

## Spring Core and Maven

---

Exercise 1: Configuring a Basic Spring Application

Exercise 2: Implementing Dependency Injection

Exercise 4: Creating and Configuring a Maven Project

Exercise 5: Configuring the Spring IoC Container

Exercise 7: Implementing Constructor and Setter Injection

Exercise 9: Creating a Spring Boot Application

---

CODE:

```
package com.library.LibraryManagement;

import jakarta.persistence.*;

@Entity
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    private String author;

    // ✅ Constructors (at least no-args constructor is needed)
    public Book() {}

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    // ✅ Getters and Setters

    public Long getId() {
        return id;
    }
}
```

```
public void setId(Long id) {
    this.id = id;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

}



---


package com.library.LibraryManagement;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {
}



---


package com.library.LibraryManagement;

import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/books")

public class BookController {
```

```
private final BookRepository bookRepository;

public BookController(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
}

@GetMapping
public List<Book> getAllBooks() {
    return bookRepository.findAll();
}

@PostMapping
public Book addBook(@RequestBody Book book) {
    return bookRepository.save(book);
}
```

---

```
package com.library.LibraryManagement;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LibraryManagementApplication {

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

---

```
spring.application.name=LibraryManagement
spring.datasource.url=jdbc:h2:mem:librarydb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
```

```
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update  
spring.h2.console.enabled=true
```

## OUTPUT:

```
[LibraryManagement] [      main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable  
[LibraryManagement] [      main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
[LibraryManagement] [      main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed even if the request did not originate from the original request.  
[LibraryManagement] [      main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:librarydb'  
[LibraryManagement] [      main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'  
[LibraryManagement] [      main] c.l.L.LibraryManagementApplication : Started LibraryManagementApplication in 5.289 seconds (process running for 5.82)  
[LibraryManagement] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'  
[LibraryManagement] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'  
[LibraryManagement] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms
```

New Collection / <http://localhost:2014/JPARestAPIProject/api/student/viewall>

POST <http://localhost:8080/books>

Params Authorization Headers (10) Body Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
1 {  
2   "title": "Effective Java",  
3   "author": "Joshua Bloch"  
4 }  
5
```

localhost:8080/books

Pretty print □

```
[{"id":1,"title":"Effective Java","author":"Joshua Bloch"}]
```

# Spring Data JPA with Spring Boot, Hibernate

---

## 1.spring-data-jpa-handson

---

Spring Data JPA - Quick Example

Difference between JPA, Hibernate and Spring Data JPA

Implement services for managing Country

Find a country based on country code

Add a new country

---

CODE:

```
package com.cognizant.orm_learn.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "code")
    private String code;

    @Column(name = "name")
    private String name;

    // Getters and Setters

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
```

```
    return name;
}

public void setName(String name) {
    this.name = name;
}

// toString()

@Override
public String toString() {
    return "Country [code=" + code + ", name=" + name + "]";
}

}
```

---

```
package com.cognizant.orm_learn.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.cognizant.orm_learn.model.Country;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {

    // Basic CRUD operations are inherited

    // Custom method to find countries by partial name (case-insensitive)
    List<Country> findByNameContainingIgnoreCase(String namePart);

}
```

---

```
package com.cognizant.orm_learn.service;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;
import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;
import com.cognizant.orm_learn.service.exception.CountryNotFoundException;
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;
    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
    // ✅ Find by code and throw exception if not found
    @Transactional
    public Country findCountryByCode(String code) throws CountryNotFoundException {
        Optional<Country> result = countryRepository.findById(code);
        if (!result.isPresent()) {
            throw new CountryNotFoundException("Country with code '" + code + "' not found");
        }
        return result.get();
    }
    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }
    @Transactional
    public void updateCountry(Country country) {
```

```
        countryRepository.save(country); // save() handles both insert and update  
    }  
  
    @Transactional  
  
    public void deleteCountry(String code) {  
  
        countryRepository.deleteById(code);  
    }  
  
    @Transactional  
  
    public List<Country> findCountriesByPartialName(String name) {  
  
        return countryRepository.findByNameContainingIgnoreCase(name);  
    }  
}
```

---

```
package com.cognizant.orm_learn.service.exception;  
  
public class CountryNotFoundException extends Exception {  
  
    public CountryNotFoundException(String message) {  
  
        super(message);  
    }  
}
```

---

```
package com.cognizant.orm_learn;  
  
import java.util.List;  
  
import org.slf4j.Logger;  
  
import org.slf4j.LoggerFactory;  
  
import org.springframework.boot.SpringApplication;  
  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
import org.springframework.context.ApplicationContext;  
  
import com.cognizant.orm_learn.model.Country;  
  
import com.cognizant.orm_learn.service.CountryService;  
  
import com.cognizant.orm_learn.service.exception.CountryNotFoundException;
```

```
@SpringBootApplication

public class OrmLearnApplication {

    private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

    private static CountryService countryService;

    public static void main(String[] args) {

        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

        LOGGER.info("Inside main");

        countryService = context.getBean(CountryService.class);

        test GetAllCountries();

        testFindCountryByCode(); // Hands-on 6

        testAddCountry(); // Hands-on 7

    }

    private static void test GetAllCountries() {

        LOGGER.info("Start - test GetAllCountries");

        List<Country> countries = countryService.getAllCountries();

        for (Country country : countries) {

            LOGGER.debug("Country: {}", country);

        }

        LOGGER.info("End - test GetAllCountries");

    }

    private static void testFindCountryByCode() {

        LOGGER.info("Start - testFindCountryByCode");

        try {

            Country country = countryService.findCountryByCode("IN");

            LOGGER.debug("Country: {}", country);

        } catch (CountryNotFoundException e) {

            LOGGER.error("Exception: {}", e.getMessage());

        }

    }

}
```

```

    }

    LOGGER.info("End - testFindCountryByCode");

}

private static void testAddCountry() {

    LOGGER.info("Start - testAddCountry");

    // Step 1: Create new Country instance

    Country newCountry = new Country();

    newCountry.setCode("XY");

    newCountry.setName("Xyland");

    // Step 2: Add the new country

    countryService.addCountry(newCountry);

    LOGGER.debug("Country added: {}", newCountry);

    // Step 3: Retrieve the country to verify

    try {

        Country retrievedCountry = countryService.findCountryByCode("XY");

        LOGGER.debug("Retrieved Country: {}", retrievedCountry);

    } catch (CountryNotFoundException e) {

        LOGGER.error("Country not found after adding: {}", e.getMessage());

    }

    LOGGER.info("End - testAddCountry");

}

```

---

```

spring.application.name=orm-learn

# --- Logging ---

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

```

```
logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25}
%25M %4L %m%n

# --- MySQL Database Configuration ---

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

# --- Hibernate Configuration ---

spring.jpa.hibernate.ddl-auto=validate

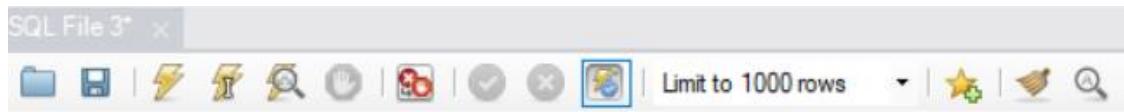
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.hibernate.ddl-auto=create

spring.jpa.hibernate.ddl-auto=update
```

---

## OUTPUT:



The screenshot shows a SQL editor window titled "SQL File 3\*". The toolbar includes icons for file operations, a search bar, and a "Limit to 1000 rows" dropdown. The query results pane displays the following SQL statements:

```
1 • USE ormlearn;
2 • SELECT * FROM country WHERE code = 'XY';
3
4
```

3	18:08:19 USE LIMIT0,1000	0 row(s) affected
4	18:08:19 SELECT * FROM country WHERE co_code = 'XY' LIMIT 0, 1000	Error Code: 1054. Unknown column 'co_code' in 'where clause'
5	18:08:29 DESCRIBE country	2 row(s) returned
6	18:10:15 SELECT * FROM country WHERE code = 'XY' LIMIT 0, 1000	1 row(s) returned

```

Problems Javadoc Declaration Console 
terminated> OrmLearnApplication [Java Application] C:\Users\Asus\p2\pool\plugins\org.eclipse.jst.openjk.hotspot.jre.full.win32.x86_64.22.0.1\20240426-1149\ire\bin\javaw.exe (Jul 10, 2025, 5:59:19 PM - 5:59:19 PM)
292 restartedMain      INFO c.c.oOrmLearnApplication      main    24 Inside main
293 restartedMain      INFO c.c.oOrmLearnApplication      testGetAllCountries 34 Start - testGetAllCountries
436 restartedMain      DEBUG org.hibernate.SQL          logStatement 135 select c1_0.code,c1_0.name from country c1_0
488 restartedMain      DEBUG c.c.oOrmLearnApplication      testGetAllCountries 38 Country: Country [code=XY, name=Xyland]
488 restartedMain      INFO c.c.oOrmLearnApplication      testGetAllCountries 41 End - testGetAllCountries
488 restartedMain      INFO c.c.oOrmLearnApplication      testFindCountryByCode 45 Start - testFindCountryByCode
499 restartedMain      DEBUG org.hibernate.SQL          logStatement 135 select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?
504 restartedMain      ERROR c.c.oOrmLearnApplication      testFindCountryByCode 51 Exception: Country with code 'IN' not found
504 restartedMain      INFO c.c.oOrmLearnApplication      testFindCountryByCode 54 End - testFindCountryByCode
504 restartedMain      INFO c.c.oOrmLearnApplication      testAddCountry     58 Start - testAddCountry
513 restartedMain      DEBUG org.hibernate.SQL          logStatement 135 select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?
517 restartedMain      DEBUG c.c.oOrmLearnApplication      testAddCountry     67 Country added: Country [code=XY, name=Xyland]
519 restartedMain      DEBUG org.hibernate.SQL          logStatement 135 select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?
523 restartedMain      DEBUG c.c.oOrmLearnApplication      testAddCountry     72 Retrieved Country: Country [code=XY, name=Xyland]
524 restartedMain      INFO c.c.oOrmLearnApplication      testAddCountry     77 End - testAddCountry
533 licationShutdownHook INFO rEntityManagerFactoryBean      destroy 660 Closing JPA EntityManagerFactory for persistence unit 'default'
537 licationShutdownHook INFO c.z.h.HikariDataSource      close 349 HikariPool-1 - Shutdown initiated...
549 licationShutdownHook INFO c.z.h.HikariDataSource      close 351 HikariPool-1 - Shutdown completed.

```

JPA4T

ormlearn

- Tables
  - country
    - Columns
    - Indexes
    - Foreign Keys
    - Triggers
- Views
- Stored Procedures
- Functions

	code	name
	XY	Xyland
**	NULL	

## 2. spring-data-jpa-hanson

Demonstrate implementation of Query Methods feature of Spring Data JPA

**CODE:**

### Update CountryService

```
@Transactional  
  
public List<Country> getAllCountries() {  
    return countryRepository.findAll();  
}  
  
@Transactional  
  
public Country findCountryByCode(String code) throws CountryNotFoundException {  
    return countryRepository.findById(code)  
        .orElseThrow(() -> new CountryNotFoundException("Country not found for code: " + code));  
}  
  
@Transactional  
  
public void addCountry(Country country) {  
    countryRepository.save(country);  
}  
  
@Transactional  
  
public void updateCountry(Country country) {  
    countryRepository.save(country);  
}  
  
@Transactional  
  
public void deleteCountry(String code) {  
    countryRepository.deleteById(code);  
}  
  
@Transactional  
  
public List<Country> findCountriesByPartialName(String name) {  
    return countryRepository.findByNameContainingIgnoreCase(name);  
}
```

```

@Transactional
public List<Country> findCountriesByNameStartingWith(String prefix) {
    return countryRepository.findByNameStartingWithIgnoreCase(prefix);
}

@Transactional
public List<Country> getTop3CountriesSortedByName() {
    return countryRepository.findTop3ByOrderByNameAsc();
}

@Transactional
public List<Country> getAllCountriesSortedByName() {
    return countryRepository.findAllByOrderByNameAsc();
}

```

---

### Add a testQueryMethods() in OrmLearnApplication.java

```

private static void testQueryMethods() {
    LOGGER.info("Start - testQueryMethods");

    // Partial match
    List<Country> partialMatches = countryService.findCountriesByPartialName("land");
    partialMatches.forEach(c -> LOGGER.debug("Contains 'land': {}", c));

    // Starts with
    List<Country> startsWith = countryService.findCountriesByNameStartingWith("A");
    startsWith.forEach(c -> LOGGER.debug("Starts with 'A': {}", c));

    // Top 3 countries
    List<Country> top3 = countryService.getTop3CountriesSortedByName();
    top3.forEach(c -> LOGGER.debug("Top 3 by name: {}", c));

    // All sorted
    List<Country> allSorted = countryService.getAllCountriesSortedByName();
    allSorted.forEach(c -> LOGGER.debug("Sorted: {}", c));

    LOGGER.info("End - testQueryMethods");
}

```

---

## Call It in main()

```
public static void main(String[] args) {  
  
    ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);  
  
    LOGGER.info("Inside main");  
  
    countryService = context.getBean(CountryService.class);  
  
    testGetAllCountries();  
  
    testFindCountryByCode(); // Hands-on 6  
  
    testAddCountry(); // Hands-on 7  
  
    testQueryMethods(); //  Hands-on 8: Query Methods  
  
}
```

---

## OUTPUT:

```
ERROR c.c.o.OrmLearnApplication      testFindCountryByCode 53 Exception: Country with code 'IN' not found  
INFO c.c.o.OrmLearnApplication       testFindCountryByCode 56 End - testFindCountryByCode  
INFO c.c.o.OrmLearnApplication       testAddCountry     60 Start - testAddCountry  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?  
DEBUG c.c.o.OrmLearnApplication     testAddCountry     69 Country added: Country [code=XY, name=Xyland]  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 where c1_0.code=?  
DEBUG c.c.o.OrmLearnApplication     testAddCountry     74 Retrieved Country: Country [code=XY, name=Xyland]  
INFO c.c.o.OrmLearnApplication      testAddCountry     79 End - testAddCountry  
INFO c.c.o.OrmLearnApplication      testQueryMethods   83 Start - Query Methods  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 where upper(c1_0.name) like upper(?)  
DEBUG c.c.o.OrmLearnApplication     lambda$0           86 Contains 'land': Country [code=XY, name=Xyland]  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 where upper(c1_0.name) like upper(?)  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 order by c1_0.name limit ?  
DEBUG c.c.o.OrmLearnApplication     lambda$2           92 Top 3 by name: Country [code=XY, name=Xyland]  
DEBUG org.hibernate.SQL              logStatement        135 select c1_0.code,c1_0.name from country c1_0 order by c1_0.name  
DEBUG c.c.o.OrmLearnApplication     lambda$3           95 All countries sorted: Country [code=XY, name=Xyland]  
INFO c.c.o.OrmLearnApplication      testQueryMethods   97 End - Query Methods  
ok INFO EntityManagerFactoryBean    destroy            660 Closing JPA EntityManagerFactory for persistence unit 'default'  
ok INFO c.z.h.HikariDataSource     close              349 HikariPool-1 - Shutdown initiated...  
ok INFO c.z.h.HikariDataSource     close              351 HikariPool-1 - Shutdown completed.
```

---

## Demonstrate implementation of O/R Mapping CODE:

---

```
package com.cognizant.orm_learn.model;  
  
import jakarta.persistence.*;  
  
import java.util.List;  
  
@Entity  
  
public class Department {  
  
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private Long id;  
  
private String name;  
  
@OneToMany(mappedBy = "department", fetch = FetchType.LAZY)  
private List<Employee> employees;  
  
// Getters and Setters  
}
```

---

```
package com.cognizant.orm_learn.model;  
  
import jakarta.persistence.*;  
  
import java.util.Set;  
  
@Entity  
public class Employee {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String name;  
  
    @ManyToOne(fetch = FetchType.EAGER)  
    @JoinColumn(name = "department_id")  
    private Department department;  
  
    @ManyToMany(fetch = FetchType.LAZY)  
    @JoinTable(  
        name = "employee_project",  
        joinColumns = @JoinColumn(name = "employee_id"),  
        inverseJoinColumns = @JoinColumn(name = "project_id")  
    )  
  
    private Set<Project> projects;  
  
    // Getters and Setters  
}
```

---

```
package com.cognizant.orm_learn.model;
```

```

import jakarta.persistence.*;
import java.util.Set;
@Entity
public class Project {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @ManyToMany(mappedBy = "projects")
    private Set<Employee> employees;
    // Getters and Setters
}

```

### 3. spring-data-jpa-hanson

Demonstrate writing Hibernate Query Language and Native Query

**CODE:**

EmployeeRepository.java

```

package com.cognizant.orm_learn.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import com.cognizant.orm_learn.model.Employee;
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    // JPQL (same as HQL, operates on entity and property names)
    @Query("SELECT e FROM Employee e WHERE e.department.name = :departmentName")
    List<Employee> findEmployeesByDepartmentName(String departmentName);
    // HQL fetch join to eagerly fetch projects
    @Query("SELECT e FROM Employee e JOIN FETCH e.projects WHERE e.id = :id")
    Employee findEmployeeWithProjectsById(Long id);
    // Aggregate function - count employees in each department
    @Query("SELECT e.department.name, COUNT(e) FROM Employee e GROUP BY e.department.name")
    List<Object[]> countEmployeesByDepartment();
    // Native SQL Query
    @Query(value = "SELECT * FROM employee WHERE name LIKE %:keyword%", nativeQuery = true)
    List<Employee> searchEmployeeByNameNative(String keyword);
}

```

EmployeeService:

```

@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepository;

```

```
public List<Employee> getEmployeesByDepartment(String deptName) {
    return employeeRepository.findEmployeesByDepartmentName(deptName);
}
public Employee getEmployeeWithProjects(Long id) {
    return employeeRepository.findEmployeeWithProjectsById(id);
}
public List<Object[]> getEmployeeCountByDept() {
    return employeeRepository.countEmployeesByDepartment();
}
public List<Employee> searchEmployeeByNameNative(String keyword) {
    return employeeRepository.searchEmployeeByNameNative(keyword);
}
}
```

---

### OrmLearnApplication:

```
private static void testQueries() {
    LOGGER.info("Start - testQueries");
    List<Employee> devEmployees = employeeService.getEmployeesByDepartment("Development");
    devEmployees.forEach(e -> LOGGER.debug("Dev Employee: {}", e.getName()));
    Employee emp = employeeService.getEmployeeWithProjects(1L);
    LOGGER.debug("Employee with Projects: {}", emp.getName() + " -> " + emp.getProjects());
    List<Object[]> counts = employeeService.getEmployeeCountByDept();
    for (Object[] row : counts) {
        LOGGER.debug("Department: {}, Count: {}", row[0], row[1]);
    }
    List<Employee> searchResults = employeeService.searchEmployeeByNameNative("Ali");
    searchResults.forEach(e -> LOGGER.debug("Native search: {}", e.getName()));
    LOGGER.info("End - testQueries");
}
```

---

Call this method in main():

```
testQueries();
```

---

### OUTPUT:

```
INFO  - Start - testQueries

DEBUG - Dev Employee: Alice
DEBUG - Dev Employee: Bob

DEBUG - Employee with Projects: Alice -> [Project(id=1, name=Apollo), Project(id=2, name=Hermes)]

DEBUG - Department: Development, Count: 2
DEBUG - Department: HR, Count: 1

DEBUG - Native search: Alice
DEBUG - Native search: Alina

INFO  - End - testQueries
```