Graph DBs (Neo4j) exercises

Assume a graph consisting of students, courses, projects, rooms, and relationships between. The nodes and relationships are specified in Cypher (the declarative graph query language used by Neo4j), as given below.

Nodes

```
CREATE (s1: Student {studentID: "1", lastName: "Doe", firstName: "Ana", middleName: "Maria"})
CREATE (s2: Student {studentID: "2", lastName: "Ung", firstName: "Peter", middleName: "John"})
CREATE (s3: Student {studentID: "3", lastName: "Doe", firstName: "John"})
CREATE (s4: Student {studentID: "4", lastName: "Berre", firstName: "Stine"})
CREATE (s5: Student {studentID: "5", lastName: "Travolta", firstName: "John"})
CREATE (c1: Course {courseNr: "1", courseName: "Databases"})
CREATE (c2: Course {courseNr: "2", courseName: "Programming"})
CREATE (c3: Course {courseNr: "3", courseName: "Graphics"})
CREATE (p1: Project {projectNr: "34", projectName: "eCommerce database"})
CREATE (p2: Project {projectNr: "24", projectName: "eCommerce website"})
CREATE (p3: Project {projectNr: "13", projectName: "User interface"})
CREATE (p4: Project {projectNr: "26", projectName: "Reporting"})
CREATE (r1: Room {roomName: "Pascal"})
CREATE (r2: Room {roomName: "Seminar C"})
CREATE (r3: Room {roomName: "Alpha"})
CREATE (r4: Room {roomName: "Beta"})
```

Relationships

```
CREATE (c1) – [:TAKESPLACEIN] –> (r1)
CREATE (c1) – [:TAKESPLACEIN] –> (r3)
CREATE (c1) – [:TAKESPLACEIN] –> (r4)
CREATE (c2) – [:TAKESPLACEIN] –> (r2)
CREATE (s1) – [:ENROLLEDIN] –> (c1)
CREATE (s2) – [:ENROLLEDIN] –> (c1)
CREATE (s3) – [:ENROLLEDIN] –> (c2)
CREATE (s4) – [:ENROLLEDIN] –> (c1)
CREATE (s1) – [:WORKSON {hours: "1"} ] –> (p1)
CREATE (s1) – [:WORKSON {hours: "2"} ] –> (p2)
CREATE (s2) – [:WORKSON {hours: "3"} ] –> (p1)
CREATE (s2) – [:WORKSON {hours: "4"} ] –> (p2)
CREATE (s2) – [:WORKSON {hours: "1"} ] –> (p3)
CREATE (s2) – [:WORKSON {hours: "1"} ] –> (p4)
CREATE (s3) – [:WORKSON {hours: "1"} ] –> (p1)
CREATE (s3) – [:WORKSON {hours: "2"} ] –> (p2)
CREATE (s3) – [:WORKSON {hours: "3"} ] –> (p4)
```

Specify the following queries in Cypher and execute them in Neo4j.

1. In which rooms does course with course number "1" take place in? Retrieve the course name and the names of the rooms in which the course takes place.

2. How many hours and in which projects does student with student number "1" works on? Retrieve the first name of the student, the project the student works on and the corresponding number of hours worked on the project.

3. Which students and how many hours do they work on the project with project number "24"? Retrieve the project name, the last name of the student and the corresponding number of hours worked on the project.

4. Which students work in which projects and how many hours? Retrieve the last name of the students, the name of the projects they work on, and the corresponding number of hours. Order the results by the last name of the students. Limit the results to four.

5. Which students work on more than two projects and on how many projects exactly? Retrieve the last name of the students and the corresponding number of projects. Order the results by the number of projects.

6. Which students have the same last name and work on the same projects? Retrieve the first name of the students and the name of projects they share.

Background materials

Get started with Neo4j: http://neo4j.com/docs/developer-manual/current/get-started/
Documentation for the Cypher query language: http://neo4j.com/docs/developer-manual/current/cypher/

Basic/Simplified syntax of some common Cypher clauses:

- Create nodes
  CREATE <node, optional labels and properties>
- Create relationships
  CREATE <relationship, relationship type and optional properties>
- Delete nodes or relationships:
  DELETE <nodes or relationships>
- Find nodes and relationships that match a pattern
  MATCH <pattern>
- Specify aggregates and other query variables
  WITH <specifications>
- Specify conditions on the data to be retrieved
  WHERE <condition>
- Specify data to be returned
  RETURN <data>
- Order the data to be returned
  ORDER BY <data>
- Limit the number of returned data items
  LIMIT <max number>