

Digital Nurture – 4.0 JAVA FSE – Week 3 Assignments

File name: Spring Core_Maven

Name: Exercise 1: Configuring a Basic Spring Application

Pom.xml;

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>com.library</groupId>
```

```
  <artifactId>LibraryManagement</artifactId>
```

```
  <version>1.0</version>
```

```
  <dependencies>
```

```
    <!-- Spring Core Dependency -->
```

```
    <dependency>
```

```
      <groupId>org.springframework</groupId>
```

```
      <artifactId>spring-context</artifactId>
```

```
      <version>5.3.36</version>
```

```
    </dependency>
```

```
  </dependencies>
```

```
</project>
```

LibraryApp.java;

```
package com.library;
```

```
import com.library.service.BookService;
```

```
import org.springframework.context.ApplicationContext;
```

```
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class LibraryApp {
```

```
  public static void main(String[] args) {
```

```
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
```

```
    BookService service = (BookService) context.getBean("bookService");
```

```
    service.addBook("Java Programming");
```

```
  }
```

```
}
```

ApplicationContext.xml;

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Repository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository"/>

    <!-- Service Bean (injecting bookRepository) -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>
```

BookService.java;

```
package com.library.service;
```

```
import com.library.repository.BookRepository;
```

```
public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

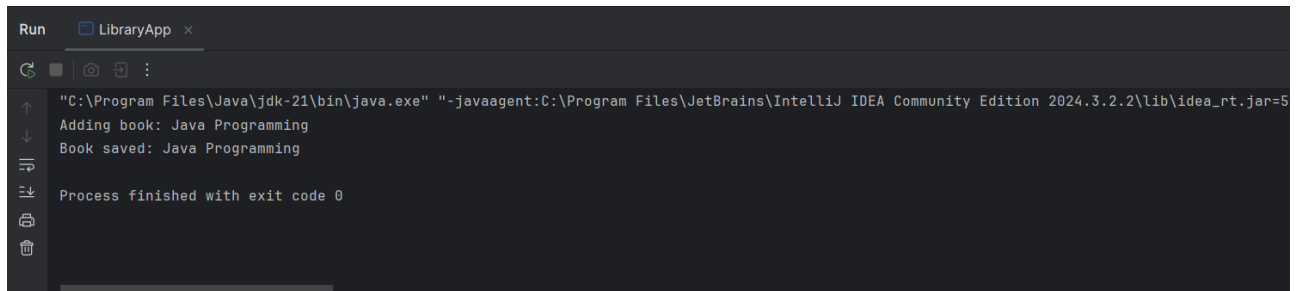
BookRepository.java;

```
package com.library.repository;
```

```
public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
    }
}
```

```
}  
}
```

Output:



```
Run LibraryApp x  
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.2\lib\idea_rt.jar=5  
Adding book: Java Programming  
Book saved: Java Programming  
Process finished with exit code 0
```

File name: Spring Core_Maven

Name: Exercise 2: Implementing Dependency Injection

Pom.xml;

```
<project xmlns="http://maven.apache.org/POM/4.0.0"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.library</groupId>  
<artifactId>LibraryManagement</artifactId>  
<version>1.0</version>
```

```
<dependencies>  
    <!-- Spring Core Dependency -->  
    <dependency>  
        <groupId>org.springframework</groupId>  
        <artifactId>spring-context</artifactId>  
        <version>5.3.36</version>  
    </dependency>  
</dependencies>
```

```
</project>
```

LibraryApp.java;

```
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```

public class LibraryApp {
    public static void main(String[] args) {
        ApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");

        BookService service = (BookService) context.getBean("bookService");
        service.addBook("Spring in Action");
    }
}

```

ApplicationContext.xml;

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- BookRepository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository"/>

    <!-- BookService Bean with dependency injected -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>

```

BookService.java;

```

package com.library.service;

```

```

import com.library.repository.BookRepository;

```

```

public class BookService {
    private BookRepository bookRepository;

    // Setter for Spring DI
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {

```

```

        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}

```

BookRepository.java;

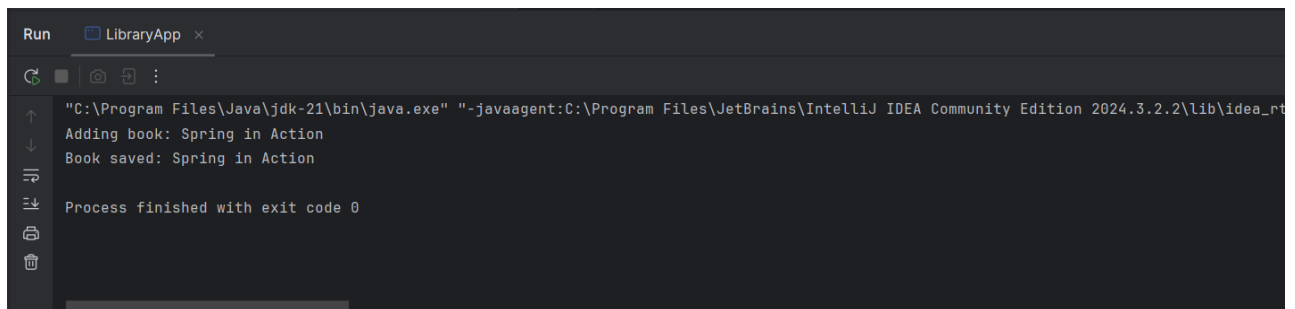
```
package com.library.repository;
```

```

public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book saved: " + bookName);
    }
}

```

Output:



File name: Spring Core_Maven

Name: Exercise 4: Creating and Configuring a Maven Project

Pom.xml;

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">

```

```

    <modelVersion>4.0.0</modelVersion>

```

```

    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0</version>

```

```

    <dependencies>
        <!-- Spring Core Dependency -->
        <dependency>

```

```
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.36</version>
    </dependency>
</dependencies>
```

```
</project>
```

```
LibraryManagementApplication.java;
```

```
package org.example;
```

```
import org.example.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class LibraryManagementApplication {
    public static void main(String[] args) {
        // Load Spring context from XML
        ApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");

        // Get the BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

        // Add multiple books
        bookService.addBook("Spring in Action");
        bookService.addBook("Clean Code");
        bookService.addBook("Effective Java");
        bookService.addBook("Head First Design Patterns");
    }
}
```

```
BookService.java;
```

```
package org.example.service;
```

```
import org.example.repository.BookRepository;
```

```
public class BookService {
    private BookRepository bookRepository;

    // Setter for Spring DI
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
}
```

```

    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}

BookRepository.java;

package org.example.repository;

import java.util.ArrayList;
import java.util.List;

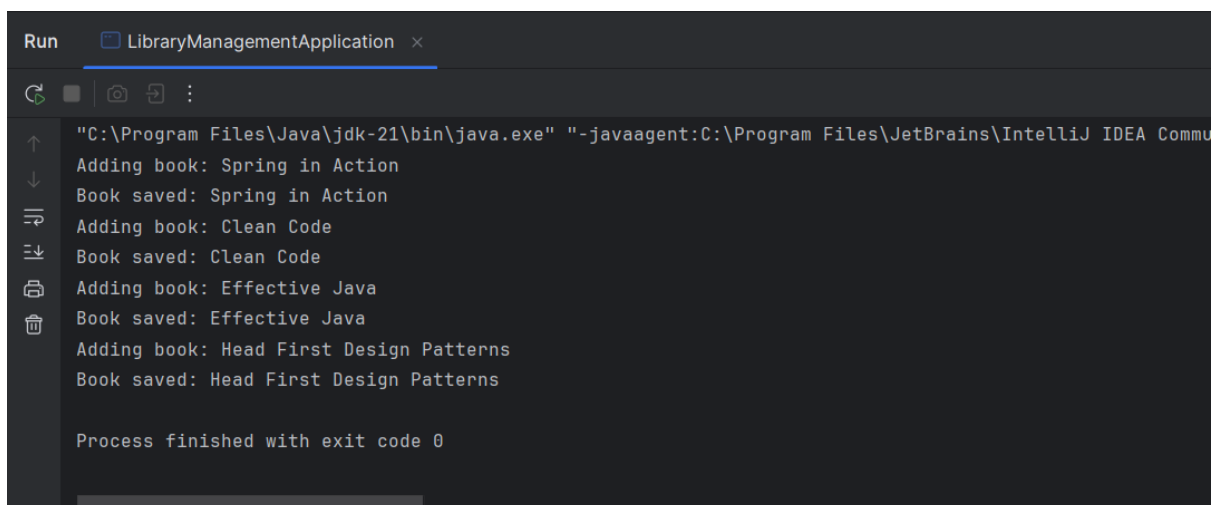
public class BookRepository {
    private List<String> bookDatabase = new ArrayList<>();

    public void saveBook(String bookName) {
        bookDatabase.add(bookName);
        System.out.println("Book saved: " + bookName);
    }

    public List<String> getAllBooks() {
        return bookDatabase;
    }
}

```

Output:



```

Run  LibraryManagementApplication x
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu
Adding book: Spring in Action
Book saved: Spring in Action
Adding book: Clean Code
Book saved: Clean Code
Adding book: Effective Java
Book saved: Effective Java
Adding book: Head First Design Patterns
Book saved: Head First Design Patterns

Process finished with exit code 0

```

File name: 1. spring-data-jpa-handson

Name: Spring Data JPA - Quick Example

OrmLearnApplication.java(Main class);

package com.cognizant.orm_learn;

```
import com.cognizant.orm_learn.model.Department;
import com.cognizant.orm_learn.repository.DepartmentRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
```

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private DepartmentRepository departmentRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

@Override

public void run(String... args) throws Exception {

Department dept = new Department("HR");

departmentRepository.save(dept);

LOGGER.info("Saved Department: " + dept.getName());

}

}

Department.java;

package com.cognizant.orm_learn.model;

import jakarta.persistence.*;


```

@Entity
@Table(name = "department")
public class Department {

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

    private String name;

    public Department() {}
    public Department(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

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

DepartmentRepository.java;

package com.cognizant.orm_learn.repository;

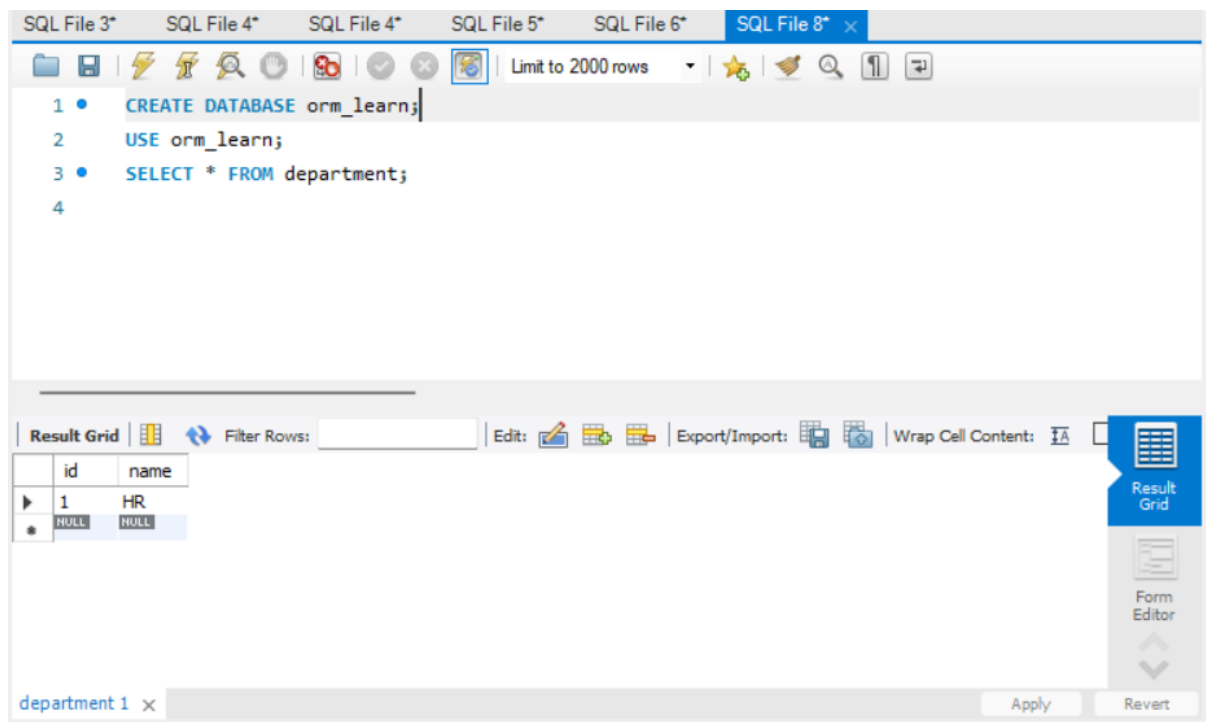
import com.cognizant.orm_learn.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}

Sql code;
Create database orm_learn;
Use database orm_learn;
Select*from Department;

```

Output:



File name: 1. spring-data-jpa-handson

Name: Difference between JPA, Hibernate and Spring Data JPA

1) Hibernate implementation;

EmployeeDAO.java;

package com.example.hibernate.dao;

import com.example.hibernate.model.Employee;

import com.example.hibernate.util.HibernateUtil;

import org.hibernate.Session;

import org.hibernate.Transaction;

```
public class EmployeeDAO {  
    public Integer addEmployee(Employee emp) {  
        Transaction tx = null;  
        Integer id = null;  
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {  
            tx = session.beginTransaction();  
            id = (Integer) session.save(emp);  
            tx.commit();  
        } catch (Exception e) {  
            if (tx != null) tx.rollback();  
            e.printStackTrace();  
        }  
    }  
}
```

```
    }  
    return id;  
}  
}
```

Employee.java;

```
package com.example.hibernate.model;
```

```
import jakarta.persistence.*;
```

```
@Entity
```

```
@Table(name = "employee")
```

```
public class Employee {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private int id;
```

```
    private String name;
```

```
    private double salary;
```

```
    public Employee() {}
```

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

```
        this.name = name;
```

```
        this.salary = salary;
```

```
    }
```

```
    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 double getSalary() {
```

```

        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    @Override
    public String toString() {
        return "Employee{" +
            "id=" + id +
            ", name='" + name + "\" +
            ", salary=" + salary +
            '}';
    }
}

```

HibernateUtil.java;

```

package com.example.hibernate.util;
import com.example.hibernate.model.Employee;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    static {
        try {
            Configuration cfg = new Configuration().configure();
            cfg.addAnnotatedClass(Employee.class);
            sessionFactory = cfg.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}

```

Main.java;

```

package com.example.hibernate;
import com.example.hibernate.dao.EmployeeDAO;
import com.example.hibernate.model.Employee;
public class Main {

```

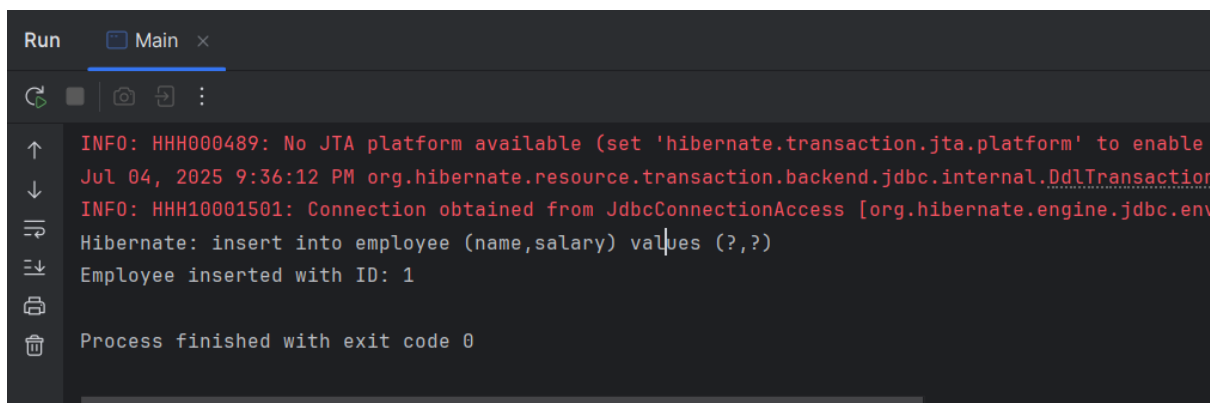
```

public static void main(String[] args) {
    Employee emp = new Employee();
    emp.setName("Lalit");
    emp.setSalary(60000);

    EmployeeDAO dao = new EmployeeDAO();
    Integer id = dao.addEmployee(emp);
    System.out.println("Employee inserted with ID: " + id);
}
}

```

IDE Output:



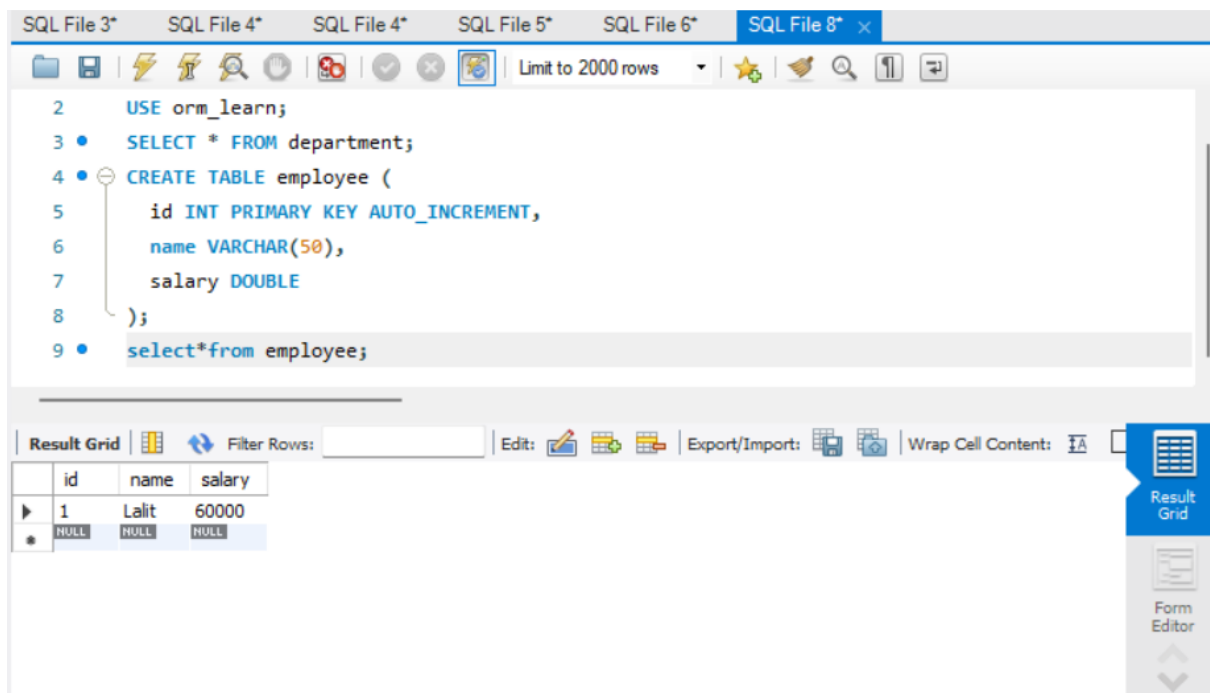
The screenshot shows the Run Output window of an IDE. The output text is as follows:

```

INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable
Jul 04, 2025 9:36:12 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransaction
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env
Hibernate: insert into employee (name,salary) values (?,?)
Employee inserted with ID: 1
Process finished with exit code 0

```

MySQL Output:



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```

2  USE orm_learn;
3  • SELECT * FROM department;
4  • CREATE TABLE employee (
5      id INT PRIMARY KEY AUTO_INCREMENT,
6      name VARCHAR(50),
7      salary DOUBLE
8  );
9  • select*from employee;

```

Below the SQL editor, the 'Result Grid' is displayed, showing the results of the last query. The grid has columns 'id', 'name', and 'salary'.

	id	name	salary
▶	1	Lalit	60000
*	NULL	NULL	NULL

2) Spring Data JPA Implementation;

OrmLearnApplication.java(Main class);

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

```
import com.cognizant.orm_learn.model.Department;
import com.cognizant.orm_learn.repository.DepartmentRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
```

```
@SpringBootApplication
```

```
public class OrmLearnApplication implements CommandLineRunner {
```

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

```
    @Autowired
```

```
    private DepartmentRepository departmentRepository;
```

```
    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
        LOGGER.info("Inside main");
    }
```

```
    @Override
```

```
    public void run(String... args) throws Exception {
        Department dept = new Department("HR");
        departmentRepository.save(dept);
        LOGGER.info("Saved Department: " + dept.getName());
    }
}
```

Department.java;

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

```
import jakarta.persistence.*;
```

```
@Entity
```

```

@Table(name = "department")
public class Department {

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

    private String name;

    public Department() {}
    public Department(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

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

```

DepartmentRepository.java;

```

package com.cognizant.orm_learn.repository;

import com.cognizant.orm_learn.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {
}

```

Sql code;

Create database orm_learn;

Use database orm_learn;

Select*from Department;

Output:

The screenshot shows a SQL IDE interface with multiple tabs at the top labeled 'SQL File 3*' through 'SQL File 8*'. The active tab is 'SQL File 8*'. Below the tabs is a toolbar with various icons, including a 'Limit to 2000 rows' dropdown. The main editor area contains the following SQL code:

```
1 • CREATE DATABASE orm_learn;  
2   USE orm_learn;  
3 • SELECT * FROM department;  
4
```

Below the editor is a 'Result Grid' section. It includes a 'Filter Rows:' input field, an 'Edit:' button, and an 'Export/Import:' button. The result grid itself displays the following data:

	id	name
▶	1	HR
*	NULL	NULL

On the right side of the result grid, there is a vertical toolbar with buttons for 'Result Grid' (highlighted in blue), 'Form Editor', and navigation arrows. At the bottom of the window, there is a status bar with the text 'department 1 x' and two buttons: 'Apply' and 'Revert'.