AI-Driven Job-Matching Service: An Ontology-Based Approach

## 1. Business Context and Justification of Approach

## Introduction to the Job-Matching Service

Modern day job matching services like indeed, linked in etc face many challenges, like mismatched candidate skills, inefficient filtering of irrelevant opportunities, and inadequate personalization. These systems usually rely on keyword based algorithms - which were good for a while but they fail to capture subtle relationships between job requirements (e.g., hybrid roles requiring both technical and soft skills) and candidate profiles (e.g., project based experience in specific industries). This gap results in suboptimal matches and dissatisfaction for both employers and job seekers.

## Why Ontology and AI?

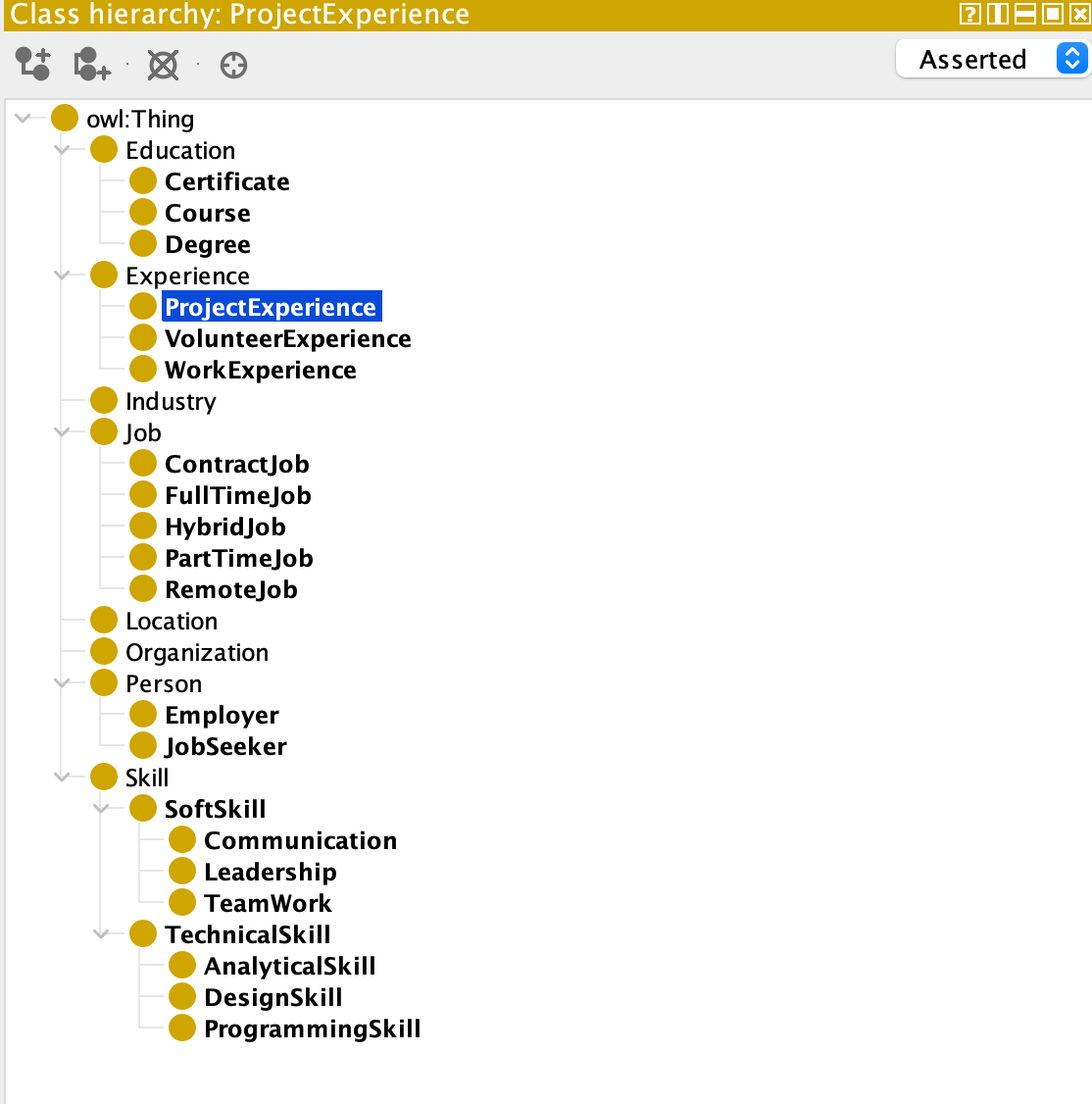
Our proposed ontology addresses these limitations through the creation of domain knowledge structures that link concepts semantically. For eg. the class hierarchy separates TechnicalSkill from SoftSkill - like Python and Java for TechnicalSkill and TeamWork and Communication for SoftSkill. This allows for detailed matching that goes beyond simple keyword matching. The formalized relationships between requiresSkill and hasSkill enable job specific competency requirements and prefersIndustry and belongsToIndustry relationships provide sector-specific context to opportunities in Technology and Finance.

The structured information enables AI systems to reason about job seeker qualifications such as the eligibility of a candidate with a ComputerScienceBachelors degree and MobileAppProject experience for SeniorDeveloper positions that need JavaScript skills. The ontology functions as a dynamic query system for eg, we can use "Find candidates in Dubai with 5+ years of Python experience” - which is not possible in static databases.

## **Ontology Design Decisions**

The ontology’s core classes - JobSeeker, Job, Skill, Education, and Location—are connected via object properties like hasEducation and jobLocatedIn, which enforces semantic consistency. For example:

* JobSeeker is linked to Skill via hasSkill, with subclasses like ProgrammingSkill and DesignSkill enabling specialized queries.
* Job uses requiresExperience to mandate prior roles. e.g., SoftwareEngineerEmaar for a SeniorDeveloper position.
* Data properties like hasSalaryRange (for jobs) and hasSalaryExpectation (for candidates) facilitate compatibility checks.



This design improves accuracy by codifying tacit requirements (e.g., a DataScientist role preferring candidates with DataScienceCertificate credentials) and supports scalability through modular industry-specific extensions.

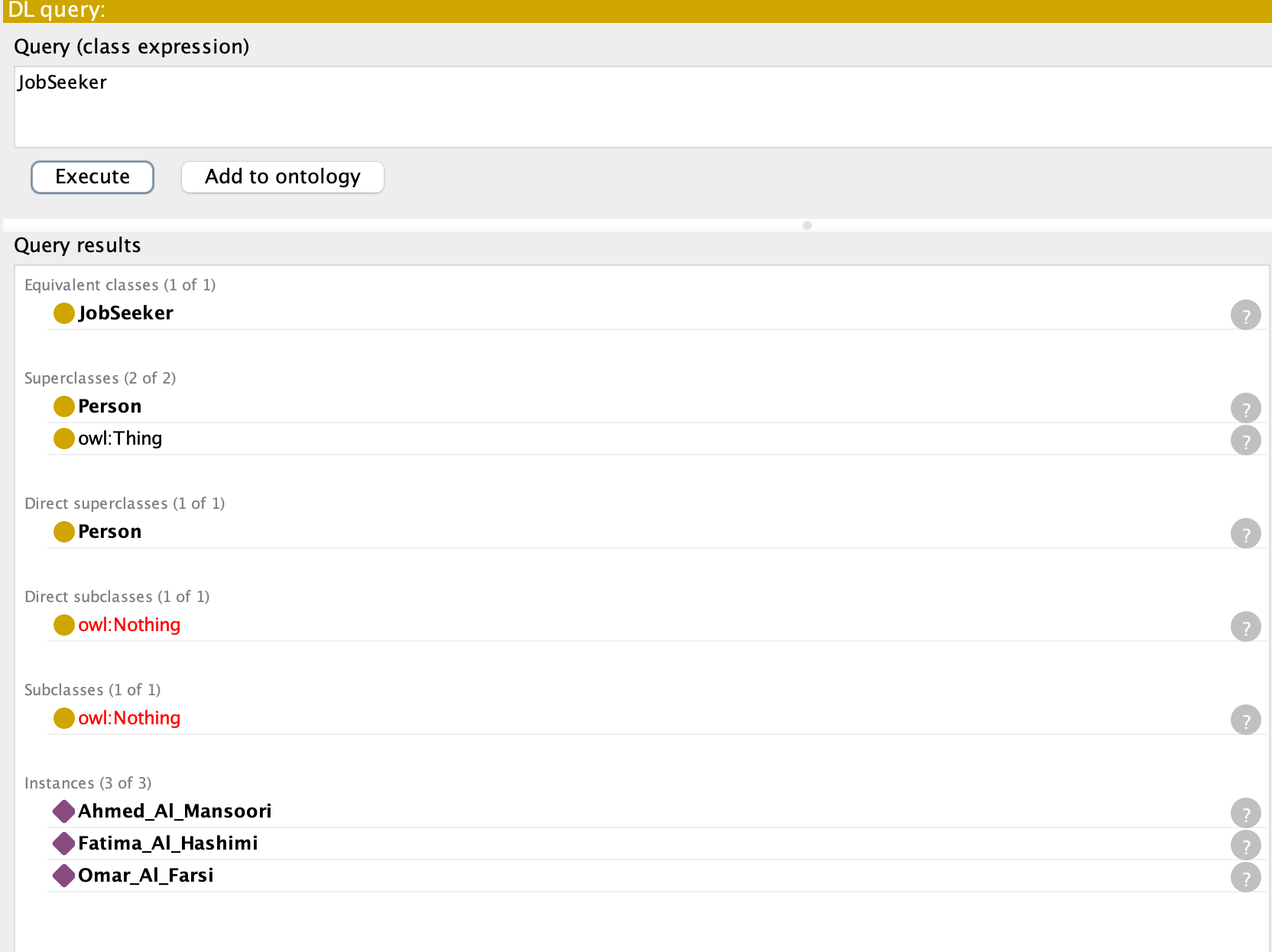
# 2. Analysis of Outputs and Testing

To evaluate how effective our job matching ontology is, we conducted a series of tests within the Protégé environment. These tests involved executing Description Logic (DL) queries to simulate the core functionalities required by an AI-driven job-matching service. This approach allowed us to verify the ontology's structure, consistency, and its ability to represent and retrieve relevant information accurately.

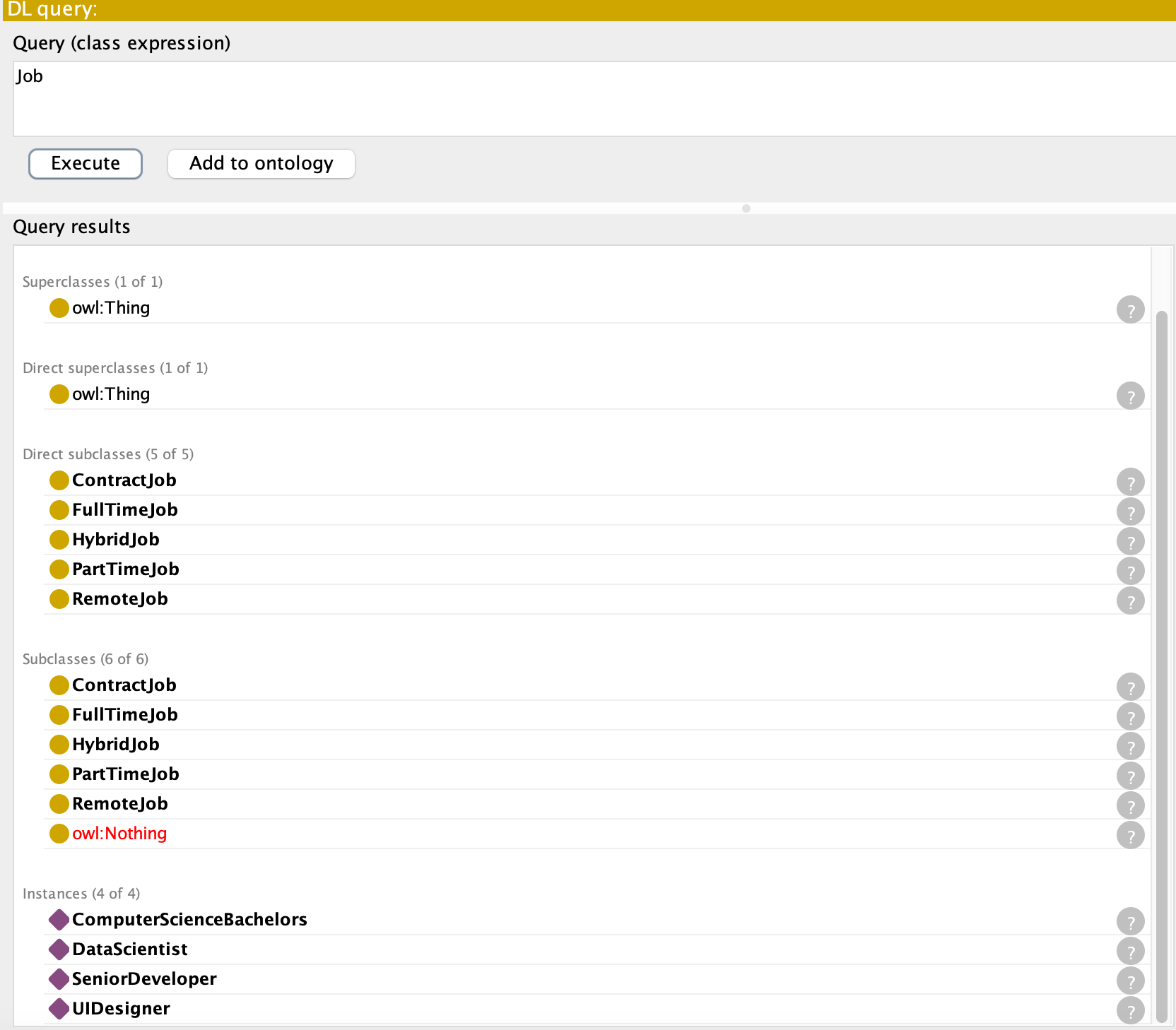
## Initial Verification and Population Checks

We began with basic queries to confirm the presence and classification of fundamental entities within the ontology. Executing simple class expressions allowed us to retrieve all instances of key concepts:

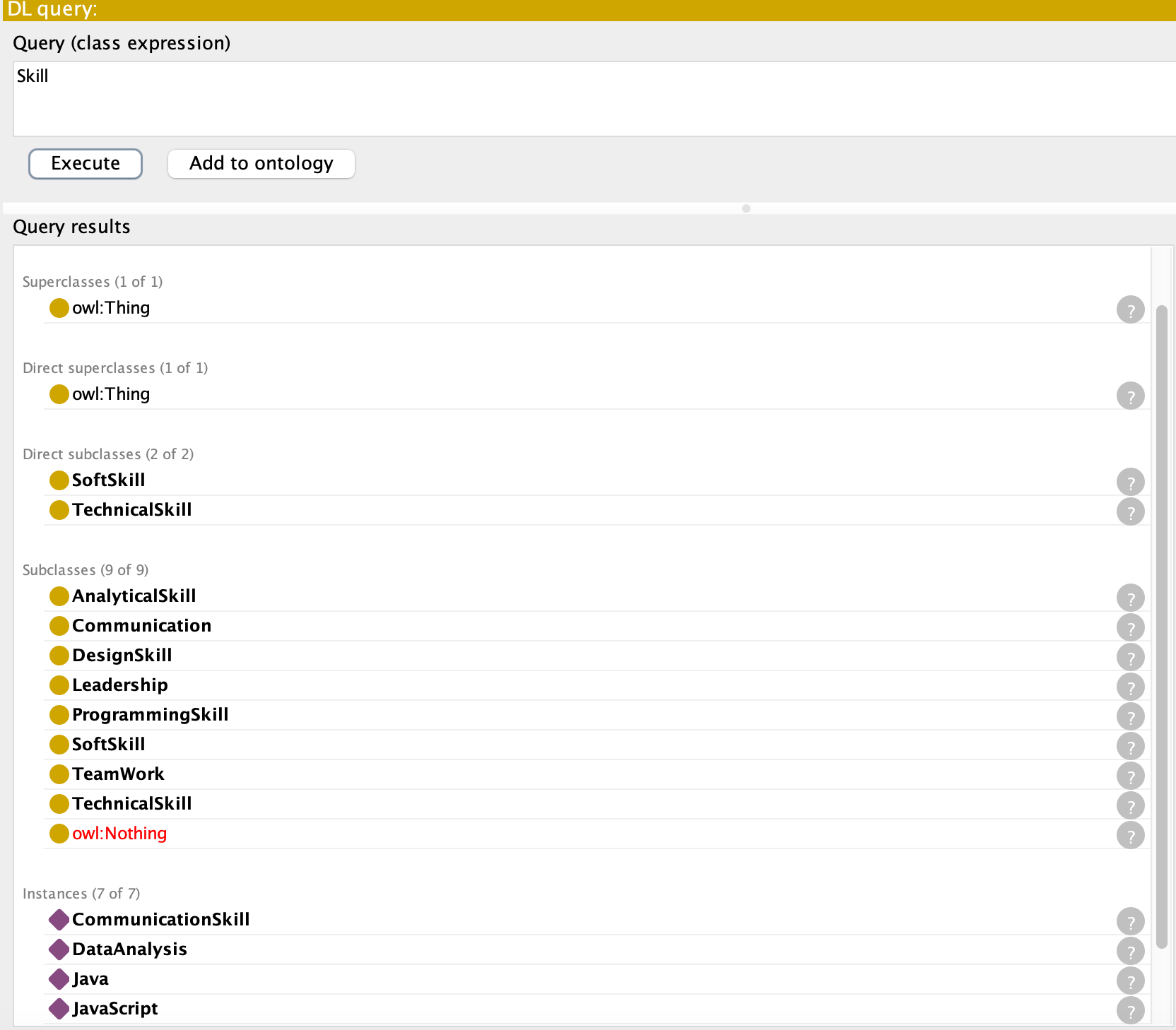
* JobSeeker: This query successfully retrieved all individuals classified as Job Seekers, such as Ahmed\_Al\_Mansoori, Fatima\_Al\_Hashimi, and Omar\_Al\_Farsi



* Job: Similarly, this query listed all job postings defined in the ontology, including SeniorDeveloper, DataScientist, and UIDesigner



* Skill: This query returned all defined skills, encompassing both technical (Python, Java, DataAnalysis, PhotoshopSkill) and soft skills (CommunicationSkill, TeamworkSkill)

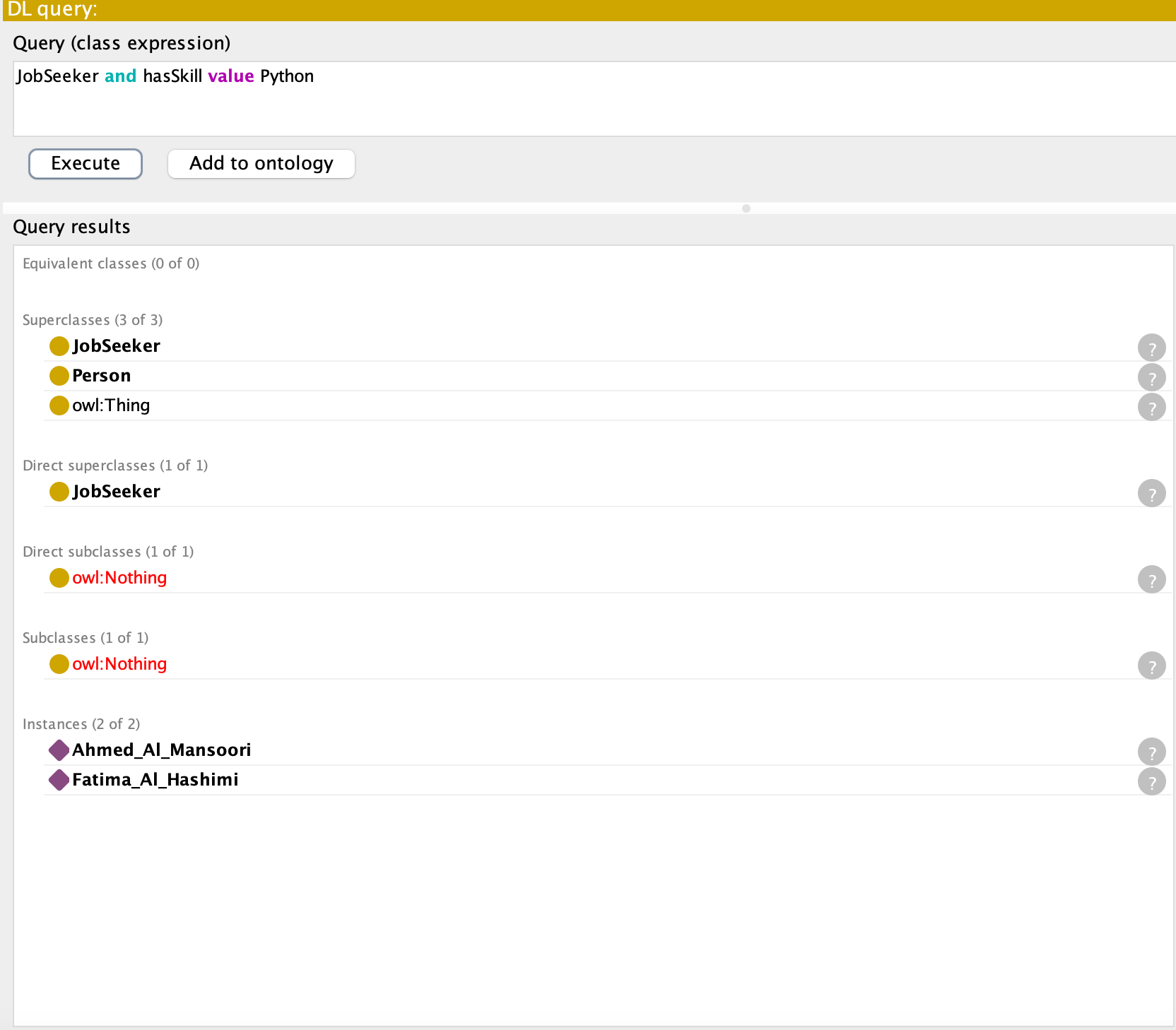


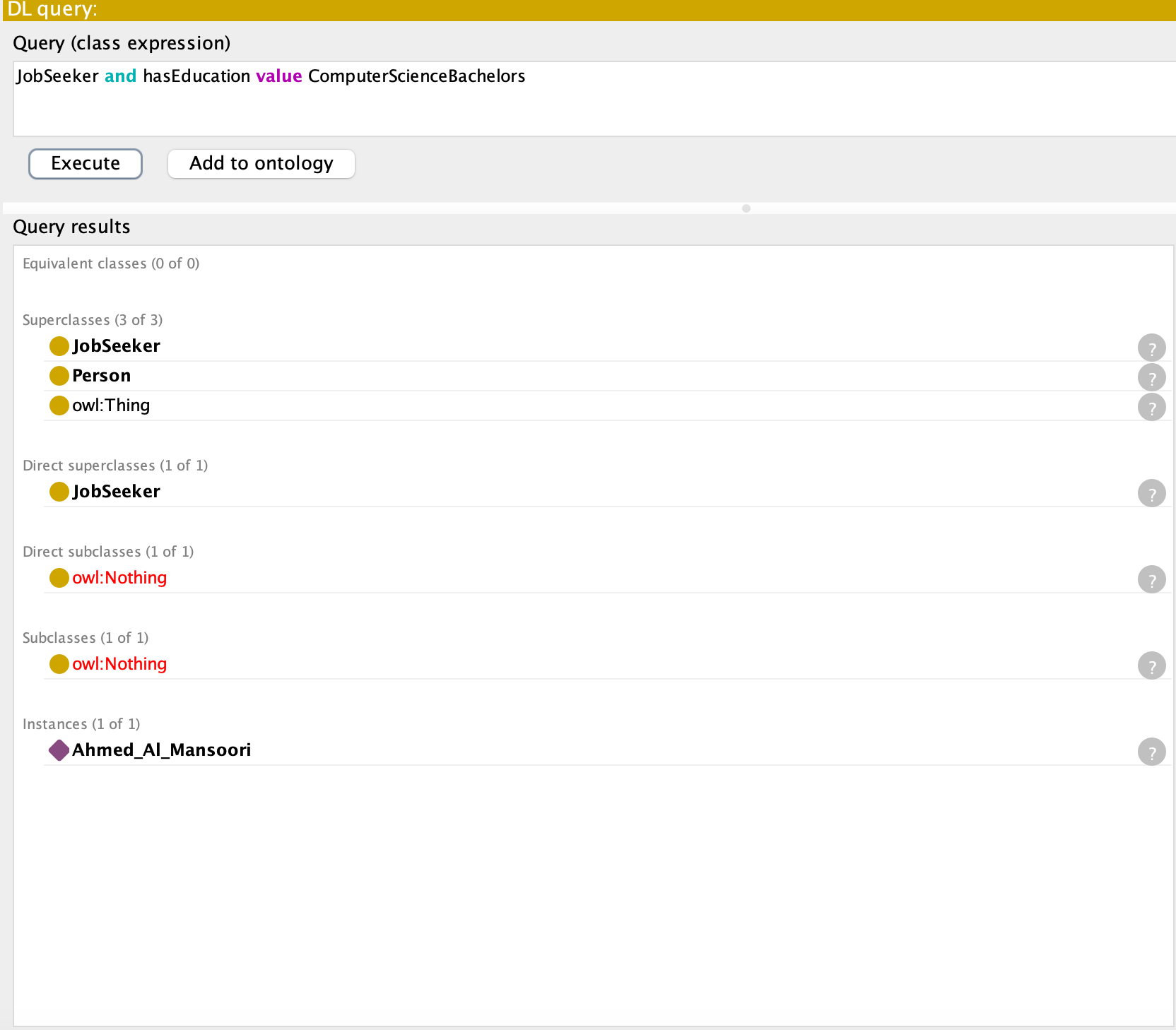
These foundational queries confirmed that our ontology was correctly populated with diverse instances representing the core entities of the job-matching domain, providing a solid base for more complex reasoning tasks.

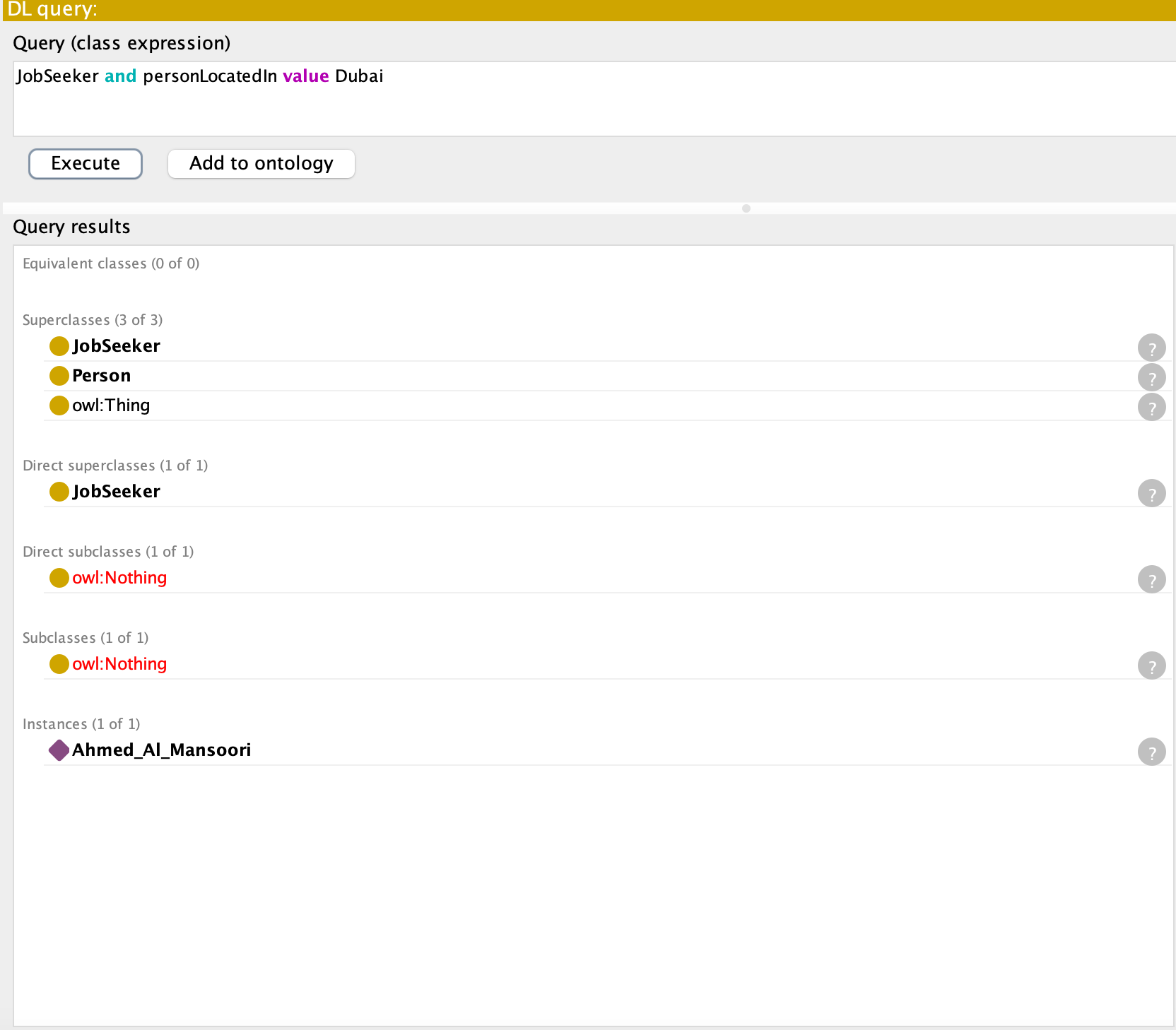
## Testing Job Seeker Filtering Capabilities

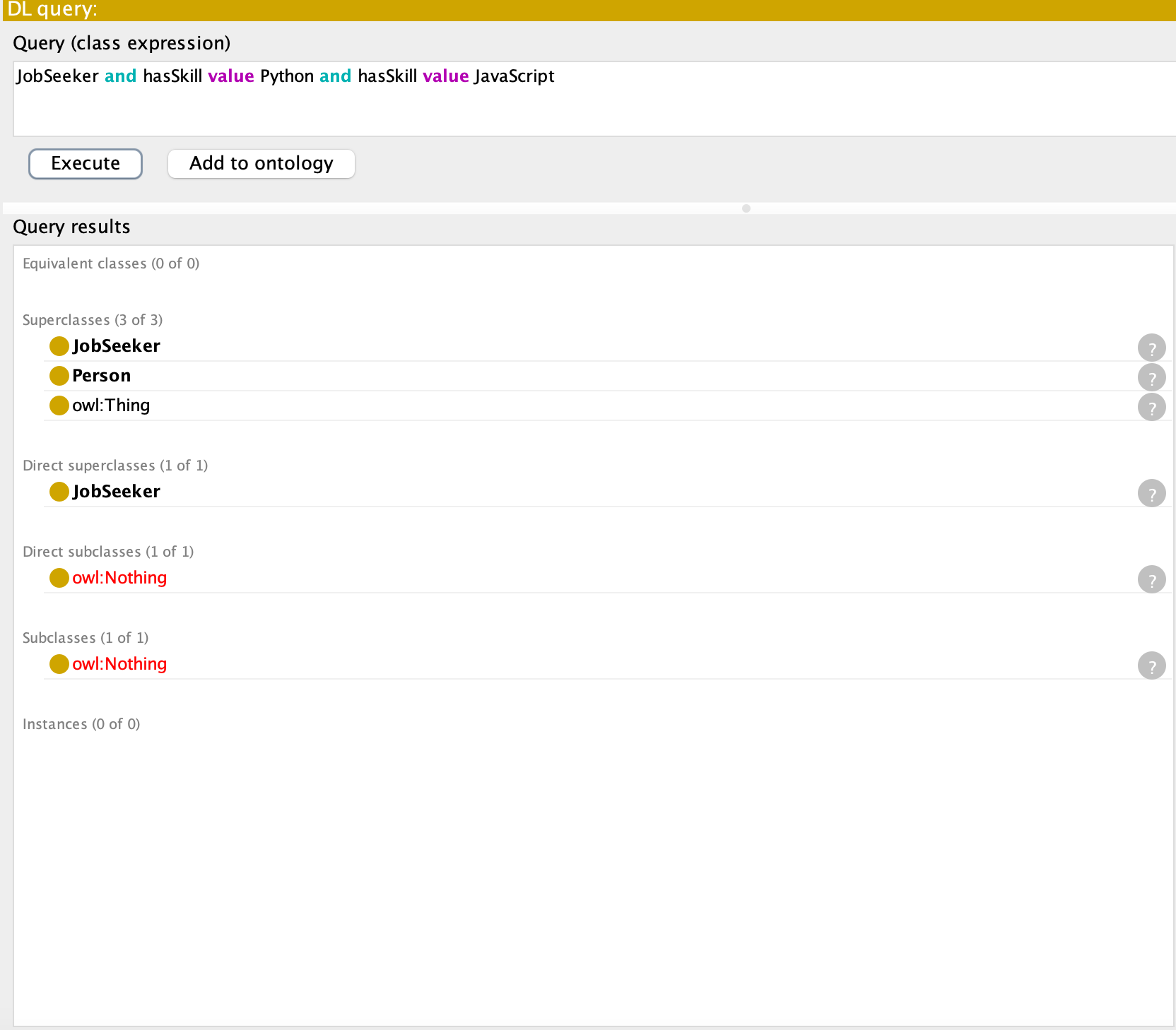
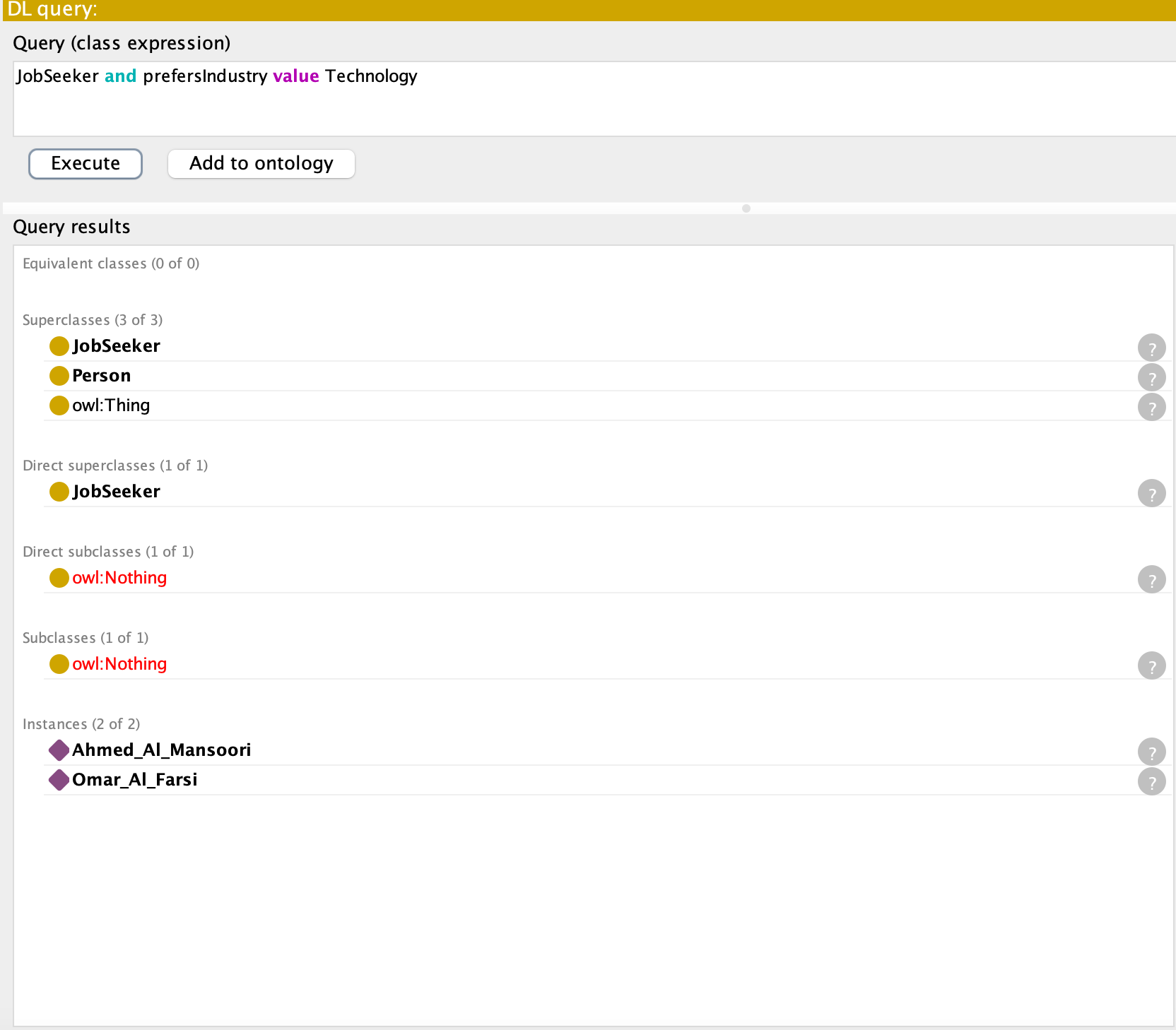
More specific queries were designed to test the ontology's ability to filter job seekers based on common recruitment criteria. These queries leveraged the defined object properties (hasSkill, hasEducation, personLocatedIn, prefersIndustry):

* Filtering by Skill: The query JobSeeker and hasSkill value Python correctly identified Ahmed\_Al\_Mansoori and Fatima\_Al\_Hashimi as job seekers possessing the Python skill. This demonstrates the ability to match candidates based on specific technical competencies.



* Filtering by Education: JobSeeker and hasEducation value ComputerScienceBachelors successfully isolated Ahmed\_Al\_Mansoori, who holds the specified degree. This highlights the capacity to filter based on formal qualifications.  
  
* Filtering by Location: JobSeeker and personLocatedIn value Dubai accurately returned Ahmed\_Al\_Mansoori, confirming the system can handle geographic constraints.

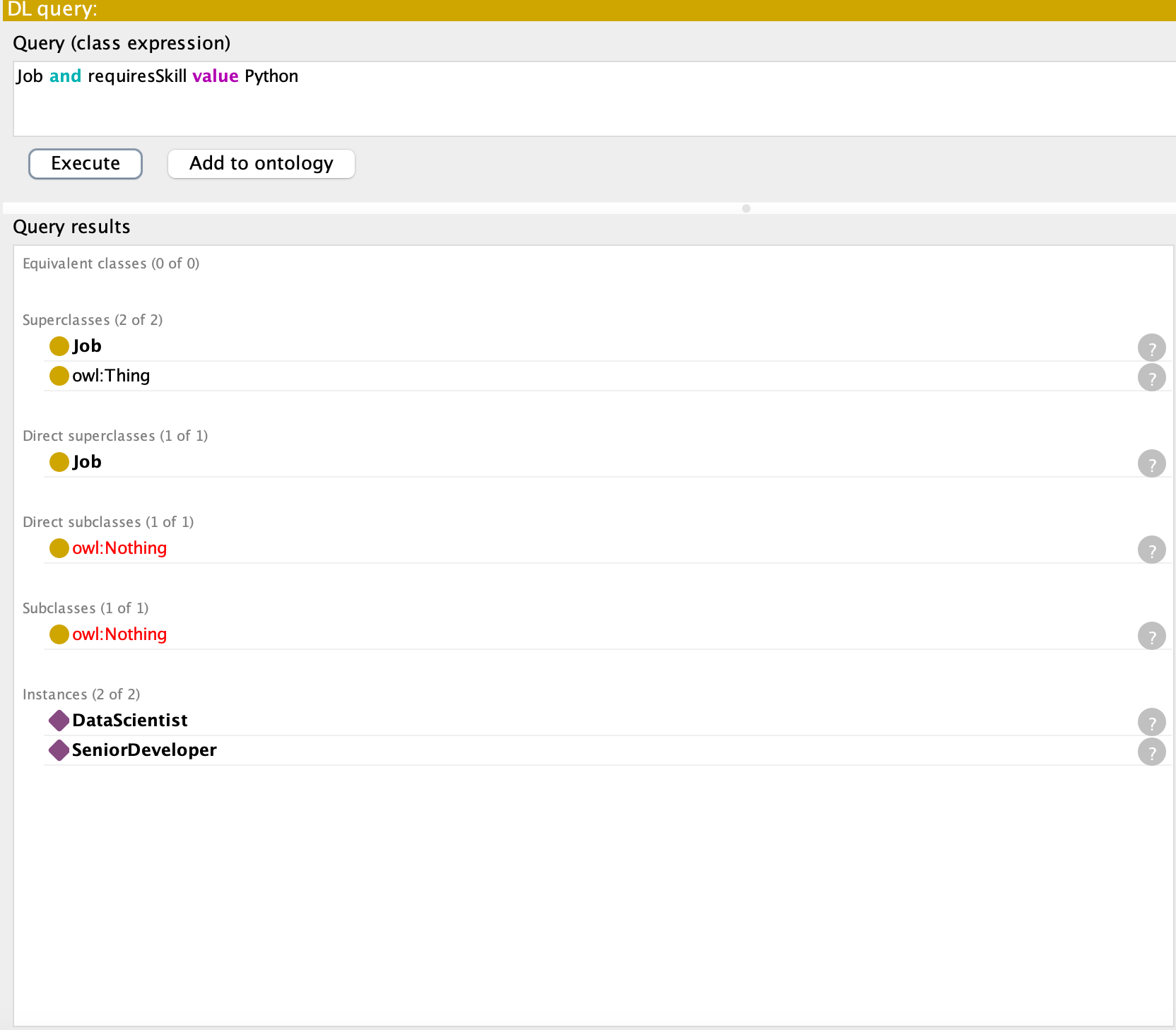


* Filtering by Multiple Skills: The query JobSeeker and hasSkill value Python and hasSkill value Java pinpointed Ahmed\_Al\_Mansoori who possesses both skills, showcasing the ontology's support for multi-criteria filtering essential for complex roles.  
  
* Filtering by Industry Preference: JobSeeker and prefersIndustry value Technology identified Ahmed\_Al\_Mansoori and Omar\_Al\_Farsi, demonstrating the ability to consider candidate preferences alongside qualifications.  
    
  These tests collectively show that the ontology effectively models job seeker attributes and allows for precise retrieval based on diverse requirements, mirroring the filtering needs of a real-world recruitment platform.

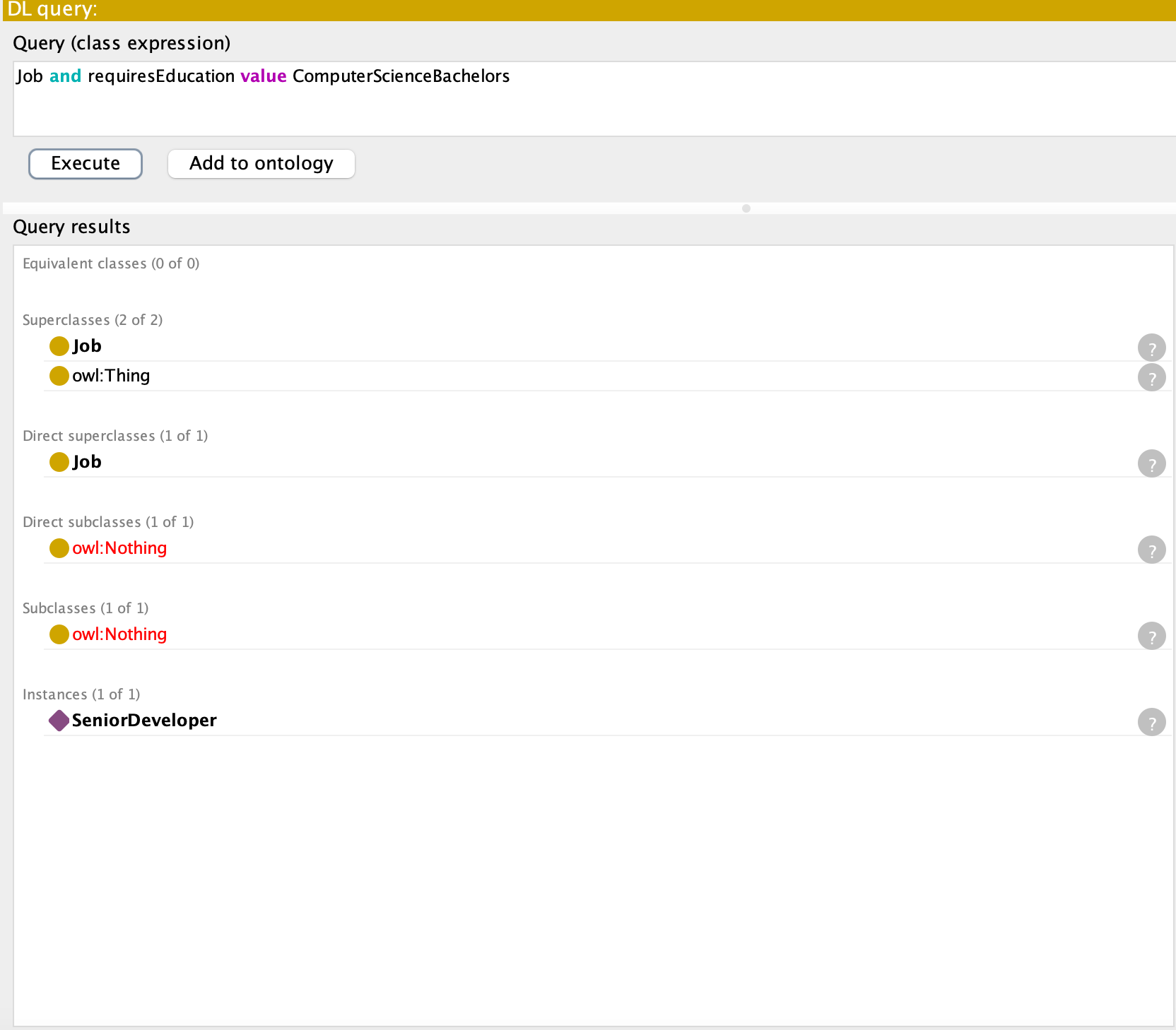
## Testing Job Filtering Capabilities

Complementary queries were executed to test the retrieval of job postings based on their requirements, utilizing properties like requiresSkill, requiresEducation, jobLocatedIn, and belongsToIndustry:

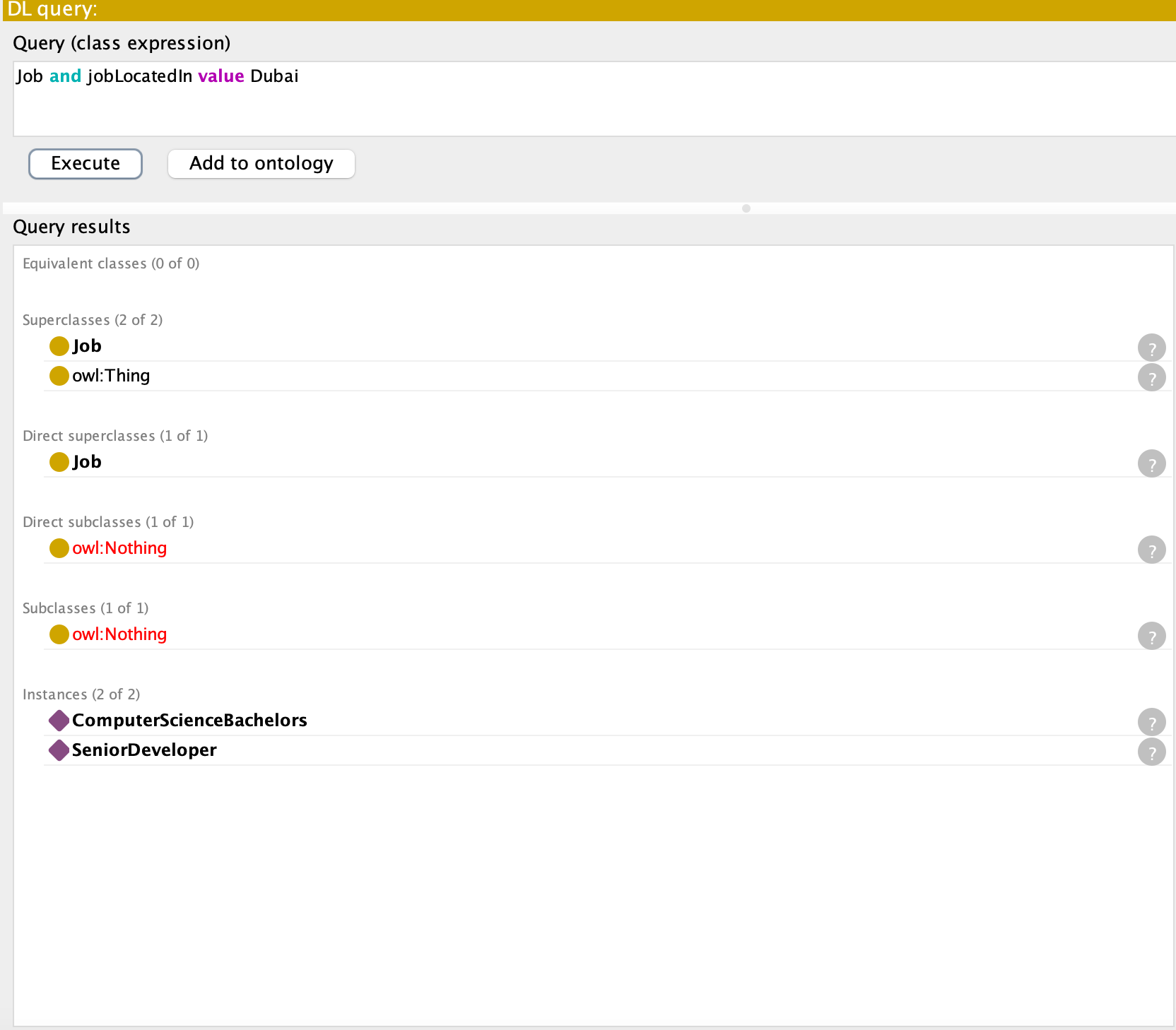
* Filtering by Required Skill: Job and requiresSkill value Python successfully identified the SeniorDeveloper and DataScientist roles, demonstrating the core matching capability of finding jobs demanding specific skills.



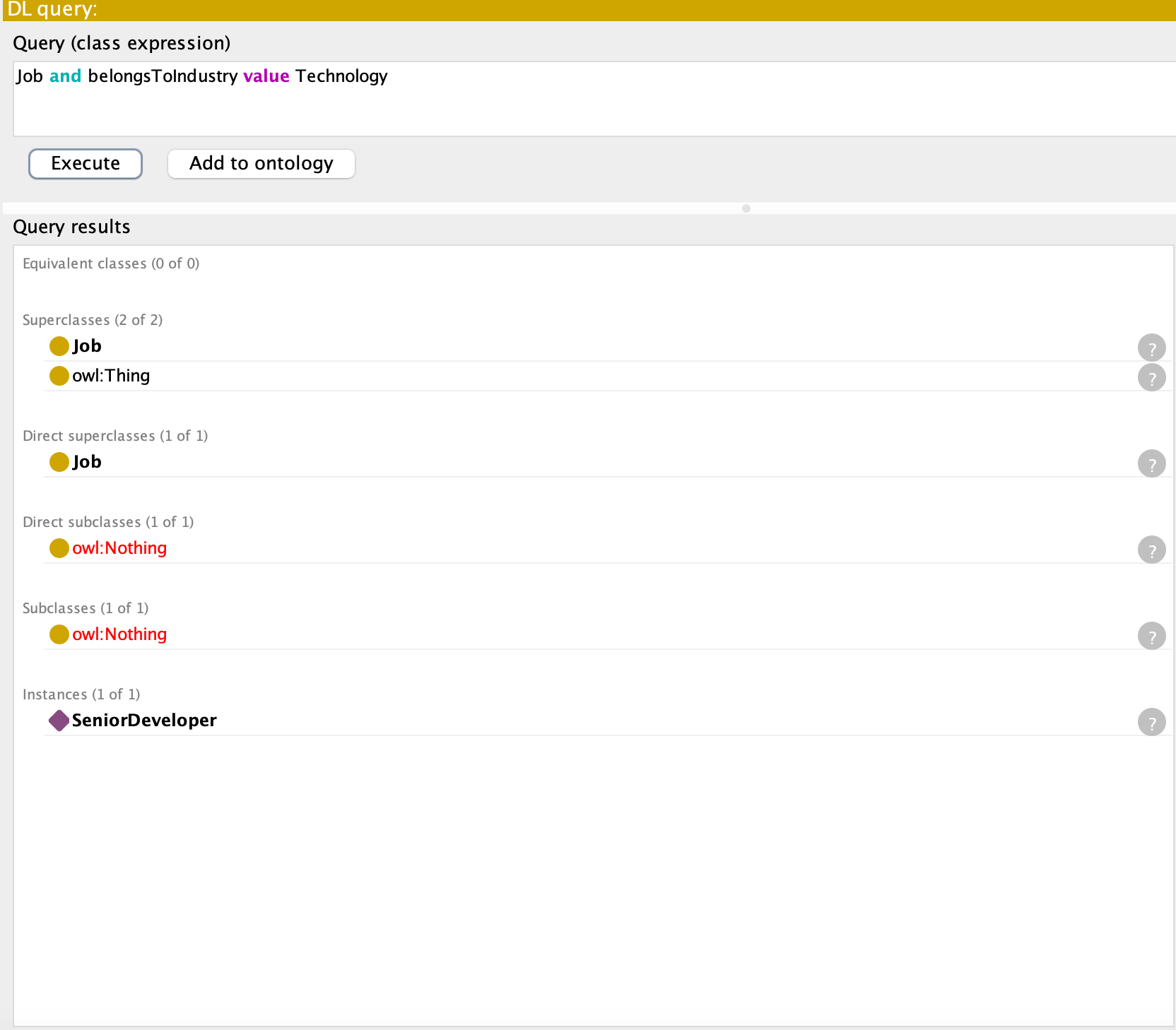
* Filtering by Required Education: Job and requiresEducation value ComputerScienceBachelors correctly returned the SeniorDeveloper position, confirming filtering based on educational prerequisites.



* Filtering by Location: Job and jobLocatedIn value Dubai accurately identified the SeniorDeveloper role located in Dubai.



* Filtering by Industry: Job and belongsToIndustry value Technology returned the SeniorDeveloper role, showcasing the ability to categorize and retrieve jobs by sector.



These query results validate that the ontology structure enables efficient searching and filtering of job opportunities based on their defined characteristics and requirements.

## Supporting the Job-Matching Process

These tests demonstrate that our ontology provides direct support for the job-matching process. The ontology establishes formal definitions for JobSeeker and Job entities with their attributes (skills, education, location, industry) and their relationships (hasSkill, requiresSkill, etc.) which create a structured unambiguous knowledge base. The ontology enables an AI system to perform advanced queries which go beyond basic keyword searches. The system can establish that a JobSeeker with ProgrammingSkill qualification would be appropriate for a job position that needs TechnicalSkill because ProgrammingSkill is a specialized form of TechnicalSkill.

The specific properties hasSkill for job seekers and requiresSkill for jobs enable exact matching logic which identifies candidates whose abilities match job requirements. The semantic structure provides better accuracy and nuanced and context-specific matching capabilities than standard approaches enable.

## Scalability and Adaptability

We designed the ontology with scalability and adaptability in mind. The hierarchical structure of classes (e.g., Skill subdivided into TechnicalSkill and SoftSkill, which are further specialized into ProgrammingSkill, AnalyticalSkill, Communication, etc.) allows for easy extension. New skills, job types (FullTimeJob, PartTimeJob, ContractJob, RemoteJob), industries, locations, or educational qualifications can be added as new subclasses or individuals without requiring fundamental changes to the core structure. For example, adding a new industry like 'Healthcare' simply involves creating a new Industry individual and linking relevant jobs and job seekers using the existing belongsToIndustry and prefersIndustry properties. Similarly, diverse candidate profiles with unique combinations of skills, experiences, and education levels can be readily represented by creating new JobSeeker individuals and asserting the relevant properties. This inherent flexibility ensures the ontology can evolve to accommodate the dynamic nature of the job market and adapt to different job sectors or increasingly complex candidate profiles over time.

## Limitations and Improvements

Testing revealed two limitations:

1. Skill Ambiguity: The ontology does not distinguish between Python (programming) and Python (data analysis) contexts, risking false positives.
2. Temporal Constraints: Properties like hasGraduationDate are not used to filter outdated credentials.

Future iterations could introduce skillContext subproperties and temporal reasoning rules to address these gaps.

## **3. Scalability and Real-World Applicability**

The ontology’s modular structure allows seamless integration of new industries (e.g., healthcare) by extending the Industry class and adding sector-specific skills. For example, a Healthcare subclass could introduce MedicalCertification requirements and PatientManagement skills.

Adaptability to diverse profiles is achieved through polymorphic relationships:

* PartTimeJob and RemoteJob subclasses accommodate flexible work preferences.
* VolunteerExperience and ProjectExperience enable non-traditional candidates to demonstrate competencies.

This flexibility ensures relevance across sectors while maintaining logical consistency.

## **4. Conclusion**

The ontology demonstrates how formal knowledge representation enhances AI-driven job matching by encoding domain semantics, enabling precise reasoning, and supporting personalized outcomes. While limitations exist in handling contextual skill variations, the framework provides a robust foundation for scalable, adaptive recruitment solutions.

**Appendix**

