**spring-data-jpa-handson**

**Spring Data JPA - Quick Example**

CREATE SCHEMA ormlearn;

USE ormlearn;

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

[20-07-2025 14:25] Sree Charitha Perumalla: insert into country (co\_code, co\_name) values ("AF", "Afghanistan");

insert into country (co\_code, co\_name) values ("AL", "Albania");

insert into country (co\_code, co\_name) values ("DZ", "Algeria");

insert into country (co\_code, co\_name) values ("AS", "American Samoa");

[20-07-2025 14:28] Sree Charitha Perumalla: insert into country (code, name) values ("AF", "Afghanistan");

insert into country (code, name) values ("AL", "Albania");

insert into country (code,name) values ("DZ", "Algeria");

insert into country (code, name) values ("AS", "American Samoa");

[20-07-2025 14:30] Sree Charitha Perumalla: INSERT INTO country (code, name) VALUES ("AD", "Andorra");

INSERT INTO country (code, name) VALUES ("AO", "Angola");

INSERT INTO country (code, name) VALUES ("AI", "Anguilla");

INSERT INTO country (code, name) VALUES ("AQ", "Antarctica");

INSERT INTO country (code, name) VALUES ("AG", "Antigua and Barbuda");

INSERT INTO country (code, name) VALUES ("AR", "Argentina");

INSERT INTO country (code, name) VALUES ("AM", "Armenia");

INSERT INTO country (code, name) VALUES ("AW", "Aruba");

INSERT INTO country (code, name) VALUES ("AU", "Australia");

INSERT INTO country (code, name) VALUES ("AT", "Austria");

INSERT INTO country (code, name) VALUES ("AZ", "Azerbaijan");

INSERT INTO country (code, name) VALUES ("BS", "Bahamas");

INSERT INTO country (code, name) VALUES ("BH", "Bahrain");

INSERT INTO country (code, name) VALUES ("BD", "Bangladesh");

INSERT INTO country (code, name) VALUES ("BB", "Barbados");

INSERT INTO country (code, name) VALUES ("BY", "Belarus");

INSERT INTO country (code, name) VALUES ("BE", "Belgium");

INSERT INTO country (code, name) VALUES ("BZ", "Belize");

INSERT INTO country (code, name) VALUES ("BJ", "Benin");

INSERT INTO country (code, name) VALUES ("BM", "Bermuda");

INSERT INTO country (code, name) VALUES ("BT", "Bhutan");

INSERT INTO country (code, name) VALUES ("BO", "Bolivia, Plurinational State of");

INSERT INTO country (code, name) VALUES ("BQ", "Bonaire, Sint Eustatius and Saba");

INSERT INTO country (code, name) VALUES ("BA", "Bosnia and Herzegovina");

INSERT INTO country (code, name) VALUES ("BW", "Botswana");

INSERT INTO country (code, name) VALUES ("BV", "Bouvet Island");

INSERT INTO country (code, name) VALUES ("BR", "Brazil");

INSERT INTO country (code, name) VALUES ("IO", "British Indian Ocean Territory");

INSERT INTO country (code, name) VALUES ("BN", "Brunei Darussalam");

INSERT INTO country (code, name) VALUES ("BG", "Bulgaria");

INSERT INTO country (code, name) VALUES ("BF", "Burkina Faso");

INSERT INTO country (code, name) VALUES ("BI", "Burundi");

INSERT INTO country (code, name) VALUES ("KH", "Cambodia");

INSERT INTO country (code, name) VALUES ("CM", "Cameroon");

INSERT INTO country (code, name) VALUES ("CA", "Canada");

INSERT INTO country (code, name) VALUES ("CV", "Cape Verde");

INSERT INTO country (code, name) VALUES ("KY", "Cayman Islands");

INSERT INTO country (code, name) VALUES ("CF", "Central African Republic");

INSERT INTO country (code, name) VALUES ("TD", "Chad");

INSERT INTO country (code, name) VALUES ("CL", "Chile");

INSERT INTO country (code, name) VALUES ("CN", "China");

INSERT INTO country (code, name) VALUES ("CX", "Christmas Island");

INSERT INTO country (code, name) VALUES ("CC", "Cocos (Keeling) Islands");

INSERT INTO country (code, name) VALUES ("CO", "Colombia");

INSERT INTO country (code, name) VALUES ("KM", "Comoros");

INSERT INTO country (code, name) VALUES ("CG", "Congo");

INSERT INTO country (code, name) VALUES ("CD", "Congo, the Democratic Republic of the");

INSERT INTO country (code, name) VALUES ("CK", "Cook Islands");

INSERT INTO country (code, name) VALUES ("CR", "Costa Rica");

INSERT INTO country (code, name) VALUES ("HR", "Croatia");

INSERT INTO country (code, name) VALUES ("CU", "Cuba");

INSERT INTO country (code, name) VALUES ("CW", "Curaçao");

INSERT INTO country (code, name) VALUES ("CY", "Cyprus");

INSERT INTO country (code, name) VALUES ("CZ", "Czech Republic");

INSERT INTO country (code, name) VALUES ("CI", "Côte d'Ivoire");

INSERT INTO country (code, name) VALUES ("DK", "Denmark");

INSERT INTO country (code, name) VALUES ("DJ", "Djibouti");

INSERT INTO country (code, name) VALUES ("DM", "Dominica");

INSERT INTO country (code, name) VALUES ("DO", "Dominican Republic");

-- And so on...

[20-07-2025 14:32] Sree Charitha Perumalla: INSERT INTO country (code, name) VALUES ("EC", "Ecuador");

INSERT INTO country (code, name) VALUES ("EG", "Egypt");

INSERT INTO country (code, name) VALUES ("SV", "El Salvador");

INSERT INTO country (code, name) VALUES ("GQ", "Equatorial Guinea");

INSERT INTO country (code, name) VALUES ("ER", "Eritrea");

INSERT INTO country (code, name) VALUES ("EE", "Estonia");

INSERT INTO country (code, name) VALUES ("ET", "Ethiopia");

INSERT INTO country (code, name) VALUES ("FK", "Falkland Islands (Malvinas)");

INSERT INTO country (code, name) VALUES ("FO", "Faroe Islands");

INSERT INTO country (code, name) VALUES ("FJ", "Fiji");

INSERT INTO country (code, name) VALUES ("FI", "Finland");

INSERT INTO country (code, name) VALUES ("FR", "France");

INSERT INTO country (code, name) VALUES ("GF", "French Guiana");

INSERT INTO country (code, name) VALUES ("PF", "French Polynesia");

INSERT INTO country (code, name) VALUES ("TF", "French Southern Territories");

INSERT INTO country (code, name) VALUES ("GA", "Gabon");

INSERT INTO country (code, name) VALUES ("GM", "Gambia");

INSERT INTO country (code, name) VALUES ("GE", "Georgia");

INSERT INTO country (code, name) VALUES ("DE", "Germany");

INSERT INTO country (code, name) VALUES ("GH", "Ghana");

INSERT INTO country (code, name) VALUES ("GI", "Gibraltar");

INSERT INTO country (code, name) VALUES ("GR", "Greece");

INSERT INTO country (code, name) VALUES ("GL", "Greenland");

INSERT INTO country (code, name) VALUES ("GD", "Grenada");

INSERT INTO country (code, name) VALUES ("GP", "Guadeloupe");

INSERT INTO country (code, name) VALUES ("GU", "Guam");

INSERT INTO country (code, name) VALUES ("GT", "Guatemala");

INSERT INTO country (code, name) VALUES ("GG", "Guernsey");

INSERT INTO country (code, name) VALUES ("GN", "Guinea");

INSERT INTO country (code, name) VALUES ("GW", "Guinea-Bissau");

INSERT INTO country (code, name) VALUES ("GY", "Guyana");

INSERT INTO country (code, name) VALUES ("HT", "Haiti");

INSERT INTO country (code, name) VALUES ("HM", "Heard Island and McDonald Islands");

INSERT INTO country (code, name) VALUES ("VA", "Holy See (Vatican City State)");

INSERT INTO country (code, name) VALUES ("HN", "Honduras");

INSERT INTO country (code, name) VALUES ("HK", "Hong Kong");

INSERT INTO country (code, name) VALUES ("HU", "Hungary");

INSERT INTO country (code, name) VALUES ("IS", "Iceland");

INSERT INTO country (code, name) VALUES ("IN", "India");

INSERT INTO country (code, name) VALUES ("ID", "Indonesia");

INSERT INTO country (code, name) VALUES ("IR", "Iran, Islamic Republic of");

INSERT INTO country (code, name) VALUES ("IQ", "Iraq");

INSERT INTO country (code, name) VALUES ("IE", "Ireland");

INSERT INTO country (code, name) VALUES ("IM", "Isle of Man");

INSERT INTO country (code, name) VALUES ("IL", "Israel");

INSERT INTO country (code, name) VALUES ("IT", "Italy");

INSERT INTO country (code, name) VALUES ("JM", "Jamaica");

INSERT INTO country (code, name) VALUES ("JP", "Japan");

INSERT INTO country (code, name) VALUES ("JE", "Jersey");

INSERT INTO country (code, name) VALUES ("JO", "Jordan");

[20-07-2025 14:33] Sree Charitha Perumalla: insert into country (code, name) values ("KZ", "Kazakhstan");

insert into country (code, name) values ("KE", "Kenya");

insert into country (code, name) values ("KI", "Kiribati");

insert into country (code, name) values ("KP", "Democratic People's Republic of Korea");

insert into country (code, name) values ("KR", "Republic of Korea");

insert into country (code, name) values ("KW", "Kuwait");

insert into country (code, name) values ("KG", "Kyrgyzstan");

insert into country (code, name) values ("LA", "Lao People's Democratic Republic");

insert into country (code, name) values ("LV", "Latvia");

insert into country (code, name) values ("LB", "Lebanon");

[20-07-2025 14:34] Sree Charitha Perumalla: insert into country (code, name) values ("LV", "Latvia");

insert into country (code, name) values ("LB", "Lebanon");

insert into country (code, name) values ("LS", "Lesotho");

insert into country (code, name) values ("LR", "Liberia");

insert into country (code, name) values ("LY", "Libya");

insert into country (code, name) values ("LI", "Liechtenstein");

insert into country (code, name) values ("LT", "Lithuania");

insert into country (code, name) values ("LU", "Luxembourg");

insert into country (code, name) values ("MO", "Macao");

insert into country (code, name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (code, name) values ("MG", "Madagascar");

insert into country (code, name) values ("MW", "Malawi");

insert into country (code, name) values ("MY", "Malaysia");

insert into country (code, name) values ("MV", "Maldives");

insert into country (code, name) values ("ML", "Mali");

insert into country (code, name) values ("MT", "Malta");

insert into country (code, name) values ("MH", "Marshall Islands");

insert into country (code, name) values ("MQ", "Martinique");

insert into country (code, name) values ("MR", "Mauritania");

insert into country (code, name) values ("MU", "Mauritius");

insert into country (code, name) values ("YT", "Mayotte");

insert into country (code, name) values ("MX", "Mexico");

insert into country (code, name) values ("FM", "Micronesia, Federated States of");

insert into country (code, name) values ("MD", "Moldova, Republic of");

insert into country (code, name) values ("MC", "Monaco");

insert into country (code, name) values ("MN", "Mongolia");

insert into country (code, name) values ("ME", "Montenegro");

insert into country (code, name) values ("MS", "Montserrat");

insert into country (code, name) values ("MA", "Morocco");

insert into country (code, name) values ("MZ", "Mozambique");

insert into country (code, name) values ("MM", "Myanmar");

insert into country (code, name) values ("NA", "Namibia");

insert into country (code, name) values ("NR", "Nauru");

insert into country (code, name) values ("NP", "Nepal");

insert into country (code, name) values ("NL", "Netherlands");

insert into country (code, name) values ("NC", "New Caledonia");

insert into country (code, name) values ("NZ", "New Zealand");

insert into country (code, name) values ("NI", "Nicaragua");

insert into country (code, name) values ("NE", "Niger");

insert into country (code, name) values ("NG", "Nigeria");

insert into country (code, name) values ("NU", "Niue");

insert into country (code, name) values ("NF", "Norfolk Island");

insert into country (code, name) values ("MP", "Northern Mariana Islands");

insert into country (code, name) values ("NO", "Norway");

insert into country (code, name) values ("OM", "Oman");

insert into country (code, name) values ("PK", "Pakistan");

insert into country (code, name) values ("PW", "Palau");

insert into country (code, name) values ("PS", "Palestine, State of");

insert into country (code, name) values ("PA", "Panama");

insert into country (code, name) values ("PG", "Papua New Guinea");

insert into country (code, name) values ("PY", "Paraguay");

insert into country (code, name) values ("PE", "Peru");

insert into country (code, name) values ("PH", "Philippines");

insert into country (code, name) values ("PN", "Pitcairn");

insert into country (code, name) values ("PL", "Poland");

insert into country (code, name) values ("PT", "Portugal");

insert into country (code, name) values ("PR", "Puerto Rico");

insert into country (code, name) values ("QA", "Qatar");

insert into country (code, name) values ("RO", "Romania");

insert into country (code, name) values ("RU", "Russian Federation");

insert into country (code, name) values ("RW", "Rwanda");

insert into country (code, name) values ("RE", "Réunion");

insert into country (code, name) values ("BL", "Saint Barthélemy");

insert into country (code, name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (code, name) values ("KN", "Saint Kitts and Nevis");

insert into country (code, name) values ("LC", "Saint Lucia");

insert into country (code, name) values ("MF", "Saint Martin (French part)");

insert into country (code, name) values ("PM", "Saint Pierre and Miquelon");

insert into country (code, name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (code, name) values ("WS", "Samoa");

insert into country (code, name) values ("SM", "San Marino");

insert into country (code, name) values ("ST", "Sao Tome and Principe");

insert into country (code, name) values ("SA", "Saudi Arabia");

insert into country (code, name) values ("SN", "Senegal");

insert into country (code, name) values ("RS", "Serbia");

insert into country (code, name) values ("SC", "Seychelles");

insert into country (code, name) values ("SL", "Sierra Leone");

insert into country (code, name) values ("SG", "Singapore");

insert into country (code, name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (code, name) values ("SK", "Slovakia");

insert into country (code, name) values ("SI", "Slovenia");

[20-07-2025 14:34] Sree Charitha Perumalla: insert into country (code, name) values ("SB", "Solomon Islands");

insert into country (code, name) values ("SO", "Somalia");

insert into country (code, name) values ("ZA", "South Africa");

insert into country (code, name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (code, name) values ("SS", "South Sudan");

insert into country (code, name) values ("ES", "Spain");

insert into country (code, name) values ("LK", "Sri Lanka");

insert into country (code, name) values ("SD", "Sudan");

insert into country (code, name) values ("SR", "Suriname");

insert into country (code, name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (code, name) values ("SZ", "Swaziland");

insert into country (code, name) values ("SE", "Sweden");

insert into country (code, name) values ("CH", "Switzerland");

insert into country (code, name) values ("SY", "Syrian Arab Republic");

insert into country (code, name) values ("TW", "Taiwan, Province of China");

insert into country (code, name) values ("TJ", "Tajikistan");

insert into country (code, name) values ("TZ", "Tanzania, United Republic of");

insert into country (code, name) values ("TH", "Thailand");

insert into country (code, name) values ("TL", "Timor-Leste");

insert into country (code, name) values ("TG", "Togo");

insert into country (code, name) values ("TK", "Tokelau");

insert into country (code, name) values ("TO", "Tonga");

insert into country (code, name) values ("TT", "Trinidad and Tobago");

insert into country (code, name) values ("TN", "Tunisia");

insert into country (code, name) values ("TR", "Turkey");

insert into country (code, name) values ("TM", "Turkmenistan");

insert into country (code, name) values ("TC", "Turks and Caicos Islands");

insert into country (code, name) values ("TV", "Tuvalu");

insert into country (code, name) values ("UG", "Uganda");

insert into country (code, name) values ("UA", "Ukraine");

insert into country (code, name) values ("AE", "United Arab Emirates");

insert into country (code, name) values ("GB", "United Kingdom");

insert into country (code, name) values ("US", "United States");

insert into country (code, name) values ("UM", "United States Minor Outlying Islands");

insert into country (code, name) values ("UY", "Uruguay");

insert into country (code, name) values ("UZ", "Uzbekistan");

insert into country (code, name) values ("VU", "Vanuatu");

insert into country (code, name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (code, name) values ("VN", "Viet Nam");

insert into country (code, name) values ("VG", "Virgin Islands, British");

insert into country (code, name) values ("VI", "Virgin Islands, U.S.");

insert into country (code, name) values ("WF", "Wallis and Futuna");

insert into country (code, name) values ("EH", "Western Sahara");

insert into country (code, name) values ("YE", "Yemen");

insert into country (code, name) values ("ZM", "Zambia");

insert into country (code, name) values ("ZW", "Zimbabwe");

insert into country (code, name) values ("AX", "Åland Islands");

**Country Repository:**

package com.cognizant.ormlearn.dao;

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

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

import org.springframework.data.repository.query.Param;

import com.cognizant.ormlearn.model.Country;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

// 1. Find countries with names starting with a specific prefix (case-insensitive)

List<Country> findByNameStartingWithIgnoreCase(String prefix);

// 2. Find countries by exact name match (case-insensitive)

List<Country> findByNameIgnoreCase(String name);

// 3. Custom JPQL query to search by partial name match

@Query("SELECT c FROM Country c WHERE LOWER(c.name) LIKE LOWER(CONCAT('%', :name, '%'))")

List<Country> searchByNameContains(@Param("name") String name);

// 4. Get all countries sorted by name

@Query("SELECT c FROM Country c ORDER BY c.name ASC")

List<Country> findAllCountriesSortedByName();

// 5. Find countries whose names contain 'u' and sorted by name

@Query("SELECT c FROM Country c WHERE LOWER(c.name) LIKE '%u%' ORDER BY c.name ASC")

List<Country> findCountriesContainingUAndSorted();

}

**application.properties**

# --- MySQL Database Connection ---

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

spring.datasource.username=root

spring.datasource.password=your\_password\_here

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

# --- JPA / Hibernate Settings ---

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**Country.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

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;

}

}

**CountryServiceTest:**

package com.cognizant.ormlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.dao.CountryRepository;

import com.cognizant.ormlearn.model.Country;

import java.util.List;

@SpringBootApplication

public class CountryServiceTest {

public static void main(String[] args) {

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

CountryRepository countryRepository = context.getBean(CountryRepository.class);

List<Country> countries = countryRepository.findAll();

System.out.println("Countries from DB:");

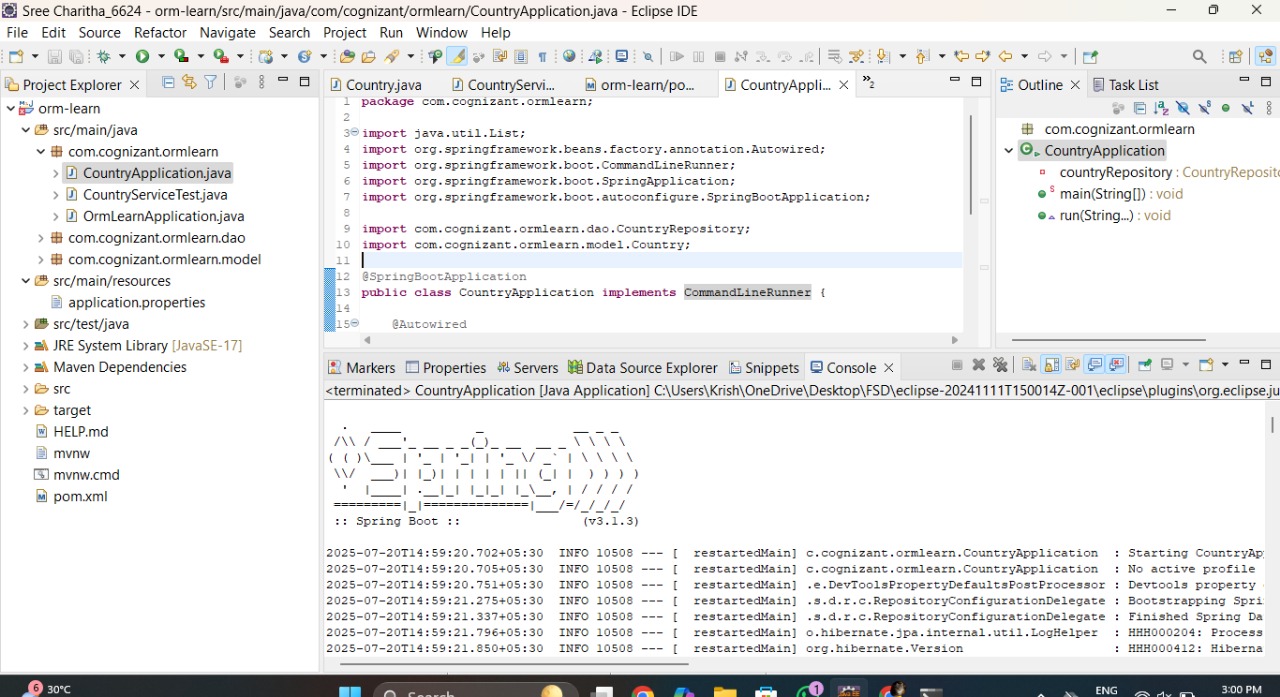
for (Country country : countries) {

System.out.println(country.getCode() + " - " + country.getName());

}

}

}



**Difference between JPA, Hibernate and Spring Data JPA**

**com.cognizant.ormlearn.repository:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String department;

private double salary;

// Constructors

public Employee() {

}

public Employee(String name, String department, double salary) {

this.name = name;

this.department = department;

this.salary = salary;

}

// Getters & Setters

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

// toString for debugging

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", department=" + department + ", salary=" + salary + "]";

}

}

**EmployeeRepository:**

package com.cognizant.ormlearn.repository;

import java.util.List;

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

import com.cognizant.ormlearn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Find employees by name

List<Employee> findByName(String name);

// Find employees whose name contains a substring (case-insensitive)

List<Employee> findByNameContainingIgnoreCase(String keyword);

// Find employees with salary greater than given value

List<Employee> findBySalaryGreaterThan(double salary);

// Find employees with salary between a range

List<Employee> findBySalaryBetween(double minSalary, double maxSalary);

// Find all employees in a given department (if department field exists)

// List<Employee> findByDepartment(String department);

}

**EmployeeService**:

package com.cognizant.ormlearn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public List<Employee> getEmployeesByName(String name) {

return employeeRepository.findByName(name);

}

public List<Employee> getEmployeesWithHighSalary(double salary) {

return employeeRepository.findBySalaryGreaterThan(salary);

}

}

**Creating table of employee:**

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

salary DECIMAL(10, 2),

permanent BOOLEAN,

date\_of\_birth DATE

);

INSERT INTO employee (id, name, salary, permanent, date\_of\_birth) VALUES

(1, 'John Doe', 55000.00, true, '1990-01-15'),

(2, 'Alice Smith', 62000.00, false, '1988-07-30'),

(3, 'Bob Johnson', 47000.00, true, '1992-03-12'),

(4, 'Carol White', 51000.00, false, '1991-08-10'),

(5, 'David Brown', 53000.00, true, '1993-11-25'),

(6, 'Eva Green', 58000.00, true, '1987-02-14'),

(7, 'Frank Black', 49500.00, false, '1985-05-23'),

(8, 'Grace Blue', 61000.00, true, '1990-12-01'),

(9, 'Henry Red', 54000.00, false, '1986-10-18'),

(10, 'Ivy Gold', 60000.00, true, '1989-03-03'),

(11, 'Jack Silver', 47000.00, true, '1991-06-07'),

(12, 'Karen Copper', 63000.00, false, '1988-09-29'),

(13, 'Leo Zinc', 52000.00, true, '1992-04-16'),

(14, 'Mona Lime', 59000.00, false, '1990-07-21'),

(15, 'Nina Teal', 61000.00, true, '1987-01-11'),

(16, 'Oscar Plum', 48500.00, true, '1985-12-31'),

(17, 'Paul Gray', 55000.00, false, '1989-10-10'),

(18, 'Quinn Snow', 57000.00, true, '1993-03-03'),

(19, 'Rita Jade', 60000.00, false, '1986-06-06'),

(20, 'Sam Indigo', 54000.00, true, '1992-05-15'),

(21, 'Tina Maroon', 48000.00, true, '1990-02-18'),

(22, 'Uma Navy', 65000.00, false, '1987-04-24'),

(23, 'Victor Olive', 56000.00, true, '1989-08-08'),

(24, 'Wendy Mint', 62000.00, false, '1988-11-20'),

(25, 'Xander Pearl', 53000.00, true, '1991-07-04'),

(26, 'Yara Coral', 51000.00, false, '1985-03-25'),

(27, 'Zack Rose', 58000.00, true, '1986-09-09'),

(28, 'Amy Violet', 47000.00, false, '1990-10-13'),

(29, 'Ben Sky', 59000.00, true, '1993-01-28'),

(30, 'Cleo Ice', 52000.00, true, '1987-06-19'),

(31, 'Dean Storm', 60000.00, false, '1986-12-15'),

(32, 'Ella Rain', 55000.00, true, '1988-02-09'),

(33, 'Finn Cloud', 53000.00, false, '1992-11-11'),

(34, 'Gina Sun', 62000.00, true, '1989-09-29'),

(35, 'Hank Breeze', 50000.00, false, '1991-05-17'),

(36, 'Irene Hill', 58000.00, true, '1990-03-08'),

(37, 'Jake Sea', 49500.00, true, '1985-08-26'),

(38, 'Kylie Dawn', 57000.00, false, '1987-10-05'),

(39, 'Liam Fire', 61000.00, true, '1986-07-07'),

(40, 'Mia Earth', 54000.00, false, '1988-12-12'),

(41, 'Noah Sky', 56000.00, true, '1992-01-01'),

(42, 'Olivia Star', 50000.00, false, '1989-02-20'),

(43, 'Perry River', 63000.00, true, '1987-03-03'),

(44, 'Quincy Wind', 51000.00, false, '1991-11-11'),

(45, 'Rachel Stone', 59000.00, true, '1985-06-06'),

(46, 'Steve Iron', 48000.00, true, '1990-09-19'),

(47, 'Tara Leaf', 61000.00, false, '1986-04-14'),

(48, 'Umar Cloud', 52000.00, true, '1988-05-05'),

(49, 'Vera Moon', 55000.00, false, '1993-08-18'),

(50, 'Will Ocean', 60000.00, true, '1987-07-23');

**OUTPUT** :

