7BDIN006W.1 BIG DATA THEORY AND PRACTICE

Week 10 Seminar Tasks

Note: These exercises require students' personal laptops; no new programs can be installed on lab PCs.

Task 1. Neo4j for Beginners Tutorial

Guide to Creating a New Graph Database in Neo4j

Step 1: Install Neo4j

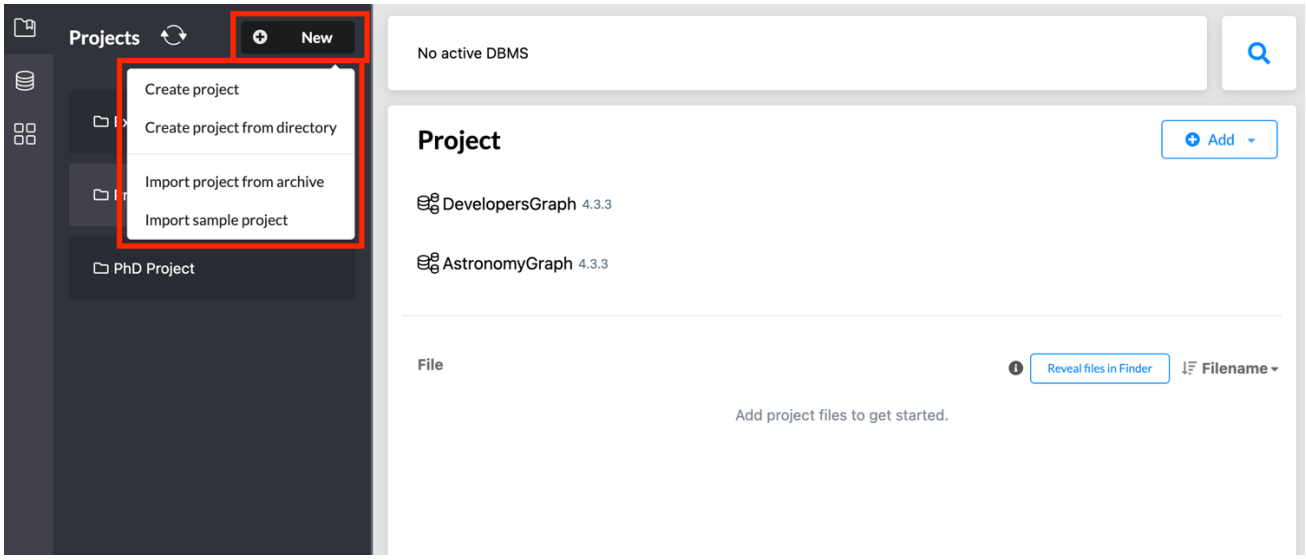
- **Action**: If you haven't already installed Neo4j, download and install it from the official Neo4j website. Choose the edition that best fits your needs (Community Edition is usually sufficient for educational purposes).
- **Note**: Follow the installation instructions provided on the website or in the installation package.

Step 2: Launch Neo4j

- **Action**: Open the Neo4j Desktop application. This is the interface where you will manage your Neo4j databases.
- **Note**: The first time you open Neo4j Desktop, you might need to set up an account or log in.

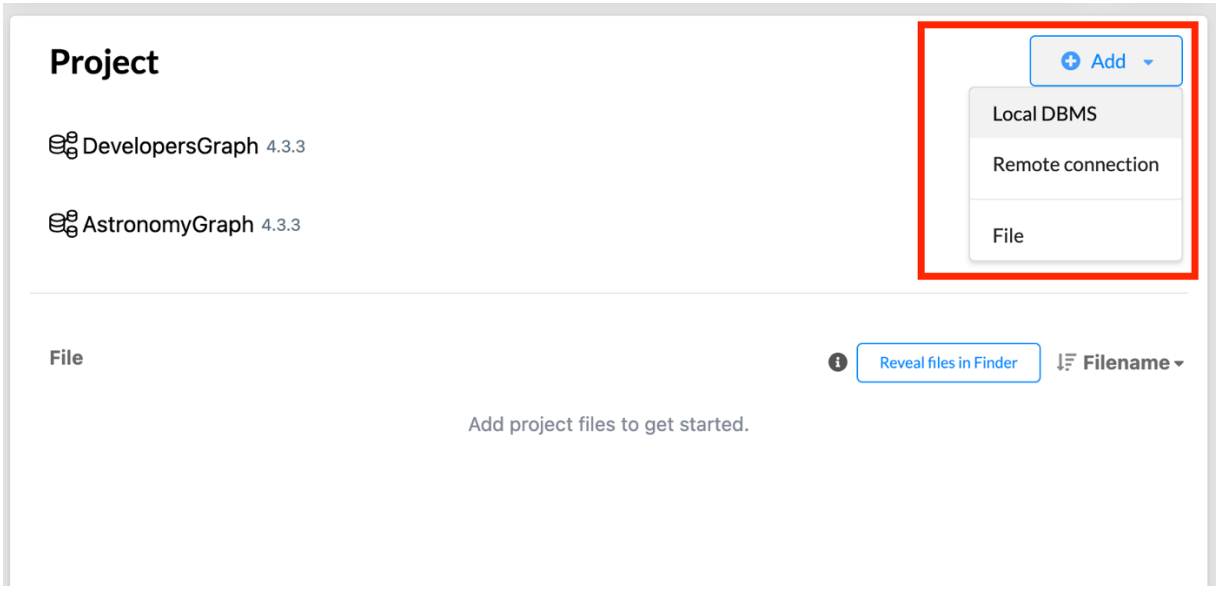
Step 3: Create a New Project

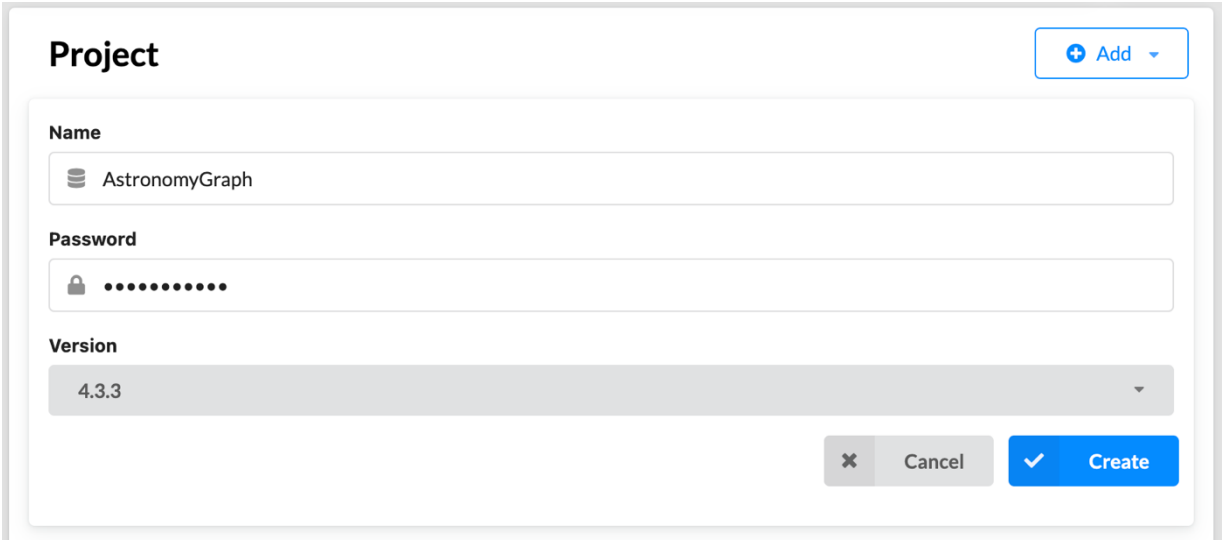
- **Action**: In the Neo4j Desktop interface, create a new project. This will be a container for your databases and their associated resources.
- **Steps**: Click on the "New" button or "Add Project" and give your project a descriptive name.



Step 4: Add a New Database

- **Action**: Within your new project, add a new database.
- **Steps**:
    a. Click on “Add” or “New Database” within your project.
    b. Choose “Local DBMS” to create a database on your machine.
    c. Assign a name to your database: AstronomyGraph.
    d. Select “Create”.



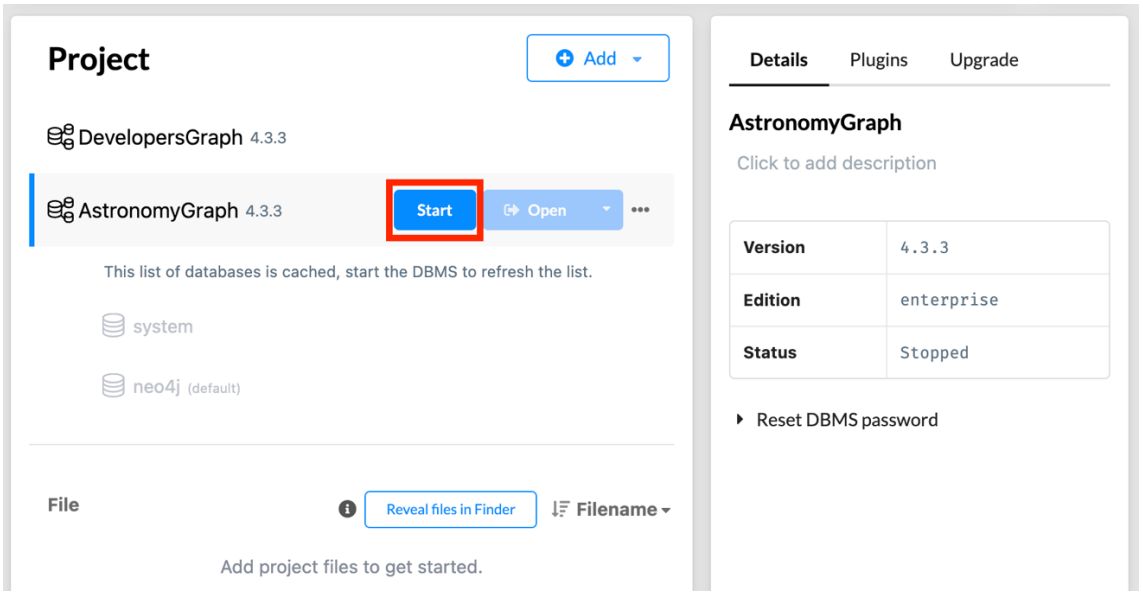


Step 5: Set Database Credentials

- **Action**: When creating the database, you will be prompted to set a password. This password will be needed to connect to the database.
- **Note**: Remember the password or store it securely, as it is essential for accessing your database.

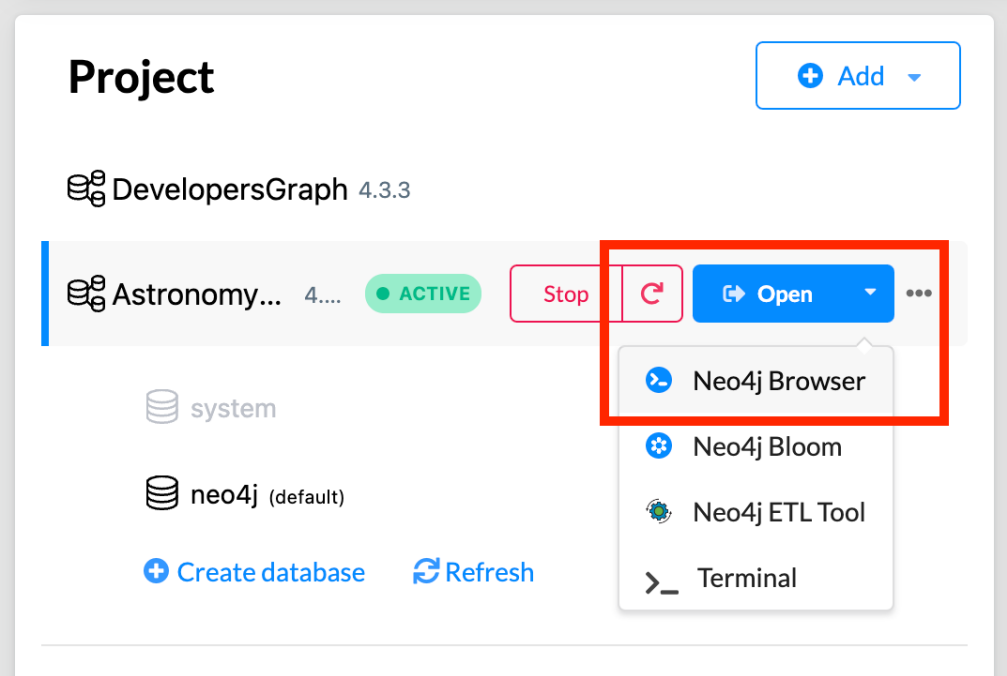
Step 6: Start the Database

- **Action**: Once the database is created, start it by clicking on the “Start” button next to the database name in Neo4j Desktop.
- **Note**: Wait for the database to start fully. You will see a status indicator showing that it’s running.



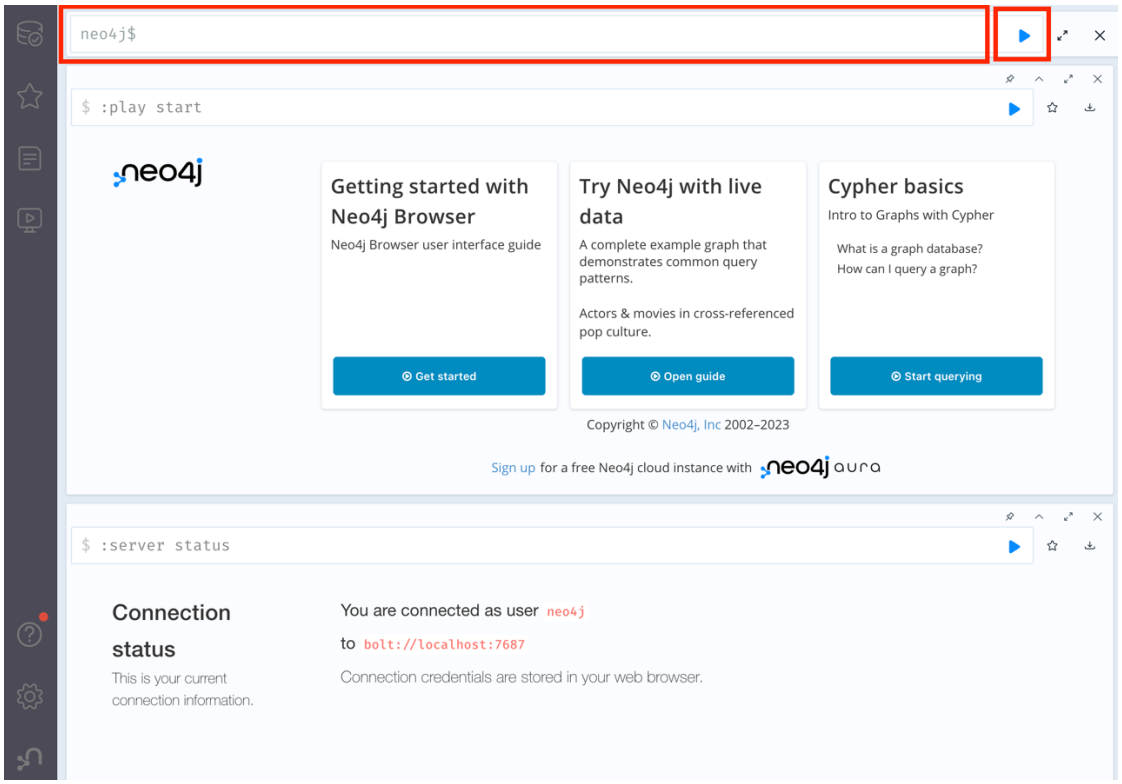
Step 7: Access Neo4j Browser

- **Action**: Ensure your Neo4j database is running. Then, open the Neo4j Browser. This is typically done by clicking on the "Open Neo4j Browser" button in the Neo4j Desktop application.



Step 8: Enter the Cypher Code

- **Action**: In the Neo4j Browser, you will find a command input area where you can type or paste Cypher queries.
- **Steps**:
    a. Copy the provided Cypher code (*AstronomyGraph.cypher* file on Blackboard) for creating the Astronomy Graph.
    b. Paste the part of the code which regards graph creation into the command input area in the Neo4j Browser *(see detailed code analysis below)*.



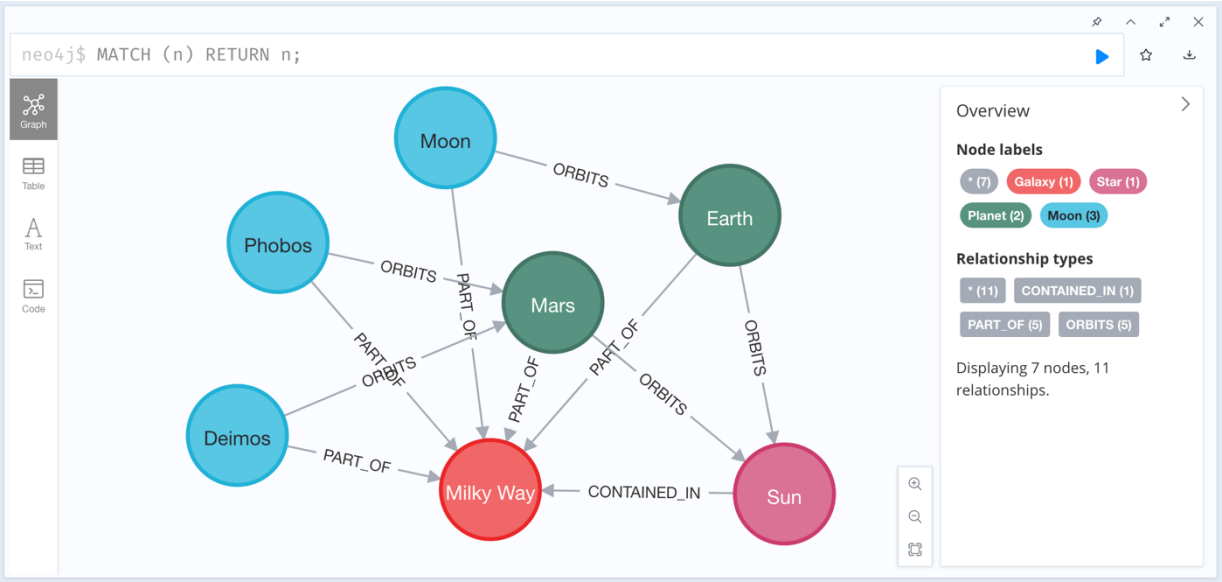
Step 9: Execute the Code

- **Action**: Run the Cypher code to create the graph.
- **Steps**:
    a. After pasting, review the code to ensure it's complete and correct.
    b. Press the "Run" button or hit "Enter" to execute the code.

Step 10: Verify the Graph Creation

- **Action**: After running the code, Neo4j Browser will process it and create the nodes and relationships as defined.
- **Verification**:
    a. Look for any messages in the output area indicating success or errors.
    b. Use the command `MATCH (n) RETURN n;` to display the entire graph and visually verify that all elements are correctly created.

Detailed Guide to Creating an Astronomy Graph



Step 1: Clear Existing Database

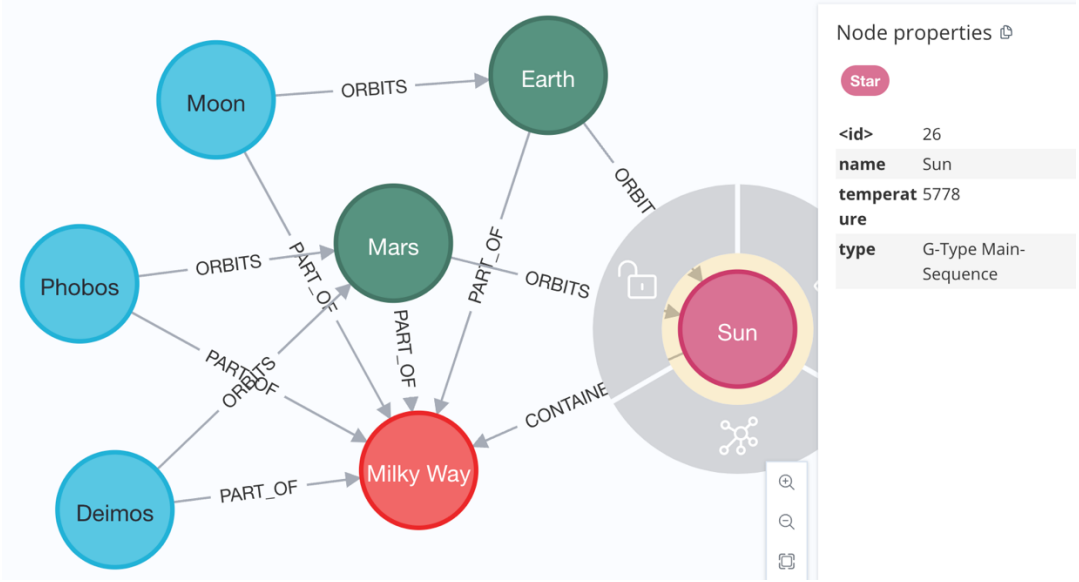
- **Action**: Begin by clearing any existing data in the database. This is crucial to avoid conflicts with the new data you are about to enter.
- **Reference to Code**: Look for the command `MATCH (n) DETACH DELETE n;`. This line of code will remove all current data from the database, so make sure this is what you want to do before running it.

Step 2: Create Nodes for Celestial Bodies

- **Action**: Create nodes for each celestial body, including the Milky Way galaxy, the Sun, planets, and moons. Each node should be defined with specific attributes.
- **Reference to Code**: The `CREATE` command is used here. For example, `(galaxy:Galaxy {name: "Milky Way", type: "Barred Spiral"})` creates a node for the Milky Way galaxy with its name and type.

Step 3: Define Celestial Properties

- **Action**: Assign properties to each celestial body, such as name, type, temperature for stars, and mass and classification for planets and moons.
- **Reference to Code**: Each node creation line includes these properties. For instance, `(star:Star {name: "Sun", type: "G-Type Main-Sequence", temperature: 5778})` defines the Sun with its name, type, and surface temperature.



Step 4: Establish Orbital Relationships

- **Action**: Create relationships that represent the orbital dynamics, like which planets orbit which stars, and which moons orbit which planets.
- **Reference to Code**: Look for the `ORBITS` relationship in the code. For example, `(planet1)-[:ORBITS]->(star)` indicates that `planet1` (Earth) orbits the `star` (Sun).

Step 5: Link Planets and Moons to the Galaxy

- **Action**: Besides orbital relationships, also establish a connection between each celestial body and the galaxy.
- **Reference to Code**: This is done using the `PART_OF` relationship, such as in `(planet1)-[:PART_OF]->(galaxy)`, which links Earth to the Milky Way galaxy.

Step 6: Visualize the Complete Graph

- **Action**: After setting up the nodes and relationships, visualize the entire graph to see the structure of your created universe.
- **Reference to Code**: Use the command `MATCH (n) RETURN n;` to display the graph. This will show all the nodes and relationships you've created.

Additional Tips:

- **Double-Check Relationships**: Ensure that each relationship correctly reflects the real-world astronomical structure.
- **Explore Variations**: Once comfortable, try adding additional celestial bodies or changing properties to see how the graph changes.
- **Discuss Findings**: After creating the graph, discuss its structure and what it represents about our understanding of the universe. This can be a great way to delve deeper into astronomy.

Adding notes about the measurements provided in the properties of the celestial bodies in the Astronomy Graph will enhance the students' understanding. Here's a brief explanation of each:

Notes on Measurements in the Astronomy Graph Properties

1. Galaxy (Milky Way) Properties

- **Type**: "Barred Spiral" - This refers to the galaxy's structure. The Milky Way is classified as a barred spiral galaxy, which means it has a central bar-shaped structure composed of stars and spiral arms that extend from this bar.

2. Star (Sun) Properties

- **Type**: "G-Type Main-Sequence" - This classification indicates the Sun is a medium-sized and medium-brightness star. Main-sequence stars fuse hydrogen into helium in their cores.
- **Temperature**: 5778 Kelvin - This is the surface temperature of the Sun, a key characteristic of its spectral type and luminosity.

3. Planet Properties

- **Name**: Each planet is given its common name, like "Earth" or "Mars".
- **Mass**:
    - Earth: 5.972 x 10^24 kilograms - This is the mass of Earth, giving an idea of its size and gravitational influence.
    - Mars: 0.641 x 10^24 kilograms - Mars' mass is less than Earth's, reflecting its smaller size and lower gravity.
- **Classification**: "Terrestrial" - This indicates that both Earth and Mars are rocky planets with solid surfaces.

4. Moon Properties

- **Name**: The names of moons, like "Moon" (Earth's moon), "Phobos", and "Deimos" (Mars' moons).
- **Mass**:
    - Moon: 0.073 x 10^24 kilograms - Earth's moon has a significant mass, influencing Earth's tides.
    - Phobos: 1.0659 x 10^16 kilograms - Mars' larger moon, but still significantly smaller than Earth's moon.
    - Deimos: 1.4762 x 10^15 kilograms - Mars' smaller moon, with even less mass.
- **Classification**: "Natural Satellite" - This indicates that these bodies are moons, naturally occurring satellites orbiting a planet.

Step 11: Perform the Sample Queries

- **Action**: Use the Queries from the RDF file to retrieve certain information from the created databases. Check the Outputs.

1. **Find all planets and their moons:**

```
MATCH (p:Planet)-[:ORBITS]->(s:Star), (m:Moon)-[:ORBITS]->(p)
RETURN p.name AS Planet, collect(m.name) AS Moons;
```

2. **Find all celestial bodies contained within the Milky Way galaxy:**

```
MATCH (g:Galaxy {name: "Milky Way"})<-[:PART_OF]-(c)
RETURN c.name AS CelestialBody, labels(c) AS Type;
```

3. **Retrieve the mass of all planets:**

```
MATCH (p:Planet)
RETURN p.name AS Planet, p.mass AS Mass;
```

4. **Get the details of the 'Earth' and its moon:**

```
MATCH (p:Planet {name: "Earth"})-[:ORBITS]->(s:Star),
(m:Moon)-[:ORBITS]->(p)
RETURN s.name AS Star, p.name AS Planet, p.mass AS PlanetMass, m.name
AS Moon, m.mass AS MoonMass;
```

5. **Count the number of moons for each planet:**

```
MATCH (p:Planet)<-[:ORBITS]-(m:Moon)
RETURN p.name AS Planet, count(m) AS NumberOfMoons;
```

6. **Query for all celestial bodies with a mass greater than a certain threshold:**

```
MATCH (c)
WHERE c.mass > 0.1 AND (c:Planet OR c:Moon)
RETURN c.name AS Name, c.mass AS Mass, labels(c) AS Type;
```

7. **Find the celestial body with the highest mass:**

```
MATCH (c)
WHERE c:Planet OR c:Moon
RETURN c.name AS Name, c.mass AS Mass
ORDER BY c.mass DESC
LIMIT 1;
```

8. **Find the temperature of stars and classify them as 'hot' or 'cold':**

```
MATCH (s:Star)
RETURN s.name AS Star, s.temperature AS Temperature,
CASE
WHEN s.temperature > 30000 THEN 'Very Hot'
WHEN s.temperature > 10000 THEN 'Hot'
WHEN s.temperature > 7500 THEN 'Medium'
WHEN s.temperature > 3700 THEN 'Cool'
ELSE 'Very Cool'
END AS StarType;
```

Note: According to this classification, the Sun, with a surface temperature of approximately 5,778 K, falls into the G-type category, often described as yellow dwarfs. They are not the hottest stars, but neither are they the coolest.

Task 1. Protégé Ontologies for Beginners Tutorial

Protégé Installation

1. **Download Protégé:**
    - Visit the Protégé website at https://protege.stanford.edu/.
    - Choose the version suitable for your operating system (Windows, macOS, Linux).
2. **Install Protégé:**
    - After downloading, run the installer and follow the instructions to install Protégé on your computer.
3. **Explore Protégé:**

- Familiarise yourself with the different views and tabs.
- The Entities tab lets you see a tree view of your classes, object properties, data properties, and individuals.

4. **Experiment:**

- Try adding more classes, properties, or individuals.
- Use the Reasoner to infer relationships and validate your ontology.

5. **Documentation and Help:**

- Protégé has a comprehensive wiki at https://protegewiki.stanford.edu/.
- There are also many tutorials available online for specific tasks within Protégé.

Guide to Creating an Astronomy Ontology in Protégé

The structure of the provided Astronomy conceptual graph, which can also be described in the RDF/XML format, reflects a basic model of astronomical objects and their characteristics. Here are some key points regarding its structure:

1. **Hierarchical Class Structure**:
    - The ontology defines a hierarchy of classes under the general class CelestialBody.
    - Galaxy, Star, Planet, and Moon are all subclasses of CelestialBody. This implies that each specific type (like a star or a planet) is a kind of celestial body.
2. **Datatype Properties**:
    - The ontology includes datatype properties like hasName, hasType, hasTemperature, and hasMass.
    - These properties are used to describe attributes of celestial bodies. For example, hasName is a string that provides the name of the celestial body, while hasMass is a float indicating the mass.
3. **Domains and Ranges**:

- Each property is associated with a domain and a range. The domain indicates the class of objects to which the property can be applied, and the range specifies the type of data that the property can have.
- For instance, hasTemperature is applicable to objects of the Star class and has a range of xsd:float, indicating that the property value should be a floating-point number.

4. **Individuals**:
   - The ontology defines specific individuals (instances) of the classes, such as MilkyWay (an instance of Galaxy), Sun (an instance of Star), and Earth (an instance of Planet).
   - These individuals are given specific values for the properties defined in the ontology. For example, Sun has a hasTemperature property with the value 5778.
5. **Annotations**:
   - Classes in the ontology are annotated with rdfs:label and rdfs:comment.
   - The rdfs:label provides a human-readable name for the class, and rdfs:comment offers a description or comment about the class, which is useful for understanding the intended meaning of the class.

Note: In an astronomical context, while moons, stars, and planets are indeed components of galaxies, they are not subclasses of galaxies in the ontological sense. In ontology, especially in the context of OWL and RDF, the subclass relationship (rdfs:subClassOf) is used to represent a strict "is a" relationship. A subclass relationship implies that every instance of the subclass is also an instance of the superclass.

- **Galaxies** are large systems that contain stars, planets, moons, dust, gas, and dark matter. They are not individual celestial objects but systems or collections of such objects.
- **Stars** are luminous spheres of plasma held together by their own gravity.
- **Planets** are astronomical objects orbiting a star or stellar remnant.
- **Moons** (or natural satellites) are celestial bodies that orbit planets.

In ontological terms, stars, planets, and moons are types of celestial bodies, but they are not types of galaxies. They are components within a galaxy. Therefore, in your ontology, each of these entities should be a subclass of CelestialBody, not of Galaxy.

If you want to represent that these entities are part of a galaxy, you would typically use object properties (such as containedIn or partOf) rather than subclass relationships. For example, you might have an object property partOf linking a Star individual to a Galaxy individual, indicating that the star is part of that galaxy. This represents a compositional relationship rather than a subclass relationship.

In summary, Galaxy, Star, Planet, and Moon are all subclasses of CelestialBody, but they should not be subclasses of each other in the ontology. Instead, their relationships (like being part of a galaxy) are best represented through object properties connecting individuals of these classes.

Students can follow these steps to create an ontology in Protégé that mirrors the structure defined in the provided *AstronomyGraph.rdf*. This guide assumes a basic familiarity with Protégé's interface:

1. Open Protégé and Create a New Ontology:

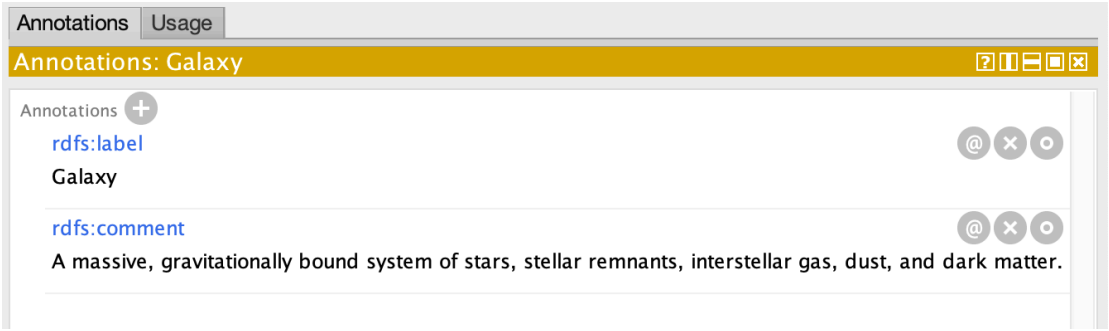
- Open Protégé.
- Go to File > New Ontology.
- Enter the IRI (e.g., http://example.org/ontology) and click OK.

2. Create Classes:

- Go to the Classes tab on the left side.
- Right-click on owl:Thing, select Create subclass, and name it CelestialBody.
- Repeat this step to create subclasses of CelestialBody named Galaxy, Star, Planet, and Moon.

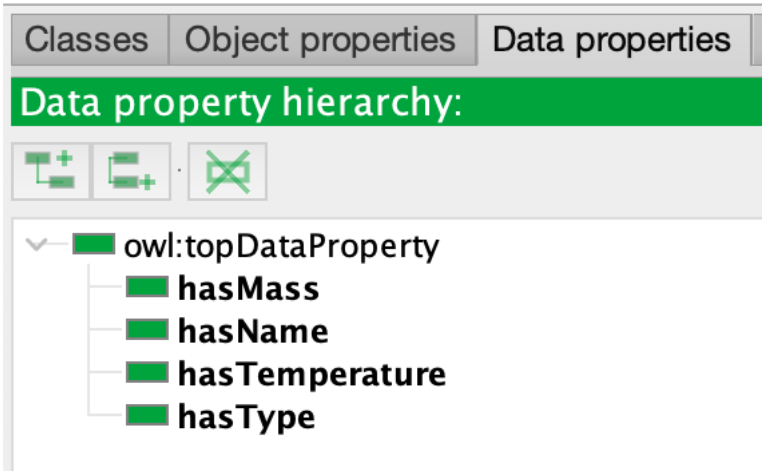
3. Add Annotations to Classes:

- Select a class (e.g., CelestialBody).
- In the Annotations view, add a rdfs:label (e.g., "Celestial Body") and a rdfs:comment (e.g., "A general class for any celestial object.")



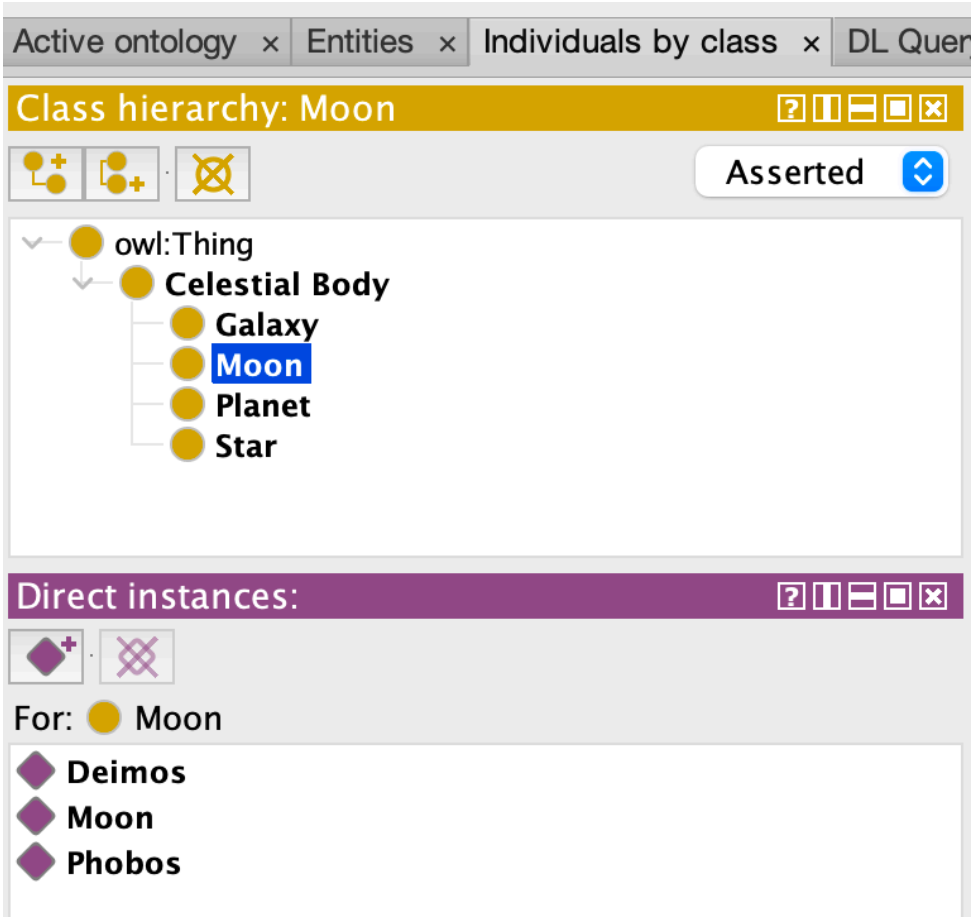
4. Create Datatype Properties:

- Go to the Object properties tab.
- Create new datatype properties: hasName, hasType, hasTemperature, and hasMass.
- For each property, specify the domain and range. For example, hasName and hasType have the domain CelestialBody and range xsd:string.



5. Create Individuals:

- Go to the Individuals tab.
- For each class, create individuals (e.g., for Galaxy, create an individual named MilkyWay).
- Add data property assertions to each individual. For example, for MilkyWay, add the properties hasName with value "Milky Way" and hasType with value "Barred Spiral".



6. **Save the Ontology:**

   - Go to File > Save As.
   - Choose the location and save your ontology.

This process will create an ontology in Protégé that closely matches the structure of the RDF/XML content provided. Students can compare the manually created ontology in Protégé with the RDF/XML to understand how the same structures are represented in different formats.

Overall, the ontology provides a simple yet illustrative example of how celestial bodies can be modelled in an ontology. It captures the essential characteristics of these bodies and their hierarchical relationships. Such an ontology can be used for educational purposes or as a basis for *more complex astronomical data modelling*.