# Hands-on 4 & 5: Working with HQL and Native Queries

## Hands-on 4: Get Average Salary using HQL

Objective: Compute the average salary of a department using HQL.

Steps to Implement:

1. Define HQL in EmployeeRepository:

@Query(value="SELECT AVG(e.salary) FROM Employee e")  
double getAverageSalary();

2. Include the above method in EmployeeService.

3. Add a test call in OrmLearnApplication.

4. Modify the query to accept a department ID:

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

Notes:

- Observe how department id is referred from "e" (i.e., e.department.id).

- Colon (:) defines a named parameter in the query.

- @Param annotation binds method input to query parameter.

- AVG() is an aggregate function; others like COUNT(), SUM(), MIN(), MAX() can also be used.

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

Objective: Use a native SQL query to fetch all employees.

About Native Queries:

- Native queries directly use SQL syntax.

- Prefer HQL over native queries for better portability.

Steps to Implement:

1. Define the native query in EmployeeRepository:

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

2. Add a corresponding method in EmployeeService.

3. Test the method from OrmLearnApplication.