# 3.Get all employees using Native Query: Update EmployeeRepository:

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();

}

# Update EmployeeService:

@Service

public class EmployeeService { @Autowired

private EmployeeRepository employeeRepository; public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

}

# Test in OrmLearnApplication.java:

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner

{

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) { SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception { List<Employee> employees =

employeeService.getAllEmployeesNative();

employees.forEach(System.out::println);

}

}

# Criteria Query with dynamic filters:

public List<Product> searchProducts(String ram, String cpu, String os) {

CriteriaBuilder cb = entityManager.getCriteriaBuilder(); CriteriaQuery<Product> cq = cb.createQuery(Product.class); Root<Product> root = cq.from(Product.class); List<Predicate> predicates = new ArrayList<>();

if (ram != null) { predicates.add(cb.equal(root.get("ramSize"), ram));

}

if (cpu != null) { predicates.add(cb.equal(root.get("cpu"), cpu));

}

if (os != null) {

predicates.add(cb.equal(root.get("operatingSystem"), os));

}

cq.where(cb.and(predicates.toArray(new Predicate[0]))); TypedQuery<Product> query = entityManager.createQuery(cq); return query.getResultList();

}

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **salary** |
| **1** | **John Doe** | **50000** |
| **2** | **Jane Smith** | **60000** |
| **3** | **Mike Johnson** | **55000** |