Spring Data JPA Hands-On Solutions (Part 3)

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

@Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = 1")  
List<Employee> getAllPermanentEmployees();

This HQL joins and fetches department and skill list to avoid LazyInitializationException.

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

Create HQL that joins User → Attempt → AttemptQuestion → Question → AttemptOption → Option.

Include fetch wherever needed to avoid LazyInitializationException.

Use nested loops to print the structure in the required output format.

# Hands-On 4: Get Average Salary Using HQL

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

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

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

# Hands-On 6: Criteria Query Example

Scenario: An e-commerce product filter with multiple dynamic conditions like price, brand, RAM, etc.

Use `CriteriaBuilder`, `CriteriaQuery`, `Root`, and `Predicate` to dynamically build query filters.

// Sample Skeleton  
CriteriaBuilder cb = em.getCriteriaBuilder();  
CriteriaQuery<Product> cq = cb.createQuery(Product.class);  
Root<Product> root = cq.from(Product.class);  
List<Predicate> predicates = new ArrayList<>();  
  
if (brand != null) {  
 predicates.add(cb.equal(root.get("brand"), brand));  
}  
if (price != null) {  
 predicates.add(cb.lessThanOrEqualTo(root.get("price"), price));  
}  
  
cq.where(cb.and(predicates.toArray(new Predicate[0])));  
TypedQuery<Product> query = em.createQuery(cq);  
List<Product> result = query.getResultList();