1. General Spring Boot and JPA Concepts

Q1: What is Spring Boot, and why is it used in these experiments?

A: Framework (simplifies development).

Q2: Explain the role of Spring Data JPA in these programs.

A: Simplifies database access.

Q3: What is an JPA Entity, and how is it defined in these codes?

A: Database-mapped class (@Entity, @Id).

Q4: Describe the purpose of `application.properties` in Spring Boot.

A: Configures app settings.

Q5: What are the key dependencies in the `pom.xml` files, and why are they needed?

A: Starters (web, JPA, MySQL).

Q6: How does Spring Boot handle database initialization in these apps?

A: CommandLineRunner or endpoint.

Q7: What is the role of `@RestController` and REST annotations like `@GetMapping` and `@PostMapping`?

A: Handles HTTP requests (JSON responses).

Q8: Explain `@Autowired` vs. constructor injection in these codes.

A: Dependency injection (field vs. constructor).

Q9: What does `spring.jpa.hibernate.ddl-auto=create-drop` do?

A: Auto-creates/drops tables.

Q10: How do you run these Spring Boot applications?

A: mvn spring-boot:run.

---

2. EXP-1 Specific Questions (Basic Spring Boot Application with Spring Data JPA)

Q11: Describe the overall architecture of EXP-1.

A: Entity-Repository-Controller-Main.

Q12: Why is the primary key in `Student` an `int sno` without `@GeneratedValue`?

A: Manual assignment.

Q13: Explain the `CommandLineRunner` in `StudentApplication`.

A: Auto-inserts data on startup.

Q14: What are the REST endpoints in `StudentController`, and what do they do?

A: POST (save), GET (list).

Q15: How would you test the endpoints in EXP-1? Provide example requests and expected outputs.

A: Postman/curl (JSON list).

Q16: What happens if you run EXP-1 without a MySQL database named 'mca'?

A: Connection error.

Q17: Why is `spring-boot-starter-data-jdbc` included in POM alongside JPA?

A: Redundant (basic JDBC).

Q18: Explain the no-args constructor in `Student`.

A: JPA requirement.

---

3. EXP-2 Specific Questions (Pagination and Sorting in Spring Data JPA)

Q19: What is the main focus of EXP-2, and how does it differ from EXP-1 in data insertion?

A: Pagination/sorting (manual init).

Q20: Describe the `Book` entity and its differences from `Student`.

A: Auto-ID (@GeneratedValue).

Q21: Explain the pagination and sorting logic in `BookController`'s `@GetMapping`.

A: PageRequest (params, Sort).

Q22: What is the output structure of the paginated endpoint in EXP-2?

A: Page<Book> (content, metadata).

Q23: Provide example API calls for EXP-2 and explain the results.

A: /init (add), /books (paginated JSON).

Q24: Why use `Page<Book>` instead of `List<Book>` for the endpoint?

A: Efficient metadata.

Q25: What database is used in EXP-2, and how does it differ from EXP-1?

A: 'new' (port 8843).

Q26: Explain `PageRequest.of(page, size, sort)` in detail.

A: Builds Pageable (LIMIT/ORDER BY).

---

4. Comparative Questions (Between EXP-1 and EXP-2)

Q27: Compare the repositories in both experiments.

A: JpaRepository (basic vs. pageable).

Q28: How do the controllers differ in handling requests?

A: Simple vs. parametrized.

Q29: Discuss similarities and differences in POM files.

A: Similar starters (versions differ).

Q30: Why might EXP-2 be more scalable than EXP-1 for large datasets?

A: Lazy loading.

Q31: If you combine features from both, how would you add pagination to EXP-1?

A: Add Pageable to GET.

Q32: What common issues could arise in both apps, and how to fix?

A: DB/port errors (verify config).

Q33: Explain how both apps handle JSON serialization.

A: Jackson (auto).

Q34: Compare data insertion outputs and when they occur.

A: Auto vs. manual (console/response).