandCruiser, EX.alternateSpelling, Literal("Land
Cruiser", lang="en")))
g.add((LandCruiser, EX.alternateName, Literal(
   "Toyota Rando-Kurūzā", lang="Hepburn")))
g.add((LandCruiser, EX.alternateName, Literal("Toyota
Land Cruiser", lang="en")))
g.add((LandCruiser, EX.parentCompany, Toyota))
g.add((Toyota, EX.fullName, Literal("Toyota Motor
Corporation", datatype=XSD.string)))
g.add((Toyota, EX.countryOfOrigin, Japan))
g.add((Toyota, EX.business, Literal("Automobile
Manufacturer", lang="en")))
g.add((LandCruiser, EX.sales, LandCruiserSales))
g.add((LandCruiserSales, EX.tillYear, Literal(2019,
datatype=XSD.gYear)))
g.add((LandCruiserSales, EX.unitsSold, Literal(10000000,
datatype=XSD.integer)))
```

```
g.add((LandCruiserSales, EX.whereSold,
Literal("Worldwide", datatype=XSD.string)))
g.add((LandCruiser, EX.productionOf,
LandCruiserFirstGenProduction))
g.add((LandCruiserFirstGenProduction, EX.startYear,
      Literal(1951, datatype=XSD.gYear)))
g.add((LandCruiserFirstGenProduction, EX.modelName,
      Literal("First Generation Land Cruiser",
lang="en")))
g.add((LandCruiser, EX.bodyStyleProduced, bodyStyleBag))
g.add((bodyStyleBag, RDF.type, RDF.Bag))
g.add((bodyStyleBag, RDF._1, Literal("Hardtop",
lang="en")))
g.add((bodyStyleBag, RDF._2, Literal("Station Wagon",
lang="en")))
g.add((bodyStyleBag, RDF._3, Literal("Cab Chassis",
lang="en")))
g.add((bodyStyleBag, RDF._4, Literal("Convertible",
lang="en")))
g.add((Toyota, EX.majorCompetitors, competitorsBag))
g.add((competitorsBag, RDF.type, RDF.Bag))
g.add((competitorsBag, RDF._1, Ford))
g.add((competitorsBag, RDF._2, Hyundai))
g.add((competitorsBag, RDF._3, Volkswagon))
g.add((Toyota, EX.manufactures, manufactureBag))
g.add((manufactureBag, RDF.type, RDF.Bag))
g.add((manufactureBag, RDF._1, Suv))
g.add((manufactureBag, RDF._2, Cars))
```

```
g.add((manufactureBag, RDF._3, ElectricVehicle))
g.add((Toyota, RDFS.subClassOf, VehicleBrands))
g.add((Ford, RDFS.subClassOf, VehicleBrands))
g.add((Hyundai, RDFS.subClassOf, VehicleBrands))
g.add((Volkswagon, RDFS.subClassOf, VehicleBrands))
g.add((Ford, EX.manufactures, Cars))
g.add((Hyundai, EX.manufactures, Cars))
g.add((Volkswagon, EX.manufactures, Cars))
g.add((Toyota, EX.headquarters, ToyotaHeadquarters))
g.add((ToyotaHeadquarters, EX.city, TokyoCity))
g.add((ToyotaHeadquarters, EX.prefecture, Aichi))
g.add((ToyotaHeadquarters, EX.country, Japan))
g.add((Toyota, EX.historicalInformation, ToyotaHistory))
g.add((ToyotaHistory, EX.foundedBy, ToyotaFounder))
g.add((ToyotaFounder, EX.firstName, Literal("Kiichiro",
lang="jp")))
g.add((ToyotaFounder, EX.lastName, Literal("Toyoda",
lang="jp")))
g.add((ToyotaHistory, EX.incorporationDate,
      Literal('1937-08-28', datatype=XSD.date)))
g.add((Toyota, EX.longestRunningSeriesOfModels,
LandCruiser))
g.add((Toyota, EX.currentFocusOn, Literal('Sustainable
Growth', lang="en")))
g.add((Toyota, EX.currentFocusOn,
ToyotaFocussedProduction))
g.add((ToyotaFocussedProduction, EX.productionOf,
ElectricVehicle))
```

```
g.add((ToyotaFocussedProduction, EX.reason, EVReason))
g.add((EVReason, EX.impactOn, Literal("Environment",
lang="en")))
g.add((EVReason, EX.intensity, Literal("Very Low",
lang="en")))
g.add((Toyota, EX.market, market1))
g.add((market1, RDF.first, NorthAmerica))
g.add((market1, RDF.rest, market2))
g.add((market2, RDF.first, Japan))
g.add((market2, RDF.rest, market3))
g.add((market3, RDF.first, Asia))
g.add((market3, RDF.rest, RDF.nil))
g.add((NorthAmerica, RDF.type, Place))
g.add((Asia, RDF.type, Place))
g.add((Japan, RDF.type, Place))
g.add((NorthAmerica, RDF.type, Place))
g.add((Aichi, RDF.type, Place))
g.add((TokyoCity, RDF.type, Place))
g.add((EX.firstName, RDFS.range, FOAF.firstName))
g.add((EX.firstName, RDFS.domain, FOAF.Person))
g.add((EX.lastName, RDFS.range, FOAF.lastName))
g.add((EX.lastName, RDFS.domain, FOAF.Person))
g.add((EX.foundedBy, RDFS.domain, Company))
g.add((EX.foundedBy, RDFS.range, FOAF.Person))
g.add((EX.incorporationDate, RDFS.domain, Company))
g.add((EX.incorporationDate, RDFS.range, XSD.date))
g.add((EX.historicalInformation, RDFS.domain, Company))
```

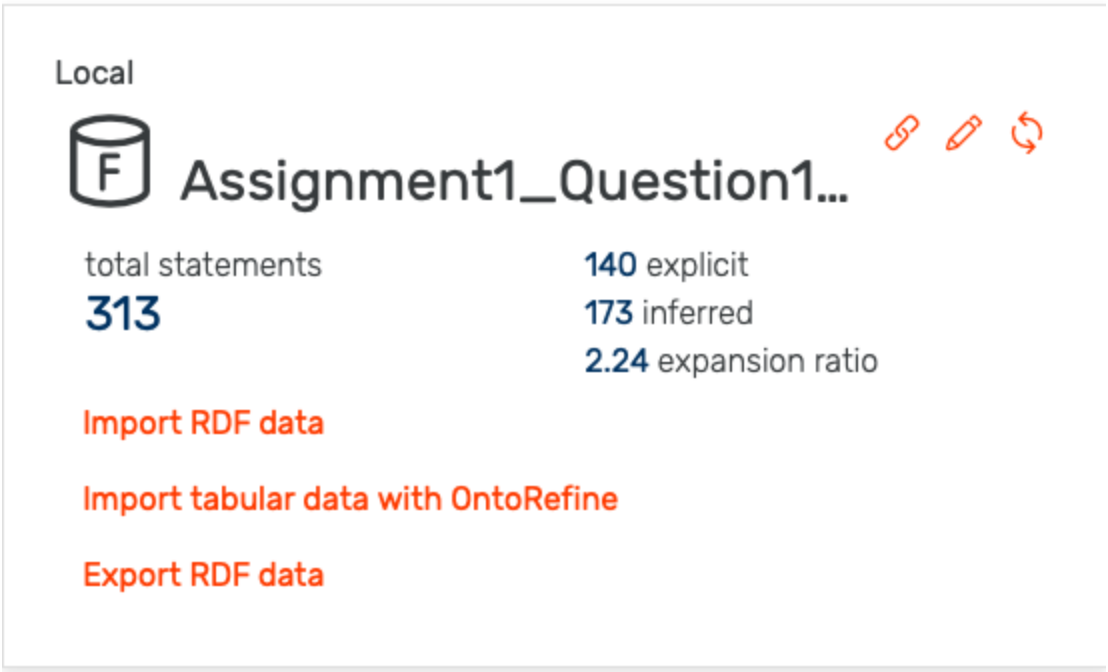
```
g.add((EX.historicalInformation, RDFS.range,
EX.Information))
g.add((EX.market, RDFS.domain, Company))
g.add((EX.market, RDFS.range, Place))
g.add((EX.prefecture, RDFS.domain, Place))
g.add((EX.prefecture, RDFS.range, Place))
g.add((EX.city, RDFS.domain, Place))
g.add((EX.city, RDFS.range, Place))
g.add((EX.country, RDFS.domain, Place))
g.add((EX.country, RDFS.range, Place))
g.add((EX.headquarters, RDFS.domain, Company))
g.add((EX.headquarters, RDFS.range, Place))
g.add((EX.fullName, RDFS.domain, Company))
g.add((EX.fullName, RDFS.range, FOAF.name))
g.add((EX.countryOfOrigin, RDFS.domain, Company))
g.add((EX.countryOfOrigin, RDFS.range, Place))
g.add((EX.currentFocusOn, RDFS.domain, Company))
g.add((EX.currentFocusOn, RDFS.range, Goal))
g.add((EX.productionOf, RDFS.domain, Company))
g.add((EX.productionOf, RDFS.range, Vehicle))
g.add((EX.business, RDFS.domain, Company))
g.add((EX.business, RDFS.range, Business))
g.add((EX.manufactures, RDFS.domain, Company))
g.add((EX.manufactures, RDFS.range, Vehicle))
g.add((EX.majorCompetitors, RDFS.domain, Company))
g.add((EX.majorCompetitors, RDFS.range, Company))
g.add((EX.parentCompany, RDFS.domain, Company))
g.add((EX.parentCompany, RDFS.range, Company))
```

```
g.add((EX.longestRunningSeriesOfModels, RDFS.domain,
Company))
g.add((EX.longestRunningSeriesOfModels, RDFS.range,
Vehicle))
g.add((EX.bodyStyleProduced, RDFS.domain, Vehicle))
g.add((EX.bodyStyleProduced, RDFS.range, CarBodyStyle))
g.add((EX.sales, RDFS.domain, Vehicle))
g.add((EX.sales, RDFS.range, Information))
g.add((EX.unitsSold, RDFS.domain, Vehicle))
g.add((EX.unitsSold, RDFS.range, XSD.nonNegativeInteger))
g.add((EX.whereSold, RDFS.domain, Vehicle))
g.add((EX.whereSold, RDFS.range, Place))
g.add((EX.tillYear, RDFS.domain, Vehicle))
g.add((EX.tillYear, RDFS.range, XSD.gYear))
g.add((EX.startYear, RDFS.domain, Vehicle))
g.add((EX.startYear, RDFS.range, XSD.gYear))
g.add((EX.modelName, RDFS.domain, Vehicle))
g.add((EX.modelName, RDFS.range, FOAF.name))
g.add((EX.alternateName, RDFS.domain, Vehicle))
g.add((EX.alternateName, RDFS.range, FOAF.name))
g.add((EX.alternateSpelling, RDFS.domain, Vehicle))
g.add((EX.alternateSpelling, RDFS.range, FOAF.name))
g.add((EX.historicalInformation, RDFS.subPropertyOf,
EX.information))
g.add((EX.currentFocusOn, RDFS.subPropertyOf,
EX.focusOn))
g.add((EX.majorCompetitors, RDFS.subPropertyOf,
EX.competitors))
```

```
g.add((EX.longestRunningSeriesOfModels,
RDFS.subPropertyOf, EX.seriesOfModels))
g.add((EX.bodyStyleProduced, RDFS.subPropertyOf,
EX.production))
g.add((EX.alternateSpelling, RDFS.subPropertyOf,
EX.alternates))
g.add((EX.alternateName, RDFS.subPropertyOf,
EX.alternates))

# generating the output in .ttl format
g.serialize(destination='Q1c_output.ttl',
format='turtle')
```

Code Documentation:

- Custom namespace was defined
- The graph was initialized and the namespace was bound with it
- Required resources and blank nodes were created
- The relevant triples were added to the graph
- The graph was then serialized and the outputs were generated in .ttl format



The entailed triples are available in the zip folder and have the name “query-results.csv”

Please note:

- I used GraphDB to upload the triples and to extract the inferred triples
- For showing the implicitly inferred triples, I have given an implicitly inferred triple and the entailment rule.
- Since there were a lot of triples that were inferred, it was not practically possible to show each and every one of them.
- As a result one example for each entailment rule in action is given.
- With the help of the rule and the graph obtained above, one can easily see how the triple has been inferred, as a result for the sake of simplicity, redundant and unnecessary explanations have been omitted.
- These explanations can easily be derived by having a look at the graph and the entailment rule mentioned along with the examples.

Some of the implicitly inferred triples:

1. **rdf:type rdf:type rdf:property**
   Rule used:

	If S contains	then S RDF entails
rdf2	xxx aaa yyy .	aaa rdf:type rdf:Property .

2. **xsd:nonNegativeInteger rdf:type rdfs:Datatype**

Rule Used:

	If S contains	then S RDFS entails
rdfs1	xxx aaa "sss"^^ddd .	ddd rdf:type rdfs:Datatype .

3. **EX:bodyStyleProduced rdfs:subPropertyOf EX:bodyStyleProduced**
   Rule Used:

	If S contains	then S RDFS entails
rdfs6	xxx rdf:type rdf:Property	xxx rdfs:subPropertyOf xxx

4. **EX:Land_Cruiser EX:alternates Toyota Rando-Kurūzā**
   Rule Used:

	If S contains	then S RDFS entails
rdfs7	aaa rdfs:subPropertyOf bbb . xxx aaa yyy .	xxx bbb yyy .

5. **EX:Land_Cruiser rdf:type EX:Vehicle**
   Rule Used:

	If S contains	then S RDFS entails
rdfs9	xxx rdfs:subClassOf yyy . zzz rdf:type xxx .	zzz rdf:type yyy .

6. **EX:Vehicle rdfs:subClassOf EX:Vehicle**
   Rule Used:

	If S contains	then S RDFS entails
rdfs10	xxx rdf:type rdfs:Class .	xxx rdfs:subClassOf xxx .

7. **EX:Electric_Vehicle rdfs:subClassOf EX:Vehicle**

Rule Used:

	If S contains	then S RDFS entails
rdfs11	xxx rdfs:subClassOf yyy . yyy rdfs:subClassOf zzz .	xxx rdfs:subClassOf zzz .

8. **rdf:XMLLiteral rdfs:subClassOf rdfs:Literal**
   Rule Used:

	If S contains	then S RDFS entails
rdfs13	xxx rdf:type rdfs:Datatype .	xxx rdfs:subClassOf rdfs:Literal .

Part d

Code:

```
from rdflib import Graph

# taking the input file and output format from the user

inputFile = input("Enter the input file name: ")
outputFormat = input("Enter the output format: ")

availFileFormats = {
   ".ttl": "turtle",
   ".nt": "nt",
   ".n3": "n3",
   ".rdf": "xml"
}

# Checking if the input file format and output file
format are available in the dictionary
inputFormat = inputFile[inputFile.rfind("."):]
if outputFormat[0] != '.':
   outputFormat = '.' + outputFormat
if inputFormat not in availFileFormats:
   print("Invalid input file")
   exit()
if outputFormat not in availFileFormats:
   print("Invalid output format")
   exit()
```

```
# creating a graph object and parsing it with the input
file
g = Graph()
g.parse(inputFile, availFileFormats[inputFormat])

# Serializing the graph and storing it in the required
file format
g.serialize(destination=inputFile[:inputFile.rfind(
   ".")] + outputFormat,
format=availFileFormats[outputFormat], encoding="utf-8")
```

The generated files are stored in the accompanying folder.

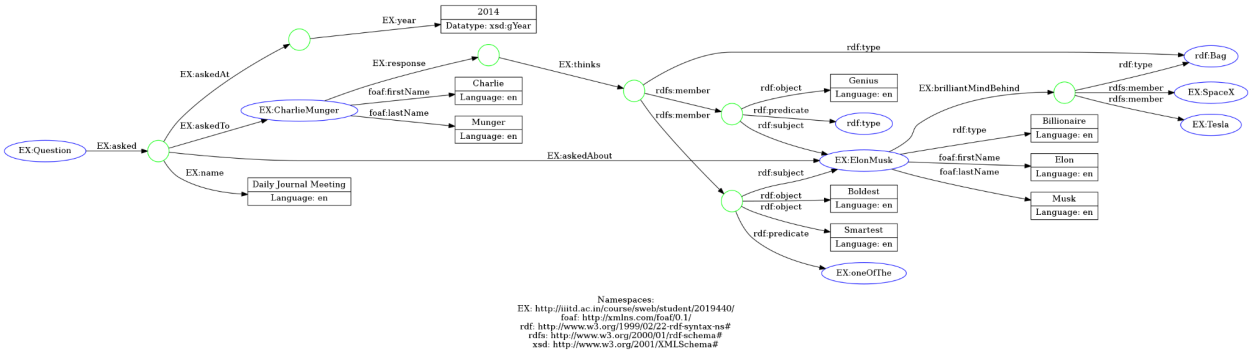
Code Documentation:

- input file and output format were taken from the user
- It was checked whether the input file format and output file format were available in the dictionary of available formats
- graph object was created and parsed with the input file
- the graph was serialized and stored it in the required file format

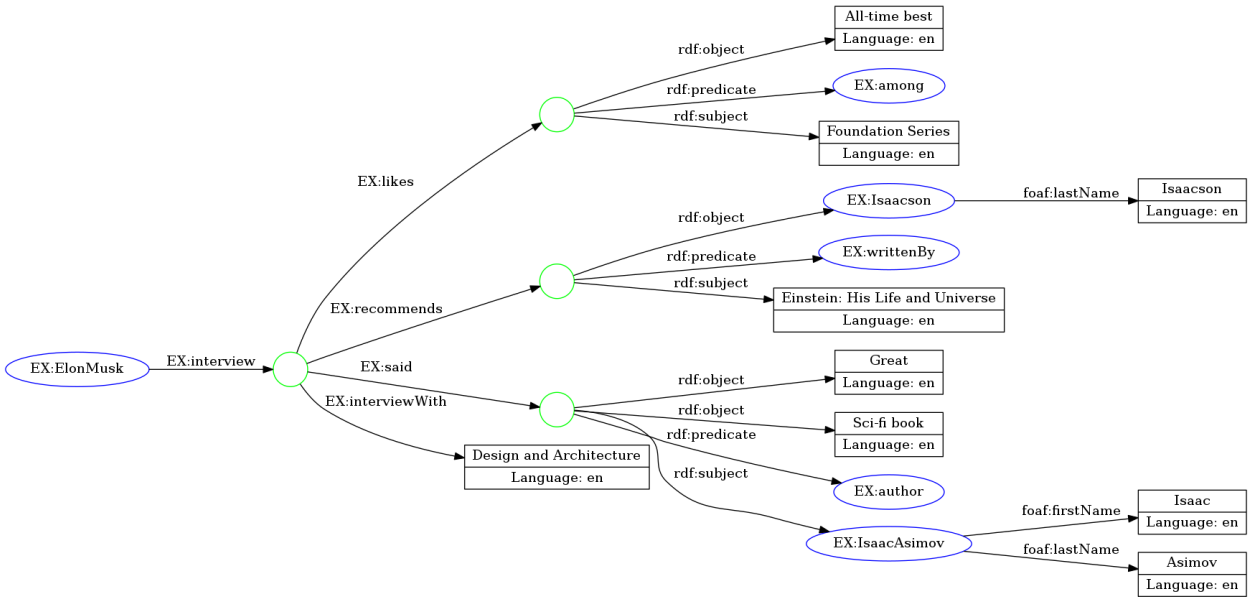
Question 2 | Reification

I used 2 different graphs to illustrate all the concepts.

1. **Named Graphs + n-ary properties + standard reification**
   Individual Named Graphs:

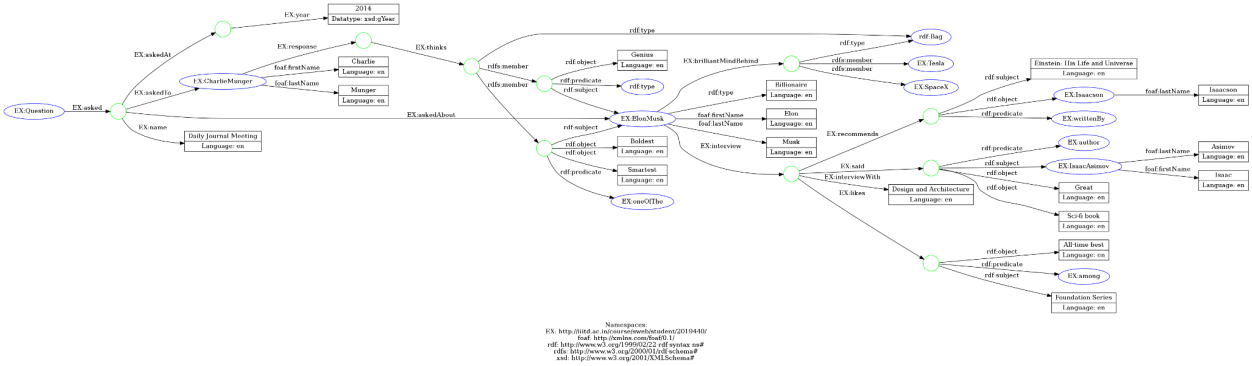


Link to the graph



Link to the graph

Combined Graph:



Link to the graph

Code for the graphs:

```
from rdflib import RDF, Graph, Literal, Namespace,
BNode, Bag, RDFS, URIRef, FOAF, XSD

# initialize the graph1 and bind the default IRI with
it
g1 = Graph('Memory', identifier=URIRef(

'http://iiitd.ac.in/course/sweb/student/2019440/Charl
ie'))
EX =
Namespace("http://iiitd.ac.in/course/sweb/student/201
9440/")
g1.bind("EX", EX)

# Creating the resources and blank nodes reqiured for
the first graph
Question = URIRef(EX.term("Question"))
ElonMusk = URIRef(EX.term("ElonMusk"))
CharlieMunger = URIRef(EX.term("CharlieMunger"))
```

```
Tesla = URIRef(EX.term("Tesla"))
SpaceX = URIRef(EX.term("SpaceX"))
CompanyBag = BNode()
CharlieQuestionBNode = BNode()
DailyJournalMeetingBNode = BNode()
CharlieResponse = BNode()
CharlieThought1 = BNode()
CharlieThought2 = BNode()
CharlieThoughtBag = BNode()

# Adding the triples to g1
g1.add((ElonMusk, RDF.type, Literal('Billionaire',
lang='en')))
g1.add((ElonMusk, EX.brilliantMindBehind,
CompanyBag))
g1.add((CompanyBag, RDF.type, RDF.Bag))
g1.add((CompanyBag, RDFS.member, SpaceX))
g1.add((CompanyBag, RDFS.member, Tesla))
g1.add((Question, EX.asked, CharlieQuestionBNode))
g1.add((CharlieQuestionBNode, EX.askedTo,
CharlieMunger))
g1.add((CharlieQuestionBNode, EX.askedAbout,
ElonMusk))
g1.add((CharlieQuestionBNode, EX.askedAt,
DailyJournalMeetingBNode))
g1.add((CharlieQuestionBNode, EX.name, Literal(
   "Daily Journal Meeting", lang='en')))
```

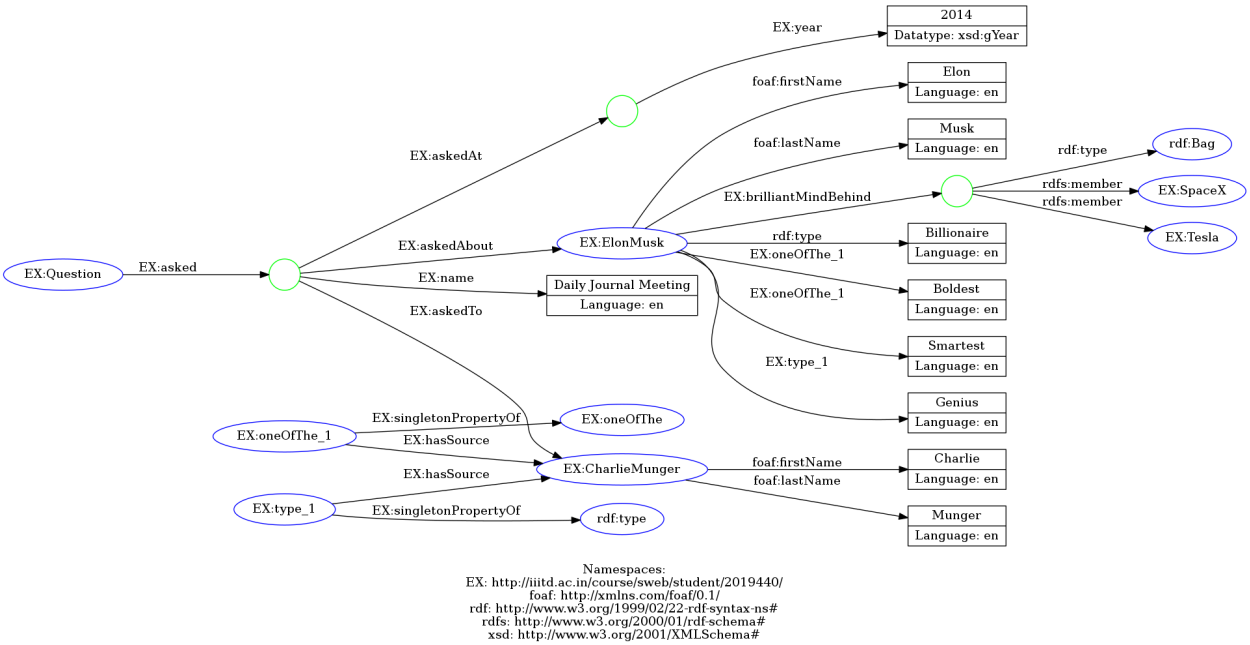
```
g1.add((DailyJournalMeetingBNode, EX.year,
Literal(2014, datatype=XSD.gYear)))
g1.add((CharlieMunger, FOAF.firstName,
Literal('Charlie', lang='en')))
g1.add((CharlieMunger, FOAF.lastName,
Literal('Munger', lang='en')))
g1.add((ElonMusk, FOAF.firstName, Literal('Elon',
lang='en')))
g1.add((ElonMusk, FOAF.lastName, Literal('Musk',
lang='en')))
g1.add((CharlieMunger, EX.response, CharlieResponse))
g1.add((CharlieResponse, EX.thinks,
CharlieThoughtBag))
g1.add((CharlieThoughtBag, RDF.type, RDF.Bag))
g1.add((CharlieThoughtBag, RDFS.member,
CharlieThought1))
g1.add((CharlieThoughtBag, RDFS.member,
CharlieThought2))
g1.add((CharlieThought1, RDF.subject, ElonMusk))
g1.add((CharlieThought1, RDF.predicate, RDF.type))
g1.add((CharlieThought1, RDF.object,
Literal('Genius', lang='en')))
g1.add((CharlieThought2, RDF.subject, ElonMusk))
g1.add((CharlieThought2, RDF.predicate, EX.oneOfThe))
g1.add((CharlieThought2, RDF.object,
Literal('Boldest', lang='en')))
g1.add((CharlieThought2, RDF.object,
Literal('Smartest', lang='en')))
```

```
# Seriealizing te graph and storing it in trig format
print(g1.serialize(format='trig'),
file=open('Q2_Graph1_part1.trig', 'w'))
print(g1.serialize(format='trig'),
file=open('Q2_Graph1_combined.trig', 'w'))


# initialize the graph2 and bind the default IRI with
it
g2 = Graph('Memory', identifier=URIRef(

'http://iiitd.ac.in/course/sweb/student/2019440/Musk'
))
g2.bind("EX", EX)

# Creating the resources and blank nodes reqiured for
the second graph
IsaacAsimov = URIRef(EX.term("IsaacAsimov"))
Isaacson = URIRef(EX.term("Isaacson"))
MuskInterviewBNode = BNode()
MuskStatement1 = BNode()
MuskStatement2 = BNode()
MuskStatement3 = BNode()

# Adding the triples to g2
g2.add((ElonMusk, EX.interview, MuskInterviewBNode))
g2.add((MuskInterviewBNode, EX.interviewWith,
```

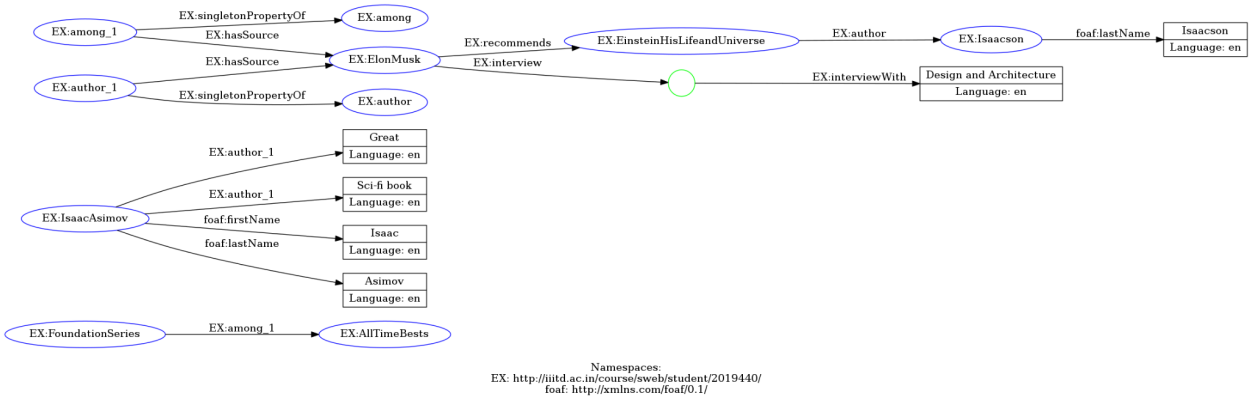
```
       Literal('Design and Architecture',
lang='en')))
g2.add((MuskInterviewBNode, EX.said, MuskStatement1))
g2.add((MuskStatement1, RDF.subject, IsaacAsimov))
g2.add((IsaacAsimov, FOAF.firstName, Literal('Isaac',
lang='en')))
g2.add((IsaacAsimov, FOAF.lastName, Literal('Asimov',
lang='en')))
g2.add((MuskStatement1, RDF.predicate, EX.author))
g2.add((MuskStatement1, RDF.object, Literal('Great',
lang='en')))
g2.add((MuskStatement1, RDF.object, Literal('Sci-fi
book', lang='en')))
g2.add((MuskInterviewBNode, EX.likes,
MuskStatement2))
g2.add((MuskStatement2, RDF.subject,
Literal('Foundation Series', lang='en')))
g2.add((MuskStatement2, RDF.predicate, EX.among))
g2.add((MuskStatement2, RDF.object, Literal('All-time
best', lang='en')))
g2.add((MuskInterviewBNode, EX.recommends,
MuskStatement3))
g2.add((MuskStatement3, RDF.subject, Literal(
   'Einstein: His Life and Universe', lang='en')))
g2.add((MuskStatement3, RDF.predicate, EX.writtenBy))
g2.add((MuskStatement3, RDF.object, Isaacson))
g2.add((Isaacson, FOAF.lastName, Literal('Isaacson',
lang='en')))
```

```
# Seriealizing te graph and storing it in trig format
print(g2.serialize(format='trig'),
file=open('Q2_Graph1_part2.trig', 'w'))
print(g2.serialize(format='trig'),
file=open('Q2_Graph1_combined.trig', 'a'))
```

2. **Named Graphs + n-ary properties + singleton properties**
   Individual Named Graphs:

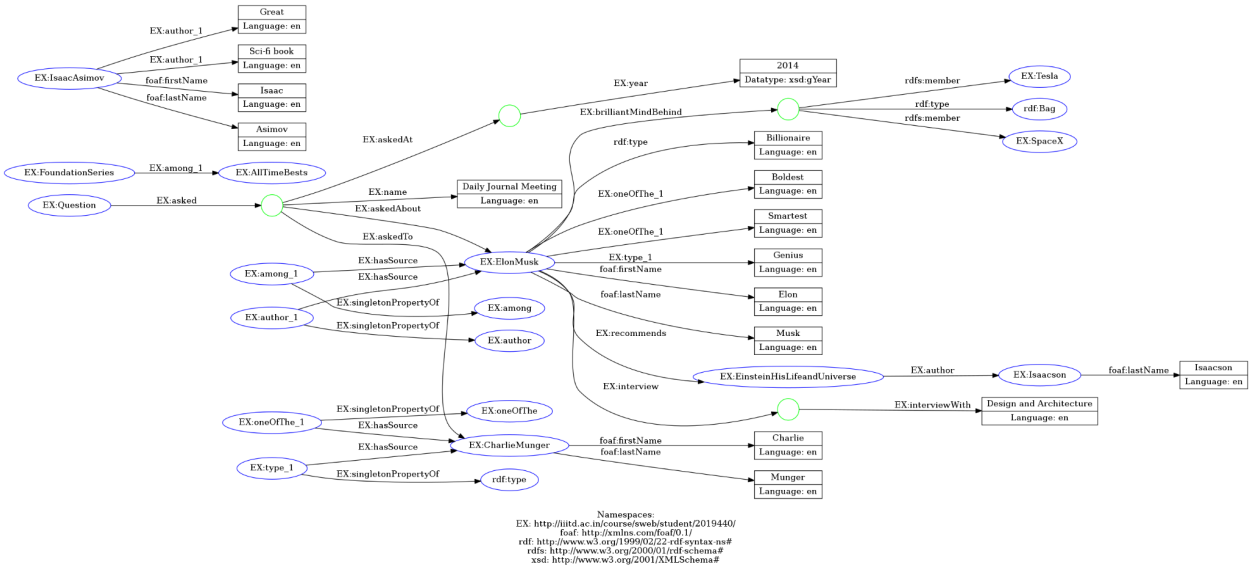


Link to the graph



[Link](#) to the graph

Combined Graph:



[Link](#) to the graph

Code for the graph:

```
from rdflib import RDF, Graph, Literal, Namespace,
BNode, Bag, RDFS, URIRef, FOAF, XSD


# initialize the graph1 and bind the default IRI with
it
g1 = Graph('Memory', identifier=URIRef(
```

```
'http://iiitd.ac.in/course/sweb/student/2019440/Charl
ie'))
EX =
Namespace("http://iiitd.ac.in/course/sweb/student/201
9440/")
g1.bind("EX", EX)

# Creating the resources and blank nodes reqiured for
the first graph
Question = URIRef(EX.term("Question"))
ElonMusk = URIRef(EX.term("ElonMusk"))
CharlieMunger = URIRef(EX.term("CharlieMunger"))
Tesla = URIRef(EX.term("Tesla"))
SpaceX = URIRef(EX.term("SpaceX"))
CompanyBag = BNode()
CharlieQuestionBNode = BNode()
DailyJournalMeetingBNode = BNode()
CharlieResponse = BNode()
CharlieThought1 = BNode()
CharlieThought2 = BNode()
CharlieThoughtBag = BNode()

# Adding the triples to g1
g1.add((ElonMusk, RDF.type, Literal('Billionaire',
lang='en')))
g1.add((ElonMusk, EX.brilliantMindBehind,
CompanyBag))
```

```
g1.add((CompanyBag, RDF.type, RDF.Bag))
g1.add((CompanyBag, RDFS.member, SpaceX))
g1.add((CompanyBag, RDFS.member, Tesla))
g1.add((Question, EX.asked, CharlieQuestionBNode))
g1.add((CharlieQuestionBNode, EX.askedTo,
CharlieMunger))
g1.add((CharlieQuestionBNode, EX.askedAbout,
ElonMusk))
g1.add((CharlieQuestionBNode, EX.askedAt,
DailyJournalMeetingBNode))
g1.add((CharlieQuestionBNode, EX.name, Literal(
   "Daily Journal Meeting", lang='en')))
g1.add((DailyJournalMeetingBNode, EX.year,
Literal(2014, datatype=XSD.gYear)))
g1.add((CharlieMunger, FOAF.firstName,
Literal('Charlie', lang='en')))
g1.add((CharlieMunger, FOAF.lastName,
Literal('Munger', lang='en')))
g1.add((ElonMusk, FOAF.firstName, Literal('Elon',
lang='en')))
g1.add((ElonMusk, FOAF.lastName, Literal('Musk',
lang='en')))
g1.add((ElonMusk, EX.type_1, Literal('Genius',
lang='en')))
g1.add((EX.type_1, EX.singletonPropertyOf, RDF.type))
g1.add((EX.type_1, EX.hasSource, CharlieMunger))
g1.add((ElonMusk, EX.oneOfThe_1, Literal('Boldest',
lang='en')))
```

```
g1.add((ElonMusk, EX.oneOfThe_1, Literal('Smartest',
lang='en')))
g1.add((EX.oneOfThe_1, EX.singletonPropertyOf,
EX.oneOfThe))
g1.add((EX.oneOfThe_1, EX.hasSource, CharlieMunger))

# Seriealizing te graph and storing it in trig format
print(g1.serialize(format='trig'),
file=open('Q2_Graph2_part1.trig', 'w'))
print(g1.serialize(format='trig'),
file=open('Q2_Graph2_combined.trig', 'w'))

# initialize the graph2 and bind the default IRI with
it
g2 = Graph('Memory', identifier=URIRef(

'http://iiitd.ac.in/course/sweb/student/2019440/Musk'
))
g2.bind("EX", EX)

# Creating the resources and blank nodes reqiured for
the second graph
IsaacAsimov = URIRef(EX.term("IsaacAsimov"))
Isaacson = URIRef(EX.term("Isaacson"))
EinsteinHisLifeandUniverse =
URIRef(EX.term("EinsteinHisLifeandUniverse"))
FoundationSeries =
URIRef(EX.term("FoundationSeries"))
```

```
AllTimeBests = URIRef(EX.term("AllTimeBests"))
MuskInterviewBNode = BNode()
MuskStatement1 = BNode()
MuskStatement2 = BNode()
MuskStatement3 = BNode()


# Adding the triples to g2
g2.add((ElonMusk, EX.interview, MuskInterviewBNode))
g2.add((MuskInterviewBNode, EX.interviewWith,
       Literal('Design and Architecture',
lang='en')))
g2.add((IsaacAsimov, FOAF.firstName, Literal('Isaac',
lang='en')))
g2.add((IsaacAsimov, FOAF.lastName, Literal('Asimov',
lang='en')))
g2.add((IsaacAsimov, EX.author_1, Literal('Great',
lang='en')))
g2.add((IsaacAsimov, EX.author_1, Literal('Sci-fi
book', lang='en')))
g2.add((EX.author_1, EX.singletonPropertyOf,
EX.author))
g2.add((EX.author_1, EX.hasSource, ElonMusk))
g2.add((FoundationSeries, EX.among_1, AllTimeBests))
g2.add((EX.among_1, EX.singletonPropertyOf,
EX.among))
g2.add((EX.among_1, EX.hasSource, ElonMusk))
```

```
g2.add((ElonMusk, EX.recommends,
EinsteinHisLifeandUniverse))
g2.add((EinsteinHisLifeandUniverse, EX.author,
Isaacson))
g2.add((Isaacson, FOAF.lastName, Literal('Isaacson',
lang='en')))


# Seriealizing te graph and storing it in trig format
print(g2.serialize(format='trig'),
file=open('Q2_Graph2_part2.trig', 'w'))
print(g2.serialize(format='trig'),
file=open('Q2_Graph2_combined.trig', 'a'))
```

- **Reification** means to express the abstract graph with the existing methods which are already available with the language.
- **Standard reification** requires stating 4 additional triples to refer to the one for which we want the metadata. Thus it is inefficient due to exchanging or persisting the RDF data and the cumbersome syntax to access and match the corresponding four reification triples.
- **N-Ary relations** model the metadata via new relationships. It is easy to understand but it too increases the complexity and is proven difficult to evolve models in a backward-compatible way.
- **Singleton Properties** introduces statement identifiers as a part of the predicate. While it created a more compact representation, but still it is highly inefficient as in the worst case, one would have to query all the corresponding singleton properties of a given property to achieve the data.
- **Named Graphs** designate each statement to be a part of the specified sub-graph. The identifier of the named graph can be treated as a node in the RDF graph so that one can easily make statements about the entire named graph. While it eliminates the need for regex-based parsings, the updates for triple source become more complicated and cumbersome to maintain.

Question 3 | RDF-star

Code:

```
/**
 *
 */
package org.example;

import static org.eclipse.rdf4j.model.util.Values.literal;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

import org.eclipse.rdf4j.model.BNode;
import org.eclipse.rdf4j.model.IRI;
import org.eclipse.rdf4j.model.Model;
import org.eclipse.rdf4j.model.Namespace;
import org.eclipse.rdf4j.model.Triple;
import org.eclipse.rdf4j.model.util.ModelBuilder;
import org.eclipse.rdf4j.model.util.Models;
import org.eclipse.rdf4j.model.util.Statements;
import org.eclipse.rdf4j.model.util.Values;
import org.eclipse.rdf4j.model.vocabulary.FOAF;
import org.eclipse.rdf4j.model.vocabulary.RDF;
import org.eclipse.rdf4j.model.vocabulary.RDFS;
import org.eclipse.rdf4j.model.vocabulary.XSD;
import org.eclipse.rdf4j.rio.RDFFormat;
import org.eclipse.rdf4j.rio.Rio;

/**
 * @author Pritish Wadhwa
 *
 */
public class HelloRDF4J {

	/**
	 * @param args
	 */
	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub

		/***** Part A *****/
```

```
		// Defining the custom namespace
		Namespace EX = Values.namespace("EX",
"http://iiitd.ac.in/course/sweb/student/2019440/");

		// Defining the relevant Resources and Blank nodes
		IRI Question = Values.iri(EX, "Question");
		IRI ElonMusk = Values.iri(EX, "ElonMusk");
		IRI CharlieMunger = Values.iri(EX, "CharlieMunger");
		IRI Tesla = Values.iri(EX, "Tesla");
		IRI SpaceX = Values.iri(EX, "SpaceX");
		IRI IsaacAsimov = Values.iri(EX, "IsaacAsimov");
		IRI Isaacson = Values.iri(EX, "Isaacson");
		BNode CompanyBag = Values.bnode();
		BNode CharlieQuestionBNode = Values.bnode();
		BNode DailyJournalMeetingBNode = Values.bnode();
		BNode MuskInterviewBNode = Values.bnode();

		// Instantiating the model builder class to generate the graph
		ModelBuilder builder = new ModelBuilder();

		// Adding the triples in the graph
		builder.setNamespace("EX",
"http://iiitd.ac.in/course/sweb/student/2019440/").subject(ElonMusk)
					.add("EX:interview", MuskInterviewBNode).add(RDF.TYPE,
literal("Billionaire", "en"))
					.add("EX:brilliantMindBehind",
CompanyBag).add(FOAF.FIRST_NAME, literal("Elon", "en"))
					.add(FOAF.LAST_NAME, literal("Musk",
"en")).subject(CompanyBag).add(RDF.TYPE, RDF.BAG)
					.add(RDFS.MEMBER, SpaceX).add(RDFS.MEMBER,
Tesla).subject(Question)
					.add("EX:asked",
CharlieQuestionBNode).subject(CharlieQuestionBNode).add("EX:askedTo", CharlieMunger)
					.add("EX:askedAbout", ElonMusk).add("EX:askedAt",
DailyJournalMeetingBNode)
					.add("EX:name", literal("Daily Journal Meeting",
"en")).subject(DailyJournalMeetingBNode)
					.add("EX:year", literal("2014",
XSD.GYEAR)).subject(CharlieMunger)
					.add(FOAF.FIRST_NAME, literal("Charlie",
"en")).add(FOAF.LAST_NAME, literal("Munger", "en"))
					.subject(MuskInterviewBNode).add("EX:interviewWith",
literal("Design and Architecture", "en"))
```

```
                                .subject(IsaacAsimov).add(FOAF.FIRST_NAME, literal("Isaac",
"en"))
                                .add(FOAF.LAST_NAME, literal("Asimov",
"en")).subject(Isaacson)
                                .add(FOAF.LAST_NAME, literal("Isaacson", "en"));

                // Building the graph
                Model model = builder.build();
                Triple reifiedTriple = Values.triple(ElonMusk, RDF.TYPE, literal("Genius", "en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "thinks"),
CharlieMunger, null));
                reifiedTriple = Values.triple(ElonMusk, Values.iri(EX, "oneOfThe"),
literal("Boldest", "en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "thinks"),
CharlieMunger, null));
                reifiedTriple = Values.triple(ElonMusk, Values.iri(EX, "oneOfThe"),
literal("Smartest", "en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "thinks"),
CharlieMunger, null));
                reifiedTriple = Values.triple(IsaacAsimov, Values.iri(EX, "author"), literal("Great",
"en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "said"), ElonMusk,
null));
                reifiedTriple = Values.triple(IsaacAsimov, Values.iri(EX, "author"), literal("Sci-fi
book", "en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "said"), ElonMusk,
null));
                reifiedTriple = Values.triple(Values.iri(EX, "FoundationSeries"), Values.iri(EX,
"among"),
                                literal("All-time best", "en"));
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "likes"), ElonMusk,
null));
                reifiedTriple = Values.triple(Values.iri(EX, "EinsteinHisLifeandUniverse"),
Values.iri(EX, "EwrittenBy"),
                                Isaacson);
                model.add(Statements.statement(reifiedTriple, Values.iri(EX, "recommends"),
ElonMusk, null));

                // Binding the namespaces with the graph
                model.setNamespace(RDF.NS);
                model.setNamespace(RDFS.NS);
                model.setNamespace(FOAF.NS);
                model.setNamespace(EX);
```

```
		// Converting the created reifications to rdf star
		Model convertedModel = Models.convertReificationToRDFStar(model);

		// Printing the model on console
		Rio.write(convertedModel, System.out, RDFFormat.TURTLESTAR);

		/***** Part B *****/

		// Saving the rdf star triples in .ttls format
		FileOutputStream ttlsFormat = new
FileOutputStream("./target/classes/org/example/reifiedTriples.ttls");
		try {
			Rio.write(convertedModel, ttlsFormat, RDFFormat.TURTLESTAR);
		} finally {
			ttlsFormat.close();
		}

		// Reading the rdf star file
		String fileName = "reifiedTriples.ttls";
		InputStream input = HelloRDF4J.class.getResourceAsStream("./" + fileName);
		Model loadedModel = Rio.parse(input, "", RDFFormat.TURTLESTAR);

		// Converting the rdf star triples to Reified Triples
		convertedModel = Models.convertRDFStarToReification(loadedModel);

		// Printing the reified triples on console
		System.out.println("\n\n\n\nRDF Star Model converted to Reification");
		Rio.write(convertedModel, System.out, RDFFormat.TURTLE);

		// Converting the reified triples back to RDF Star
		convertedModel = Models.convertReificationToRDFStar(convertedModel);

		// Printing the rdf star triples on console
		System.out.println("\n\n\n\nRDF Star Model converted to Reification");
		Rio.write(convertedModel, System.out, RDFFormat.TURTLESTAR);

	}

}
```

Output:

<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> a "Billionaire"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/interview> _:node1ft8aarvix4;
  <http://iiitd.ac.in/course/sweb/student/2019440/brilliantMindBehind> _:node1ft8aarvix1;
  <http://xmlns.com/foaf/0.1/firstName> "Elon"@en;
  <http://xmlns.com/foaf/0.1/lastName> "Musk"@en .

_:node1ft8aarvix1 a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag>;
  <http://www.w3.org/2000/01/rdf-schema#member>
<http://iiitd.ac.in/course/sweb/student/2019440/SpaceX>,
    <http://iiitd.ac.in/course/sweb/student/2019440/Tesla> .

<http://iiitd.ac.in/course/sweb/student/2019440/Question>
<http://iiitd.ac.in/course/sweb/student/2019440/asked>
    _:node1ft8aarvix2 .

_:node1ft8aarvix2 <http://iiitd.ac.in/course/sweb/student/2019440/askedTo>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>;
  <http://iiitd.ac.in/course/sweb/student/2019440/askedAbout>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
  <http://iiitd.ac.in/course/sweb/student/2019440/askedAt> _:node1ft8aarvix3;
  <http://iiitd.ac.in/course/sweb/student/2019440/name> "Daily Journal Meeting"@en .

_:node1ft8aarvix3 <http://iiitd.ac.in/course/sweb/student/2019440/year>
"2014"^^<http://www.w3.org/2001/XMLSchema#gYear> .

<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>
<http://xmlns.com/foaf/0.1/firstName>
    "Charlie"@en;
  <http://xmlns.com/foaf/0.1/lastName> "Munger"@en .

_:node1ft8aarvix4 <http://iiitd.ac.in/course/sweb/student/2019440/interviewWith> "Design and
Architecture"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://xmlns.com/foaf/0.1/firstName>
    "Isaac"@en;
  <http://xmlns.com/foaf/0.1/lastName> "Asimov"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson> <http://xmlns.com/foaf/0.1/lastName>
    "Isaacson"@en .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> "Genius"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe> "Boldest"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe> "Smartest"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://iiitd.ac.in/course/sweb/student/2019440/author> "Great"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://iiitd.ac.in/course/sweb/student/2019440/author> "Sci-fi book"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/FoundationSeries>
<http://iiitd.ac.in/course/sweb/student/2019440/among> "All-time best"@en>>
  <http://iiitd.ac.in/course/sweb/student/2019440/likes>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/EinsteinHisLifeandUniverse>
<http://iiitd.ac.in/course/sweb/student/2019440/EwrittenBy>
<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson>>>
  <http://iiitd.ac.in/course/sweb/student/2019440/recommends>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

RDF Star Model converted to Reification

<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> a "Billionaire"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/interview>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix4;

<http://iiitd.ac.in/course/sweb/student/2019440/brilliantMindBehind>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix1;
<http://xmlns.com/foaf/0.1/firstName> "Elon"@en;
<http://xmlns.com/foaf/0.1/lastName> "Musk"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix1 a
<http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag>;
<http://www.w3.org/2000/01/rdf-schema#member>
<http://iiitd.ac.in/course/sweb/student/2019440/SpaceX>,
<http://iiitd.ac.in/course/sweb/student/2019440/Tesla> .

<http://iiitd.ac.in/course/sweb/student/2019440/Question>
<http://iiitd.ac.in/course/sweb/student/2019440/asked>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix2 .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix2
<http://iiitd.ac.in/course/sweb/student/2019440/askedTo>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>;
<http://iiitd.ac.in/course/sweb/student/2019440/askedAbout>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
<http://iiitd.ac.in/course/sweb/student/2019440/askedAt>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix3;
<http://iiitd.ac.in/course/sweb/student/2019440/name> "Daily Journal Meeting"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix3
<http://iiitd.ac.in/course/sweb/student/2019440/year>
"2014"^^<http://www.w3.org/2001/XMLSchema#gYear> .

<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>
<http://xmlns.com/foaf/0.1/firstName>
"Charlie"@en;
<http://xmlns.com/foaf/0.1/lastName> "Munger"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix4
<http://iiitd.ac.in/course/sweb/student/2019440/interviewWith>
"Design and Architecture"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://xmlns.com/foaf/0.1/firstName>
"Isaac"@en;
<http://xmlns.com/foaf/0.1/lastName> "Asimov"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson> <http://xmlns.com/foaf/0.1/lastName>
"Isaacson"@en .

_:PDxodHRwOi8vaWlpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0Vsb25NdXNrIGh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyN0eXBlICJHZW5pdXMiQGVuPj43d
  a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "Genius"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

_:PDxodHRwOi8vaWlpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0Vsb25NdXNrIGh0dHA6Ly9paWl0ZC5hYy5pbi9jb3Vyc2Uvc3dlYi9zdHVkZW50LzIwMTk0NDAvb25lT2ZUaGUgIkJvbGRlc3QiQGVuPj43d
  a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "Boldest"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

_:PDxodHRwOi8vaWlpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0Vsb25NdXNrIGh0dHA6Ly9paWl0ZC5hYy5pbi9jb3Vyc2Uvc3dlYi9zdHVkZW50LzIwMTk0NDAvb25lT2ZUaGUgIlNtYXJ0ZXN0IkBlbj4-
  a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "Smartest"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

_:PDxodHRwOi8vaWlpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0lzYWFjQXNpbW92IGh0dHA6Ly9paWl0ZC5hYy5pbi9jb3Vyc2Uvc3dlYi9zdHVkZW50LzIwMTk0NDAvYXV0aG9yICJHcmVhdCJAZW4-Pg3d3d
  a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>;

<http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/author>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "Great"@en;
<http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

_:PDxodHRwOi8vaWIpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0lz
YWFjQXNpbW92IGh0dHA6Ly9paWl0ZC5hYy5pbi9jb3Vyc2Uvc3dlYi9zdHVkZW50LzIwMTk0ND
AvYXV0aG9yICJTY2ktZmkgYm9vayJAZW4-Pg3d3d
a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/author>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "Sci-fi book"@en;
<http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

_:PDxodHRwOi8vaWIpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0Z
vdW5kYXRpb25TZXJpZXMgaHR0cDovL2lpaXRkLmFjLmluL2NvdXJzZS9zd2ViL3N0dWRlbnQ
vMjAxOTQ0MC9hbW9uZyAiQWxsLXRpbWUgYmVzdCJAZW4-Pg3d3d
a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/FoundationSeries>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/among>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#object> "All-time best"@en;
<http://iiitd.ac.in/course/sweb/student/2019440/likes>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

_:PDxodHRwOi8vaWIpdGQuYWMuaW4vY291cnNlL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0V
pbnN0ZWluSGlzTGlmZWFuZFVuaXZlcnNlIGh0dHA6Ly9paWl0ZC5hYy5pbi9jb3Vyc2Uvc3dlYi9
zdHVkZW50LzIwMTk0NDAvRXdyaXR0ZW5CeSBodHRwOi8vaWIpdGQuYWMuaW4vY291cnNl
lL3N3ZWIvc3R1ZGVudC8yMDE5NDQwL0lzYWFjc29uPj43d
a <http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#subject>
<http://iiitd.ac.in/course/sweb/student/2019440/EinsteinHisLifeandUniverse>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
<http://iiitd.ac.in/course/sweb/student/2019440/EwrittenBy>;
<http://www.w3.org/1999/02/22-rdf-syntax-ns#object>
<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson>;
<http://iiitd.ac.in/course/sweb/student/2019440/recommends>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

RDF Star Model converted to Reification

<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> a "Billionaire"@en;
  <http://iiitd.ac.in/course/sweb/student/2019440/interview>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix4;
  <http://iiitd.ac.in/course/sweb/student/2019440/brilliantMindBehind>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix1;
  <http://xmlns.com/foaf/0.1/firstName> "Elon"@en;
  <http://xmlns.com/foaf/0.1/lastName> "Musk"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix1 a
<http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag>;
  <http://www.w3.org/2000/01/rdf-schema#member>
<http://iiitd.ac.in/course/sweb/student/2019440/SpaceX>,
    <http://iiitd.ac.in/course/sweb/student/2019440/Tesla> .

<http://iiitd.ac.in/course/sweb/student/2019440/Question>
<http://iiitd.ac.in/course/sweb/student/2019440/asked>
    _:genid-854c21750030469789904419a675f2ae-node1ft8aarvix2 .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix2
<http://iiitd.ac.in/course/sweb/student/2019440/askedTo>
    <http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>;
  <http://iiitd.ac.in/course/sweb/student/2019440/askedAbout>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>;
  <http://iiitd.ac.in/course/sweb/student/2019440/askedAt>
_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix3;
  <http://iiitd.ac.in/course/sweb/student/2019440/name> "Daily Journal Meeting"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix3
<http://iiitd.ac.in/course/sweb/student/2019440/year>
    "2014"^^<http://www.w3.org/2001/XMLSchema#gYear> .

<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger>
<http://xmlns.com/foaf/0.1/firstName>
    "Charlie"@en;
  <http://xmlns.com/foaf/0.1/lastName> "Munger"@en .

_:genid-854c21750030469789904419a675f2ae-node1ft8aarvix4
<http://iiitd.ac.in/course/sweb/student/2019440/interviewWith>
    "Design and Architecture"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://xmlns.com/foaf/0.1/firstName>
"Isaac"@en;
<http://xmlns.com/foaf/0.1/lastName> "Asimov"@en .

<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson> <http://xmlns.com/foaf/0.1/lastName>
"Isaacson"@en .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> "Genius"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe> "Boldest"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk>
<http://iiitd.ac.in/course/sweb/student/2019440/oneOfThe> "Smartest"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/thinks>
<http://iiitd.ac.in/course/sweb/student/2019440/CharlieMunger> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://iiitd.ac.in/course/sweb/student/2019440/author> "Great"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/IsaacAsimov>
<http://iiitd.ac.in/course/sweb/student/2019440/author> "Sci-fi book"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/said>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/FoundationSeries>
<http://iiitd.ac.in/course/sweb/student/2019440/among> "All-time best"@en>>
<http://iiitd.ac.in/course/sweb/student/2019440/likes>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

<<<http://iiitd.ac.in/course/sweb/student/2019440/EinsteinHisLifeandUniverse>
<http://iiitd.ac.in/course/sweb/student/2019440/EwrittenBy>
<http://iiitd.ac.in/course/sweb/student/2019440/Isaacson>>>
<http://iiitd.ac.in/course/sweb/student/2019440/recommends>
<http://iiitd.ac.in/course/sweb/student/2019440/ElonMusk> .

Advantages of RDF Star over other reification techniques:

- It proposes a more efficient reification serialization syntax.
- It reduces document size.
- This improves the efficiency of data exchange.
- Shorter SPARQL Queries for improved comprehensibility.

RDF Libraries Used:
Ques1: RDFlib (Python)
Ques2: RDFlib (Python)
Ques3: RDF4J (Python)