SPRING DATA JPA assignments:

1. Create a program for student / department for spring data JPA, create entities, perform required CRUD operations, sorting, pagination to it, also apply custom queries using spring data JPA. Apply cashing to the code, use all suitable functionalities to execute the program more efficiently.

```java
package relationship.model;

import java.io.Serializable;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonManagedReference;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

@Entity
@Table(name = "department")
@NoArgsConstructor
@AllArgsConstructor
@Data
@ToString
public class Department implements Serializable{

    private static final long serialVersionUID =1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)  // Changed to
IDENTITY for MySQL auto-increment
    private Long departmentId;

    private String location;
    private String name;

    @OneToMany(
        mappedBy = "department",
        cascade = CascadeType.ALL,
        orphanRemoval = true
    )
    @JsonManagedReference
    private List<Student> students;
}
```

```java
package relationship.model;

import java.io.Serializable;

import com.fasterxml.jackson.annotation.JsonBackReference;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

@Entity
@AllArgsConstructor
@NoArgsConstructor
@Data
@ToString
public class Student implements Serializable {
    private static final long serialVersionUID =1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long studentId;

    private String email;
    private int marks;
    private String name;
    private int age;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "department_id")
    @JsonBackReference
    private Department department;
}
```

```java
package relationship.repository;

import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import relationship.model.Department;


@Repository
public interface DepartmentRepository extends JpaRepository<Department, Long>
{
    Department findByName(String deptname);
    Department findByLocation(String location);
    @Query("SELECT d from Department d ORDER by d.name ASC")
    List<Department> findAllByOrderByNameAsc();
    @Query("SELECT d from Department d ORDER by d.name DESC")
    List<Department> findAllByOrderByNameDesc();
}
```

```java
package relationship.repository;

import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
//import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import relationship.model.Student;

@Repository
public interface StudentRepository extends JpaRepository<Student, Long> {

    List<Student> findByName(String name);

//  @Query("SELECT s FROM Student s WHERE s.marks > :marks")
    List<Student> findByMarksGreaterThan(int marks);


//  @Query("SELECT s FROM Student ORDER  by s.age ASC")
    List<Student> findAllByOrderByAgeAsc();

//  @Query("SELECT s FROM Student ORDER  by s.age DESC")
    List<Student> findAllByOrderByAgeDesc();

    List<Student> findByEmail(String email);
}
```

```java
package relationship.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import jakarta.persistence.EntityNotFoundException;
import jakarta.transaction.Transactional;
import relationship.model.Department;
import relationship.model.Student;
import relationship.repository.DepartmentRepository;
import relationship.repository.StudentRepository;
import java.util.List;


@Service
public class StudentService {

    @Autowired
    private StudentRepository  studentRepository ; //student: name,marks,
department, email
    @Autowired
    private DepartmentRepository departmentRepository;

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    @Transactional
    public Student saveStudent(Long DeptId, Student student) {

        Department dep = departmentRepository.findById(DeptId).orElseThrow(()-
>new EntityNotFoundException());
            student.setDepartment(dep);

            return studentRepository.save(student);
        }

    public Student getStudentById(Long id) {
        return studentRepository.findById(id).orElseThrow(() -> new
RuntimeException("Student with ID " + id + " not found"));
    }

    public List<Student> getStudentByName(String name) {
        return studentRepository.findByName(name);  // .orElseThrow(() -> new
RuntimeException("Student with name " + name + " not found"));
    }
    public List<Student> getStudentByEmail(String email) {
        return studentRepository.findByEmail(email); //.orElseThrow(() -> new
RuntimeException("Student with email " + email + " not found"));
    }

    public List<Student> getStudentByMarksGreaterThan(int marks) {
        return studentRepository.findByMarksGreaterThan(marks);
//.orElseThrow(() -> new RuntimeException("Student with marks > " +marks   + "
not found"));
    }

    public void updateStudent(Long id, Student student) {
        Student existingStudent =
studentRepository.findById(id).orElseThrow(EntityNotFoundException::new);

        if (student.getName() != null)
existingStudent.setName(student.getName());
        if (student.getEmail() != null)
existingStudent.setEmail(student.getEmail());
        if (student.getAge() != 0) existingStudent.setAge(student.getAge());
        if (student.getDepartment() != null)
existingStudent.setDepartment(student.getDepartment());

        studentRepository.save(existingStudent);
    }


    public List<Student> sortStudentAgeInAsc(){
        return studentRepository.findAllByOrderByAgeAsc();
    }

    public List<Student> sortStudentAgeInDesc(){
        return studentRepository.findAllByOrderByAgeDesc();
    }

    public String deleteStudentById(Long id) {
        if (studentRepository.existsById(id)) {
        studentRepository.deleteById(id);
        } else {
            return "Student with ID " + id + " not found";
        }
        return "Student with ID " + id + "deleted successfully";
    }
}
```

```java
package relationship;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DepartmentStudentJpaRelationshipsApplication {

    public static void main(String[] args) {
        SpringApplication.run(DepartmentStudentJpaRelationshipsApplication.cla
ss, args);
    }

}
```

```properties
spring.application.name=Department-Student-JPA-Relationships
server.port=2002

# DataSource Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/Collegedb
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
# ddl-auto values: update, validate, create, create-drop
```