Slide 11

1. **Find the subclasses of MotorVehicle**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {  ?x      rdfs:subClassOf                ns:MotorVehicle }
```

Result

```
<http://example.org/schemas/vehicles#Truck>
<http://example.org/schemas/vehicles#Van>
<http://example.org/schemas/vehicles#PassengerVehicle>
<http://example.org/schemas/vehicles#MotorVehicle>
<http://example.org/schemas/vehicles#MiniVan>
```

2. **Find the subclasses of MiniVan**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE { ?x       rdfs:subClassOf            ex:MiniVan }
```

Result (rdfs:subClassOf is reflexive)

```
<http://example.org/schemas/vehicles#MiniVan>
```

3. **Find the superclasses of MiniVan**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?y
WHERE { ex:MiniVan      rdfs:subClassOf            ?y }
```

Result (the answer contains the predefined class rdf:Resource)

```
<http://example.org/schemas/vehicles#PassengerVehicle>
<http://example.org/schemas/vehicles#Van>
<http://example.org/schemas/vehicles#MiniVan>
<http://example.org/schemas/vehicles#MotorVehicle>
<http://www.w3.org/2000/01/rdf-schema#Resource>
```

Slide 20

1. **Return the full names of all people in the graph**

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?fullName
WHERE {?x    vCard:FN    ?fullName}
```

Result

```
"John Smith"
"Mary Smith"
```

2. **Return the relation between John and Mary**

```
PREFIX ex:<http://example.org/data#> .
SELECT ?p
WHERE { ex:john          ?p        ex :mary }
```

Result

```
<http://example.org/data#marriedTo>
```

3. **Return the spouse of a person by the name of John Smith**

```
PREFIX vCard: http://www.w3.org/2001/vcard-rdf/3.0#.
PREFIX ex:<http://example.org/data#> .
SELECT ?y
WHERE {?x       vCard:FN          "John Smith".
          ?x       ex:marriedTo     ?y}
```

Result:

```
<http://example.org/data#mary>
```

4. **Return the name and the first name of all people in the KB**

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?familyName, ?firstName
WHERE {?x       vCard:N           ?name .
          ?name vCard:Given      ?firstName .
          ?name vCard:Family     ?familyName .}
```

Slide 44

1. **Find the superclasses of MiniVan that are not rdf:Resource.**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?y
WHERE { ex:MiniVan     rdfs:subClassOf          ?y .
          FILTER(?y != rdf:Resource)
}
```

Result

```
<http://example.org/schemas/vehicles#PassengerVehicle>
<http://example.org/schemas/vehicles#Van>
<http://example.org/schemas/vehicles#MiniVan>
<http://example.org/schemas/vehicles#MotorVehicle>
```

2. **Find all the motor vehicles**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntaxns#>
SELECT ?x
WHERE { ?x      rdf:type            ex:MotorVehicle }
```

Result

```
<http://example.org/instances#mv1>
<http://example.org/instances#pv1>
<http://example.org/instances#v1>
<http://example.org/instances#tr1>
```

3. **Find all the vans and passenger vehicles**

```
PREFIX ex: <http://example.org/schemas/vehicles#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?x
WHERE { { ?x    rdf:type            ex:Van }
          UNION
            { ?x    rdf:type          ex:PassengerVehicle }
}
```

Result

```
<http://example.org/instances#v1>
<http://example.org/instances#mv1>
<http://example.org/instances#pv1>
<http://example.org/instances#mv1>
```

4. **Return all people over 30 in the KB**

```
PREFIX ex: <http://example.org/data#>
SELECT ?x
WHERE {?x       ont:has Age       ?age .
           FILTER(?age > 30)}
```

Slide 51

Find the names of people with name but **no** expressed age **or** age less than 60 years.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/schema/>
SELECT ?name
WHERE {
          ?x   foaf:givenName   ?name .
          OPTIONAL {  ?x   ex:age   ?age .
                            FILTER(?age >= 60) } .
          FILTER (!bound(?age))
}
```

Slide 53

1. Display the name of all involved persons.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name
WHERE {
        ?p rdf:type foaf:Person.
        ?p foaf:name ?name}
```

2. Display all female persons and persons that don't have a gender.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person
WHERE {
        ?person rdf:type foaf:Person.
        OPTIONAL {?person foaf:gender ?g . FILTER (?g = "male")}
        FILTER (!(bound(?g)))
}
```

3. Display all pairs of people that know each other.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person1 ?person2
WHERE {
        ?person1 rdf:type foaf:Person.
        ?person2 rdf:type foaf:Person.
        ?person1 foaf:knows ?person2
}
```

4. Display all people that don't know Anakin.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person
WHERE {
        ?person rdf:type foaf:Person.
        OPTIONAL {?person foaf:knows ?a . FILTER (?a =<#Anakin>)}
        FILTER (!(bound(?a)))
}
```

5. Display all people that know somebody who knows Anakin.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person
WHERE {
        ?person rdf:type foaf:Person.
        ?person foaf:knows ?person2.
        ?person2 foaf:knows <#Anakin>}
```

6. Display all people that know somebody who knows Anakin and don't know Anakin directly.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person
WHERE {
        ?person rdf:type foaf:Person.
        ?person foaf:knows ?person2.
        ?person2 foaf:knows <#Anakin>.
        OPTIONAL {?person foaf:knows ?a . FILTER (?a =<#Anakin>)}
        FILTER (!(bound(?a)))
}
```

7. Assume that the name is build out of givenname-whitspace-familyname. Find the guy for whom this is not true.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX fn: <http://www.w3.org/2004/07/xpath-functions>
SELECT ?person
WHERE {
        ?person rdf:type foaf:Person.
        OPTIONAL {
                ?person foaf:name ?name.
                ?person foaf:familyname ?familyname.
                ?person foaf:givenname ?givennam
        }
        FILTER (!(?name = fn:concat(?givenname," ",?familyname)))
}
```

8. Put all different types of names (i.e. name, familyname and givenanme) from Persons into one variable.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name
WHERE {
        {?person foaf:name ?name} UNION
        {?person foaf:familyname ?name} UNION
        {?person foaf:givenname ?name}
}
```