**3. Spring Data Jpa Hands On**

**Demonstrate writing Hibernate Query Language and Native Query**

| **Feature** | **HQL (Hibernate)** | **JPQL (JPA)** |
| --- | --- | --- |
| **Defined By** | Hibernate | JPA specification |
| **Vendor Specific** | Yes (Hibernate only) | No (works across all JPA providers) |
| **Entity-Based** | Yes | Yes |
| **SQL Syntax** | Abstracted — does not query tables directly | Abstracted — uses entity names and fields |
| **Fetch Joins** | Supported | Supported |
| **Compatibility** | Tied to Hibernate version | Portable across vendors like EclipseLink, etc. |

**. Using @Query in Spring Data JPA**

Spring Data JPA allows you to use HQL/JPQL or native SQL directly via the @Query annotation.

**HQL/JPQL Example**

@Query("SELECT e FROM Employee e WHERE e.salary > :minSalary")

List<Employee> findEmployeesWithHighSalary(@Param("minSalary") double salary);

**Explanation:**

* "Employee" is the name of the entity (not the table).
* You must use **field names**, not column names.

**HQL with fetch Keyword (for eager loading)**

@Query("SELECT e FROM Employee e JOIN FETCH e.department")

List<Employee> findAllEmployeesWithDepartment();

**Purpose:** Prevents the **LazyInitializationException** by eagerly loading department.

**HQL with Aggregate Functions**

@Query("SELECT AVG(e.salary) FROM Employee e")

Double findAverageSalary();

@Query("SELECT COUNT(e) FROM Employee e WHERE e.permanent = true")

Long countPermanentEmployees();

**Native SQL Queries**

Use nativeQuery = true when writing SQL instead of JPQL.

@Query(value = "SELECT \* FROM employee WHERE em\_salary > :salary", nativeQuery = true)

List<Employee> findHighEarners(@Param("salary") double salary);

**Note:**

* SQL column names and table names must match the actual database schema.
* You can also use nativeQuery = true for more complex or vendor-specific SQL.

**Comparison Table**

| **Query Type** | **Uses Entities?** | **Uses SQL Table Names?** | **Portable?** | **Annotation** |
| --- | --- | --- | --- | --- |
| JPQL / HQL | Yes | No | Yes | @Query |
| Native SQL | No | Yes | No | @Query(nativeQuery = true) |

**Example Repository**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT e FROM Employee e WHERE e.salary > :minSalary")

List<Employee> getEmployeesWithSalaryAbove(@Param("minSalary") double salary);

@Query("SELECT e FROM Employee e JOIN FETCH e.department")

List<Employee> getAllEmployeesWithDepartment();

@Query("SELECT COUNT(e) FROM Employee e WHERE e.permanent = true")

long countPermanentEmployees();

@Query(value = "SELECT \* FROM employee WHERE em\_salary > :minSalary", nativeQuery = true)

List<Employee> getEmployeesUsingNativeQuery(@Param("minSalary") double salary);

}