Pill Neo4j Solutions

Create a graph in which we represent a series of courses and students, where students are associated with courses through a realized type relationship.

NODE LIST:

- COURSE: JAVA, course: 'Standard Java Programming', duration: 120, price: 80
- COURSE: ANGULAR, course: 'Angular', duration: 30, price: 110
- COURSE: SPRING, course: 'Spring', duration: 80, price: 200
- STUDENT: PEPE, name: 'Pepe', age: 20
- STUDENT: ANA, name: 'Ana', age: 40
- STUDENT: ELENA, name: 'Elena', age: 34
- STUDENT: MARIO, name: 'Mario', age: 19

RELATIONSHIP LIST:

- PEPE DOES THE JAVA COURSE IN THE MORNING SCHEDULE
- PEPE CARRIES OUT THE ANGULAR COURSE IN THE AFTERNOON
- ELENA TAKES THE JAVA COURSE IN THE AFTERNOON SCHEDULE
- ANA CARRIES OUT THE ANGULAR COURSE IN THE MORNING SCHEDULE
- MARIO DOES THE SPRING COURSE IN THE MORNING SCHEDULE

Solutions:

```
// Create course and student nodes
CREATE
    (java:course {name:'Standard Java Programming', duration:120, price:80}),
    (angular:course {name:'Angular', duration:30, price:110}),
    (spring:course {name:'Spring', duration:80, price:200});
CREATE
    (pepe:student {name:'Pepe', age:20}),
    (elena:student {name:'Elena', age:34}),
    (ana:student {name:'Ana', age:40}),
    (mario:student {name:'Mario', age:19});
```

```
// Create relationships
MATCH
        (pepe:student {name:'Pepe', age:20}),
        (java:course {name:'Standard Java Programming', duration:120, price:80});
CREATE
        (pepe)-[:does {schedule:'Morning'}] -> (java);
MATCH
        (elena: student {name:'Elena'}),
        (java:course {name:'Standard Java Programming'});
CREATE
        (elena)-[:does {schedule:'Afternoon'}] -> (java);
MATCH
        (pepe:student {name:'Pepe'}),
        (angular:course {name:'Angular'})
CREATE
        (pepe)-[:does {schedule:'Afternoon'}] -> (angular);
MATCH
        (ana:student {name:'Ana'}),
        (angular:course {name:'Angular'})
CREATE
        (ana)-[:does {schedule:'Morning'}] -> (angular);
MATCH
        (mario: student {name:'Mario'}),
        (spring:course {name:'Spring'})
CREATE
        (mario)-[:does {schedule:'Tomorrow'}] -> (spring);
```

Once the tree is created, we want to perform certain operations on it:

1. Show the name of the students who take courses:
2. Search for the spring course
3. Show the courses that Pepe takes
4. Show Pepe's data and the number of courses where he has enrolled
5. Show the name of the students who take the angular course
6. And if we want to know the amount?
7. Modify Ana's age from 40 to 25 years. Show the person whose age =25
8. Show all ages of students
9. Show the price of all courses
10. We discharge Mario
11. Obtain the maximum and minimum age and the sum of students
12. Modify the relationship (pepe)-[:perform{shift:"Tomorrow"}]->(java), since it happens to perform it in the afternoon
13. We remove the tree

Solutions:

1) MATCH (student)-[:does] -> (course) RETURN student.name
2) MATCH (course {name: "Spring"}) RETURN course
3) MATCH (Pepe {name: "Pepe"})-[:does]->(course) RETURN course
4) MATCH (n {NAME:"Pepe"})-[r]->()
   RETURN labels(n), n.AGE, count(*);
5) MATCH (student)-[:does]-> (course{name:"Angular"}) RETURN student.name
6) MATCH (student)-[:does]-> (course{name:"Angular"}) RETURN count(student)
7) MATCH (n{name:'Ana'}) SET n.age=25 RETURN n.name,n.age
8) MATCH (n:student) RETURN collect(n.age)
9) MATCH (n:course) RETURN collect(n.price)
10) MATCH (n:student {name: 'Mario'}) detach DELETE n
11) MATCH (n:student)
    RETURN min(n.age), max(n.age), sum(n.age);
12) MATCH (pepe)-[old:does {schedule:'Morning'}]->(java)
    CREATE (pepe)-[new: realiza{schedule:'afternoon'}]->(java)
    DELETE old;
13) MATCH (n) DETACH DELETE n