Hands-On: Using HQL and JPQL with Spring Data JPA in Java

1. Create a new Spring Boot project with Spring Data JPA set up.
2. Define two JPA entities: `**Book**` and `**Author**`, with a one-to-many relationship (one `**Author**` can have many `**Book**s`).
3. Use the `**@Query**` annotation to define the following custom queries in your `**BookRepository**` interface:
    - Find all books published in a specific year, ordered by their title.
    - Find all authors who have written at least one book published in a specific year, ordered by their last name.
    - Find the average number of pages for all books written by a specific author.
4. Use the HQL fetch keyword or JPQL JOIN FETCH syntax to optimize the following queries:
    - Find all books and their authors.
    - Find all authors and their books, but only fetch the books published in a specific year.
5. Use aggregate functions in HQL to define the following custom queries in your `**BookRepository**` interface:
    - Find the number of books published by a specific author.
    - Find the year with the most books published.
    - Find the average number of pages for all books published in a specific year.
6. Test your queries by calling them from a service or controller class, and verify that the results are correct.

Here's an example implementation for the first query:

```
public interface BookRepository extends JpaRepository<Book, Long> {
   @Query("SELECT b FROM Book b WHERE b.year = :year ORDER BY b.title")
   List<Book> findByYear(@Param("year") int year);
}
```

This interface defines a method that will return a list of books published in a specific year, ordered by their title, using the custom HQL query defined in the @Query annotation.

You can implement the other queries similarly, using HQL or JPQL syntax as appropriate.