**Spring Data JPA-Additional-Hands-on**

**Hands-on 1: Introduction to HQL and JPQL**

JPQL (Java Persistence Query Language) and HQL (Hibernate Query Language) are object-oriented query languages used for data access in JPA and Hibernate respectively. JPQL is a subset of HQL.

## Example JPQL Query

SELECT e FROM Employee e WHERE e.permanent = true

JPQL is portable and does not rely on underlying database features. HQL allows more flexibility including INSERT operations.

# Hands-on 2: Get All Permanent Employees Using HQL

## EmployeeRepository.java

@Query(value="SELECT e FROM Employee e left join fetch e.department d left join fetch e.skillList WHERE e.permanent = 1")  
List<Employee> getAllPermanentEmployees();

## Output:

Start  
Permanent Employees:[Employee(id=1, name=Anusha,...), Employee(id=2, name=Priya,...)]  
Skills:[Java, Spring Boot]  
End

# Hands-on 3: Fetch Quiz Attempt Details Using HQL

This hands-on involves creating entity mappings for a quiz attempt system and writing a JOIN query to fetch attempt data, including selected answers, correct answers, and scores.

## AttemptRepository.java

@Query("SELECT a FROM Attempt a JOIN FETCH a.user u JOIN FETCH a.attemptQuestions aq JOIN FETCH aq.question q JOIN FETCH aq.attemptOptions ao JOIN FETCH ao.option o WHERE u.id = :userId AND a.id = :attemptId")  
Attempt getAttempt(@Param("userId") int userId, @Param("attemptId") int attemptId);

## Output:

What is HTML?  
1) .xhtm 0.0 false  
2) .ht 0.0 false  
3) .html 1.0 true  
4) .htmx 0.0 false

# Hands-on 4: Get Average Salary Using HQL

## EmployeeRepository.java

@Query(value="SELECT AVG(e.salary) FROM Employee e where e.department.id = :id")  
double getAverageSalary(@Param("id") int id);

## Output:

Average salary for Department 3 is: 78333.33

# Hands-on 5: Get All Employees Using Native Query

## EmployeeRepository.java

@Query(value="SELECT \* FROM employee", nativeQuery = true)  
List<Employee> getAllEmployeesNative();

## Output:

Employee List: [Employee(id=1,...), Employee(id=2,...)]

# Hands-on 6: Criteria Query Example

Criteria Query is used to dynamically build WHERE clauses depending on selected filters in real-time applications.

## Sample Criteria Query Usage

CriteriaBuilder cb = em.getCriteriaBuilder();  
CriteriaQuery<Product> cq = cb.createQuery(Product.class);  
Root<Product> product = cq.from(Product.class);  
List<Predicate> predicates = new ArrayList<>();  
if (ramSelected) predicates.add(cb.equal(product.get("ram"), "16GB"));  
cq.where(predicates.toArray(new Predicate[0]));

# Comparison Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPQL/HQL | Native Query | Criteria Query |
| Syntax Style | Object-oriented | SQL-like | Programmatic |
| Portability | High | Low | High |
| Dynamic Building | No | No | Yes |
| Use Case | General queries | Performance/complex logic | Dynamic filters |