**Question 01:**

Consider a database application with **Department** and **Employee** classes.

The **Department** class represents various departments within an organization, while the **Employee** class represents individual employees.

Each employee is associated with a specific department.

1. Define a Java class called **Department** with the necessary fields and attributes to accurately represent a department within an organizational database application.

The fields will look like below

1. **id** (int): A unique identifier for the department.
2. **name** (String): The name of the department.
3. **description** (String): A brief description or summary of the department's purpose.
4. **location** (String): The physical location or office where the department operates.
5. **manager** (Employee): A reference to the employee who manages the department.
6. **employees** (List<Employee>): A list of employees who belong to the department.

Department.java

import javax.persistence.\*;

import java.util.List;

@Entity

@Table(name = "departments")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "name")

private String name;

@Column(name = "description")

private String description;

@Column(name = "location")

private String location;

@ManyToOne

@JoinColumn(name = "manager\_id")

private Employee manager;

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

private List<Employee> employees;

public Department() {

// Default constructor

}

public Department(String name, String description, String location, Employee manager) {

this.name = name;

this.description = description;

this.location = location;

this.manager = manager;

}

// Getters and setters for the fields

// Additional methods as needed

// You can also add additional methods to handle department-related operations

}

1. To accurately represent employees in an organizational database application, create a Java class called **Employee** with the necessary fields and attributes. Define the key fields that should be included in the **Employee** class to effectively model and manage employee-related information. Explain the significance of each field and how it contributes to accurately representing employee data within the application.

The fields will look like below

1. **id** (int): A unique identifier for the employee.
2. **firstName** (String): The employee's first name.
3. **lastName** (String): The employee's last name.
4. **email** (String): The employee's email address.
5. **phoneNumber** (String): The employee's contact phone number.
6. **hireDate** (Date): The date the employee was hired.
7. **jobTitle** (String): The employee's job title or position within the organization.
8. **salary** (double): The employee's salary or compensation.
9. **department** (Department): A reference to the department to which the employee belongs.

Employee.java

import javax.persistence.\*;

import java.util.Date;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "email")

private String email;

@Column(name = "phone\_number")

private String phoneNumber;

@Column(name = "hire\_date")

private Date hireDate;

@Column(name = "job\_title")

private String jobTitle;

@Column(name = "salary")

private double salary;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

public Employee() {

// Default constructor

}

public Employee(String firstName, String lastName, String email, String phoneNumber, Date hireDate, String jobTitle, double salary, Department department) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.phoneNumber = phoneNumber;

this.hireDate = hireDate;

this.jobTitle = jobTitle;

this.salary = salary;

this.department = department;

}

// Getters and setters for the fields

// Additional methods as needed

// You can also add additional methods to handle employee-related operations

}

1. **Now create another class TestDepartmentEmployees.java, in which you need to store 5 departments and 50 employees**

**The data interms of objects has given for you**

// Create and store 5 departments with real department names

Department department1 = new Department("Human Resources", "HR Department", "Location A", null);

Department department2 = new Department("Information Technology", "IT Department", "Location B", null);

Department department3 = new Department("Sales and Marketing", "Sales Department", "Location C", null);

Department department4 = new Department("Finance and Accounting", "Finance Department", "Location D", null);

Department department5 = new Department("Research and Development", "R&D Department", "Location E", null);

// Create and store 25 employee objects

Employee employee1 = new Employee("John", "Doe", "john.doe@example.com", "123-456-7890", new Date(), "Software Developer", 60000.0, department1);

Employee employee2 = new Employee("Alice", "Smith", "alice.smith@example.com", "987-654-3210", new Date(), "HR Specialist", 55000.0, department1);

Employee employee3 = new Employee("Bob", "Johnson", "bob.johnson@example.com", "555-123-7890", new Date(), "Data Analyst", 52000.0, department1);

Employee employee4 = new Employee("Emily", "Brown", "emily.brown@example.com", "456-789-1230", new Date(), "Marketing Coordinator", 51000.0, department1);

Employee employee5 = new Employee("Daniel", "Williams", "daniel.williams@example.com", "222-333-4444", new Date(), "Recruiter", 53000.0, department1);

Employee employee6 = new Employee("Sara", "Wilson", "sara.wilson@example.com", "111-222-3333", new Date(), "System Analyst", 58000.0, department2);

Employee employee7 = new Employee("Michael", "Johnson", "michael.johnson@example.com", "333-444-5555", new Date(), "Software Engineer", 62000.0, department2);

Employee employee8 = new Employee("Olivia", "Smith", "olivia.smith@example.com", "777-888-9999", new Date(), "Network Administrator", 59000.0, department2);

Employee employee9 = new Employee("William", "Brown", "william.brown@example.com", "444-555-6666", new Date(), "Database Administrator", 60000.0, department2);

Employee employee10 = new Employee("Sophia", "Davis", "sophia.davis@example.com", "999-888-7777", new Date(), "UI/UX Designer", 58000.0, department2);

Employee employee11 = new Employee("Emma", "Johnson", "emma.johnson@example.com", "555-666-7777", new Date(), "Sales Manager", 65000.0, department3);

Employee employee12 = new Employee("Noah", "Smith", "noah.smith@example.com", "444-333-2222", new Date(), "Sales Representative", 54000.0, department3);

Employee employee13 = new Employee("Ava", "Brown", "ava.brown@example.com", "999-888-7777", new Date(), "Marketing Manager", 63000.0, department3);

Employee employee14 = new Employee("Liam", "Davis", "liam.davis@example.com", "111-222-3333", new Date(), "Marketing Specialist", 55000.0, department3);

Employee employee15 = new Employee("Oliver", "Wilson", "oliver.wilson@example.com", "888-777-6666", new Date(), "Sales Representative", 54000.0, department3);

Employee employee16 = new Employee("Mia", "Anderson", "mia.anderson@example.com", "555-444-3333", new Date(), "Sales Representative", 54000.0, department4);

Employee employee17 = new Employee("Lucas", "Martinez", "lucas.martinez@example.com", "777-666-5555", new Date(), "Marketing Coordinator", 52000.0, department4);

Employee employee18 = new Employee("Sophia", "Hernandez", "sophia.hernandez@example.com", "333-222-1111", new Date(), "Sales Representative", 54000.0, department4);

Employee employee19 = new Employee("Logan", "Lopez", "logan.lopez@example.com", "222-333-4444", new Date(), "Marketing Specialist", 52000.0, department4);

Employee employee20 = new Employee("Evelyn", "Gonzalez", "evelyn.gonzalez@example.com", "111-222-3333", new Date(), "Sales Manager", 62000.0, department4);

Employee employee21 = new Employee("James", "Anderson", "james.anderson@example.com", "555-444-3333", new Date(), "Research Scientist", 65000.0, department5);

Employee employee22 = new Employee("Emma", "Martinez", "emma.martinez@example.com", "777-666-5555", new Date(), "Lab Technician", 52000.0, department5);

Employee employee23 = an Employee("William", "Hernandez", "william.hernandez@example.com", "333-222-1111", new Date(), "Data Analyst", 54000.0, department5);

Employee employee24 = new Employee("Sophia", "Lopez", "sophia.lopez@example.com", "222-333-4444", new Date(), "Research Assistant", 52000.0, department5);

Employee employee25 = new Employee("Oliver", "Gonzalez", "oliver.gonzalez@example.com", "111-222-3333", new Date(), "R&D Manager", 62000.0, department5);

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.util.Date;

public class TestDepartmentEmployees {

public static void main(String[] args) {

// Create a Hibernate configuration and session factory

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Department.class)

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

// Create a Hibernate session

Session session = factory.getCurrentSession();

try {

// Start a transaction

session.beginTransaction();

// Create and store 5 departments with real department names

Department department1 = new Department("Human Resources", "HR Department", "Location A", null);

Department department2 = new Department("Information Technology", "IT Department", "Location B", null);

Department department3 = new Department("Sales and Marketing", "Sales Department", "Location C", null);

Department department4 = new Department("Finance and Accounting", "Finance Department", "Location D", null);

Department department5 = new Department("Research and Development", "R&D Department", "Location E", null);

// Create and store 25 employee objects

Employee employee1 = new Employee("John", "Doe", "john.doe@example.com", "123-456-7890", new Date(), "Software Developer", 60000.0, department1);

Employee employee2 = new Employee("Alice", "Smith", "alice.smith@example.com", "987-654-3210", new Date(), "HR Specialist", 55000.0, department1);

Employee employee3 = new Employee("Bob", "Johnson", "bob.johnson@example.com", "555-123-7890", new Date(), "Data Analyst", 52000.0, department1);

Employee employee4 = new Employee("Emily", "Brown", "emily.brown@example.com", "456-789-1230", new Date(), "Marketing Coordinator", 51000.0, department1);

Employee employee5 = new Employee("Daniel", "Williams", "daniel.williams@example.com", "222-333-4444", new Date(), "Recruiter", 53000.0, department1);

Employee employee6 = new Employee("Sara", "Wilson", "sara.wilson@example.com", "111-222-3333", new Date(), "System Analyst", 58000.0, department2);

Employee employee7 = new Employee("Michael", "Johnson", "michael.johnson@example.com", "333-444-5555", new Date(), "Software Engineer", 62000.0, department2);

Employee employee8 = new Employee("Olivia", "Smith", "olivia.smith@example.com", "777-888-9999", new Date(), "Network Administrator", 59000.0, department2);

Employee employee9 = new Employee("William", "Brown", "william.brown@example.com", "444-555-6666", new Date(), "Database Administrator", 60000.0, department2);

Employee employee10 = new Employee("Sophia", "Davis", "sophia.davis@example.com", "999-888-7777", new Date(), "UI/UX Designer", 58000.0, department2);

Employee employee11 = new Employee("Emma", "Johnson", "emma.johnson@example.com", "555-666-7777", new Date(), "Sales Manager", 65000.0, department3);

Employee employee12 = new Employee("Noah", "Smith", "noah.smith@example.com", "444-333-2222", new Date(), "Sales Representative", 54000.0, department3);

Employee employee13 = new Employee("Ava", "Brown", "ava.brown@example.com", "999-888-7777", new Date(), "Marketing Manager", 63000.0, department3);

Employee employee14 = new Employee("Liam", "Davis", "liam.davis@example.com", "111-222-3333", new Date(), "Marketing Specialist", 55000.0, department3);

Employee employee15 = new Employee("Oliver", "Wilson", "oliver.wilson@example.com", "888-777-6666", new Date(), "Sales Representative", 54000.0, department3);

Employee employee16 = new Employee("Mia", "Anderson", "mia.anderson@example.com", "555-444-3333", new Date(), "Sales Representative", 54000.0, department4);

Employee employee17 = new Employee("Lucas", "Martinez", "lucas.martinez@example.com", "777-666-5555", new Date(), "Marketing Coordinator", 52000.0, department4);

Employee employee18 = new Employee("Sophia", "Hernandez", "sophia.hernandez@example.com", "333-222-1111", new Date(), "Sales Representative", 54000.0, department4);

Employee employee19 = new Employee("Logan", "Lopez", "logan.lopez@example.com", "222-333-4444", new Date(), "Marketing Specialist", 52000.0, department4);

Employee employee20 = new Employee("Evelyn", "Gonzalez", "evelyn.gonzalez@example.com", "111-222-3333", new Date(), "Sales Manager", 62000.0, department4);

Employee employee21 = new Employee("James", "Anderson", "james.anderson@example.com", "555-444-3333", new Date(), "Research Scientist", 65000.0, department5);

Employee employee22 = new Employee("Emma", "Martinez", "emma.martinez@example.com", "777-666-5555", new Date(), "Lab Technician", 52000.0, department5);

Employee employee23 = an Employee("William", "Hernandez", "william.hernandez@example.com", "333-222-1111", new Date(), "Data Analyst", 54000.0, department5);

Employee employee24 = new Employee("Sophia", "Lopez", "sophia.lopez@example.com", "222-333-4444", new Date(), "Research Assistant", 52000.0, department5);

Employee employee25 = new Employee("Oliver", "Gonzalez", "oliver.gonzalez@example.com", "111-222-3333", new Date(), "R&D Manager", 62000.0, department5);

// Save the departments to the database

session.save(department1);

session.save(department2);

session.save(department3);

session.save(department4);

session.save(department5);

session.save(employee1);

session.save(employee2);

session.save(employee3);

session.save(employee4);

session.save(employee5);

session.save(employee6);

session.save(employee7);

session.save(employee8);

session.save(employee9);

session.save(employee10);

session.save(employee11);

session.save(employee12);

session.save(employee13);

session.save(employee14);

session.save(employee15);

session.save(employee16);

session.save(employee17);

session.save(employee18);

session.save(employee19);

session.save(employee20);

session.save(employee21);

session.save(employee22);

session.save(employee23);

session.save(employee24);

session.save(employee25);

// Commit the transaction

session.getTransaction().commit();

} finally {

factory.close();

}

}

}

**4. Now try to solve the following problems**

1. **Retrieve all departments with their respective employee counts.**
2. **Retrieve the department name and the highest salary among its employees for each department.**
3. **Find the employees who earn more than $60,000 in the "IT" department.**
4. **List all employees in the "HR" department in alphabetical order.**
5. **Retrieve the departments with no employees.**
6. **Find the employee with the highest salary in the organization.**
7. **List employees in the "Finance" department hired after a specific date.**
8. **Retrieve the total number of employees in each department.**
9. **Find the employees in the "Research and Development" department with a salary between $50,000 and $60,000.**
10. **List employees and their respective departments in alphabetical order.**
11. **Retrieve a list of employees along with their department names.**
12. **Find employees in the "Finance" department and include their department details.**
13. **List employees and their respective departments using an INNER JOIN.**
14. **Retrieve employees in the "IT" department along with their department descriptions.**
15. **Find employees hired after a specific date and include their department names.**
16. **Retrieve the department names along with the total salary of employees in each department.**
17. **List departments with their respective employees' email addresses.**
18. **Retrieve the departments with the highest and lowest average employee salaries.**
19. **List employees in the "HR" department and their supervisors.**
20. **Retrieve employees who have the same supervisors as "John Doe."**
21. **Retrieve employees and their respective departments, but only for the first three employees.**
22. **Find employees who are in departments with a specific description.**
23. **List employees in the "IT" department and their supervisors using join.**
24. **Retrieve employees who are supervised by "John Doe" and include their department names.**
25. **List the departments with their respective department managers.**
26. **Find employees in the "HR" department and include their supervisors using join.**
27. **Retrieve the department names and the average salary of employees in each department using a join.**
28. **List employees in the "Finance" department and their department manager's name.**

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class HibernateHQLExample {

public static void main(String[] args) {

// Create a Hibernate configuration and session factory

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Department.class)

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

// Create a Hibernate session

Session session = factory.getCurrentSession();

try {

// Start a transaction

session.beginTransaction();

// Query 1: Retrieve all departments with their respective employee counts.

String hql1 = "SELECT d.name, COUNT(e) FROM Department d LEFT JOIN d.employees e GROUP BY d";

List<Object[]> result1 = session.createQuery(hql1).list();

System.out.println("Query 1: All departments with employee counts:");

for (Object[] row : result1) {

System.out.println("Department: " + row[0] + ", Employee Count: " + row[1]);

}

// Query 2: Retrieve the department name and the highest salary among its employees for each department.

String hql2 = "SELECT d.name, MAX(e.salary) FROM Department d LEFT JOIN d.employees e GROUP BY d";

List<Object[]> result2 = session.createQuery(hql2).list();

System.out.println("\nQuery 2: Department names and highest salaries:");

for (Object[] row : result2) {

System.out.println("Department: " + row[0] + ", Highest Salary: " + row[1]);

}

// Query 3: Find the employees who earn more than $60,000 in the "IT" department.

String hql3 = "SELECT e FROM Employee e WHERE e.salary > 60000 AND e.department.name = 'Information Technology'";

List<Employee> result3 = session.createQuery(hql3, Employee.class).list();

System.out.println("\nQuery 3: Employees in IT department with salary > $60,000:");

for (Employee employee : result3) {

System.out.println(employee.getFirstName() + " " + employee.getLastName());

}

// Query 4: List all employees in the "HR" department in alphabetical order.

String hql4 = "SELECT e FROM Employee e WHERE e.department.name = 'Human Resources' ORDER BY e.lastName, e.firstName";

List<Employee> result4 = session.createQuery(hql4, Employee.class).list();

System.out.println("\nQuery 4: Employees in HR department (sorted by last name, first name):");

for (Employee employee : result4) {

System.out.println(employee.getFirstName() + " " + employee.getLastName());

}

// Query 5: Retrieve the departments with no employees.

String hql5 = "SELECT d FROM Department d WHERE SIZE(d.employees) = 0";

List<Department> result5 = session.createQuery(hql5, Department.class).list();

System.out.println("\nQuery 5: Departments with no employees:");

for (Department department : result5) {

System.out.println(department.getName());

}

// Query 6: Find the employee with the highest salary in the organization.

String hql6 = "SELECT e FROM Employee e WHERE e.salary = (SELECT MAX(e2.salary) FROM Employee e2)";

Employee result6 = session.createQuery(hql6, Employee.class).uniqueResult();

System.out.println("\nQuery 6: Employee with the highest salary in the organization:");

System.out.println(result6.getFirstName() + " " + result6.getLastName());

// Query 7: List employees in the "Finance" department hired after a specific date.

String hql7 = "SELECT e FROM Employee e WHERE e.department.name = 'Finance' AND e.hireDate > :specificDate";

List<Employee> result7 = session.createQuery(hql7, Employee.class)

.setParameter("specificDate", yourSpecificDate) // Provide the specific date

.list();

System.out.println("\nQuery 7: Employees in Finance department hired after a specific date:");

for (Employee employee : result7) {

System.out.println(employee.getFirstName() + " " + employee.getLastName());

}

// Query 8: Retrieve the total number of employees in each department.

String hql8 = "SELECT d.name, COUNT(e) FROM Department d LEFT JOIN d.employees e GROUP BY d";

List<Object[]> result8 = session.createQuery(hql8).list();

System.out.println("\nQuery 8: Total number of employees in each department:");

for (Object[] row : result8) {

System.out.println("Department: " + row[0] + ", Total Employees: " + row[1]);

}

// Query 9: Find the employees in the "Research and Development" department with a salary between $50,000 and $60,000.

String hql9 = "SELECT e FROM Employee e WHERE e.department.name = 'Research and Development' AND e.salary BETWEEN 50000 AND 60000";

List<Employee> result9 = session.createQuery(hql9, Employee.class).list();

System.out.println("\nQuery 9: Employees in R&D department with salary between $50,000 and $60,000:");

for (Employee employee : result9) {

System.out.println(employee.getFirstName() + " " + employee.getLastName());

}

// Query 10: List employees and their respective departments in alphabetical order.

String hql10 = "SELECT e, e.department FROM Employee e ORDER BY e.lastName, e.firstName";

List<Object[]> result10 = session.createQuery(hql10).list();

System.out.println("\nQuery 10: Employees and their departments (sorted by last name, first name):");

for (Object[] row : result10) {

Employee employee = (Employee) row[0];

Department department = (Department) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Department: " + department.getName());

}

// Commit the transaction

session.getTransaction().commit();

} finally {

factory.close();

}

}

}

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class HibernateHQLWithJoinExample {

public static void main(String[] args) {

// Create a Hibernate configuration and session factory

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Department.class)

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

// Create a Hibernate session

Session session = factory.getCurrentSession();

try {

// Start a transaction

session.beginTransaction();

// Query 11: Retrieve a list of employees along with their department names.

String hql11 = "SELECT e.firstName, e.lastName, d.name FROM Employee e JOIN e.department d";

List<Object[]> result11 = session.createQuery(hql11).list();

System.out.println("Query 11: Employees and their department names:");

for (Object[] row : result11) {

System.out.println("Employee: " + row[0] + " " + row[1] + ", Department: " + row[2]);

}

// Query 12: Find employees in the "Finance" department and include their department details.

String hql12 = "SELECT e, d FROM Employee e JOIN e.department d WHERE d.name = 'Finance'";

List<Object[]> result12 = session.createQuery(hql12).list();

System.out.println("\nQuery 12: Employees in the Finance department with department details:");

for (Object[] row : result12) {

Employee employee = (Employee) row[0];

Department department = (Department) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Department: " + department.getName());

}

// Query 13: List employees and their respective departments using an INNER JOIN.

String hql13 = "SELECT e, d FROM Employee e JOIN e.department d";

List<Object[]> result13 = session.createQuery(hql13).list();

System.out.println("\nQuery 13: Employees and their respective departments using INNER JOIN:");

for (Object[] row : result13) {

Employee employee = (Employee) row[0];

Department department = (Department) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Department: " + department.getName());

}

// Query 14: Retrieve employees in the "IT" department along with their department descriptions.

String hql14 = "SELECT e, d.description FROM Employee e JOIN e.department d WHERE d.name = 'Information Technology'";

List<Object[]> result14 = session.createQuery(hql14).list();

System.out.println("\nQuery 14: Employees in IT department with department descriptions:");

for (Object[] row : result14) {

Employee employee = (Employee) row[0];

String departmentDescription = (String) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Department Description: " + departmentDescription);

}

// Query 15: Find employees hired after a specific date and include their department names.

String hql15 = "SELECT e.firstName, e.lastName, d.name FROM Employee e JOIN e.department d WHERE e.hireDate > :specificDate";

List<Object[]> result15 = session.createQuery(hql15)

.setParameter("specificDate", yourSpecificDate) // Provide the specific date

.list();

System.out.println("\nQuery 15: Employees hired after a specific date with their department names:");

for (Object[] row : result15) {

System.out.println("Employee: " + row[0] + " " + row[1] + ", Department: " + row[2]);

}

// Query 16: Retrieve the department names along with the total salary of employees in each department.

String hql16 = "SELECT d.name, SUM(e.salary) FROM Department d JOIN d.employees e GROUP BY d.name";

List<Object[]> result16 = session.createQuery(hql16).list();

System.out.println("\nQuery 16: Department names and total salary of employees:");

for (Object[] row : result16) {

System.out.println("Department: " + row[0] + ", Total Salary: " + row[1]);

}

// Query 17: List departments with their respective employees' email addresses.

String hql17 = "SELECT d, e.email FROM Department d LEFT JOIN d.employees e";

List<Object[]> result17 = session.createQuery(hql17).list();

System.out.println("\nQuery 17: Departments and their employees' email addresses:");

for (Object[] row : result17) {

Department department = (Department) row[0];

String employeeEmail = (String) row[1];

System.out.println("Department: " + department.getName() + ", Employee Email: " + employeeEmail);

}

// Query 18: Retrieve the departments with the highest and lowest average employee salaries.

String hql18 = "SELECT d.name, AVG(e.salary) FROM Department d JOIN d.employees e GROUP BY d.name " +

"HAVING AVG(e.salary) = MAX(AVG(e.salary)) OR AVG(e.salary) = MIN(AVG(e.salary))";

List<Object[]> result18 = session.createQuery(hql18).list();

System.out.println("\nQuery 18: Departments with highest and lowest average employee salaries:");

for (Object[] row : result18) {

System.out.println("Department: " + row[0] + ", Average Salary: " + row[1]);

}

// Query 19: List employees in the "HR" department and their supervisors.

String hql19 = "SELECT e.firstName, e.lastName, e.supervisor.firstName, e.supervisor.lastName " +

"FROM Employee e WHERE e.department.name = 'Human Resources'";

List<Object[]> result19 = session.createQuery(hql19).list();

System.out.println("\nQuery 19: Employees in HR department and their supervisors:");

for (Object[] row : result19) {

System.out.println("Employee: " + row[0] + " " + row[1] + ", Supervisor: " + row[2] + " " + row[3]);

}

// Query 20: Retrieve employees who have the same supervisors as "John Doe."

String hql20 = "SELECT e.firstName, e.lastName FROM Employee e WHERE e.supervisor.firstName = 'John' AND e.supervisor.lastName = 'Doe'";

List<Object[]> result20 = session.createQuery(hql20).list();

System.out.println("\nQuery 20: Employees with the same supervisors as John Doe:");

for (Object[] row : result20) {

System.out.println("Employee: " + row[0] + " " + row[1]);

}

// Commit the transaction

session.getTransaction().commit();

} finally {

factory.close();

}

}

}

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class HibernateHQLWithJoinExample {

public static void main(String[] args) {

// Create a Hibernate configuration and session factory

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Department.class)

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

// Create a Hibernate session

Session session = factory.getCurrentSession();

try {

// Start a transaction

session.beginTransaction();

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

String hql21 = "SELECT e, d FROM Employee e JOIN e.department d";

List<Object[]> result21 = session.createQuery(hql21).setMaxResults(3).list();

System.out.println("Query 21: Employees and their departments (first three only):");

for (Object[] row : result21) {

Employee employee = (Employee) row[0];

Department department = (Department) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Department: " + department.getName());

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

String hql22 = "SELECT e FROM Employee e JOIN e.department d WHERE d.description = :description";

List<Employee> result22 = session.createQuery(hql22, Employee.class)

.setParameter("description", yourDescription) // Provide the specific description

.list();

System.out.println("\nQuery 22: Employees in departments with a specific description:");

for (Employee employee : result22) {

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName());

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

String hql23 = "SELECT e, e.supervisor FROM Employee e JOIN e.department d WHERE d.name = 'Information Technology'";

List<Object[]> result23 = session.createQuery(hql23).list();

System.out.println("\nQuery 23: Employees in the IT department and their supervisors:");

for (Object[] row : result23) {

Employee employee = (Employee) row[0];

Employee supervisor = (Employee) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Supervisor: " + supervisor.getFirstName() + " " + supervisor.getLastName());

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

// Query 24: Retrieve employees who are supervised by "John Doe" and include their department names.

String hql24 = "SELECT e.firstName, e.lastName, d.name FROM Employee e JOIN e.supervisor s JOIN e.department d WHERE s.firstName = 'John' AND s.lastName = 'Doe'";

List<Object[]> result24 = session.createQuery(hql24).list();

System.out.println("\nQuery 24: Employees supervised by John Doe with their department names:");

for (Object[] row : result24) {

System.out.println("Employee: " + row[0] + " " + row[1] + ", Department: " + row[2]);

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

// Query 24: Retrieve employees who are supervised by "John Doe" and include their department names.

// Query 25: List the departments with their respective department managers.

String hql25 = "SELECT d, d.manager FROM Department d";

List<Object[]> result25 = session.createQuery(hql25).list();

System.out.println("\nQuery 25: Departments and their managers:");

for (Object[] row : result25) {

Department department = (Department) row[0];

Employee manager = (Employee) row[1];

System.out.println("Department: " + department.getName() + ", Manager: " + manager.getFirstName() + " " + manager.getLastName());

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

// Query 24: Retrieve employees who are supervised by "John Doe" and include their department names.

// Query 25: List the departments with their respective department managers.

// Query 26: Find employees in the "HR" department and include their supervisors using join.

String hql26 = "SELECT e, e.supervisor FROM Employee e JOIN e.department d WHERE d.name = 'Human Resources'";

List<Object[]> result26 = session.createQuery(hql26).list();

System.out.println("\nQuery 26: Employees in the HR department and their supervisors:");

for (Object[] row : result26) {

Employee employee = (Employee) row[0];

Employee supervisor = (Employee) row[1];

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() + ", Supervisor: " + supervisor.getFirstName() + " " + supervisor.getLastName());

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

// Query 24: Retrieve employees who are supervised by "John Doe" and include their department names.

// Query 25: List the departments with their respective department managers.

// Query 26: Find employees in the "HR" department and include their supervisors using join.

// Query 27: Retrieve the department names and the average salary of employees in each department using a join.

String hql27 = "SELECT d.name, AVG(e.salary) FROM Department d JOIN d.employees e GROUP BY d.name";

List<Object[]> result27 = session.createQuery(hql27).list();

System.out.println("\nQuery 27: Department names and average salaries:");

for (Object[] row : result27) {

System.out.println("Department: " + row[0] + ", Average Salary: " + row[1]);

}

// Query 21: Retrieve employees and their respective departments, but only for the first three employees.

// Query 22: Find employees who are in departments with a specific description.

// Query 23: List employees in the "IT" department and their supervisors using join.

// Query 24: Retrieve employees who are supervised by "John Doe" and include their department names.

// Query 25: List the departments with their respective department managers.

// Query 26: Find employees in the "HR" department and include their supervisors using join.

// Query 27: Retrieve the department names and the average salary of employees in each department using a join.

// Query 28: List employees in the "Finance" department and their department manager's name.

String hql28 = "SELECT e.firstName, e.lastName, d.manager.firstName, d.manager.lastName FROM Employee e JOIN e.department d WHERE d.name = 'Finance'";

List<Object[]> result28 = session.createQuery(hql28).list();

System.out.println("\nQuery 28: Employees in the Finance department and their manager:");

for (Object[] row : result28) {

System.out.println("Employee: " + row[0] + " " + row[1] + ", Manager: " + row[2] + " " + row[3]);

}

// Commit the transaction

session.getTransaction().commit();

} finally {

factory.close();

}

}

}