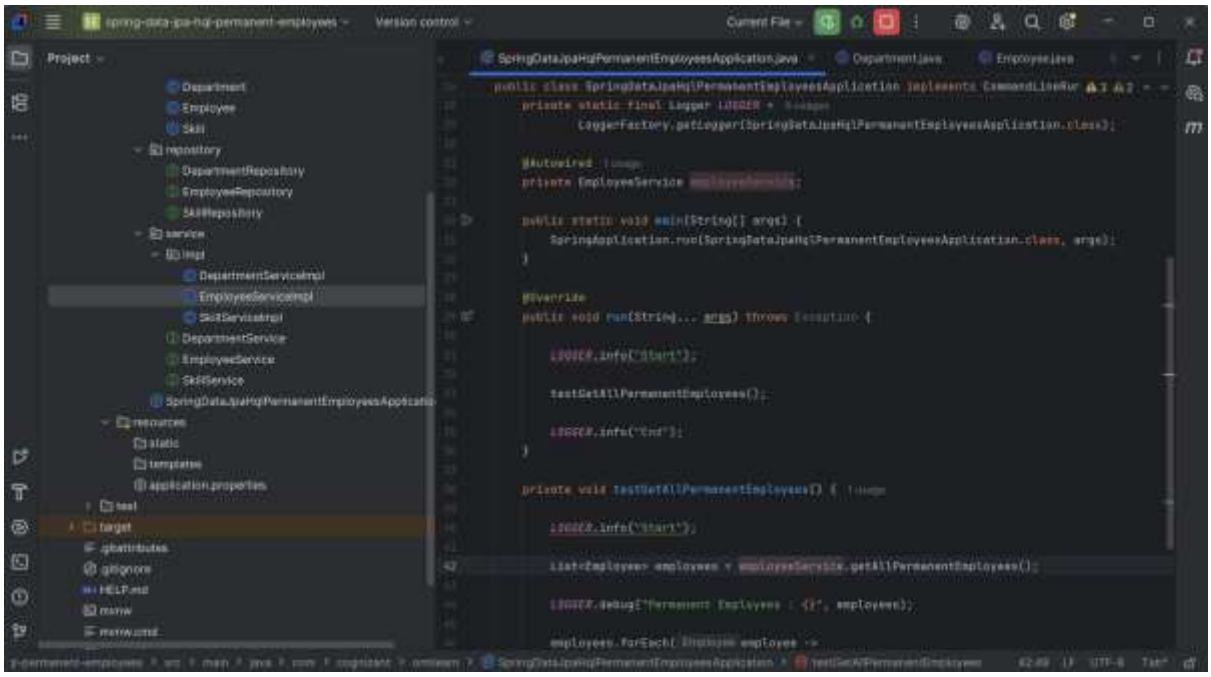
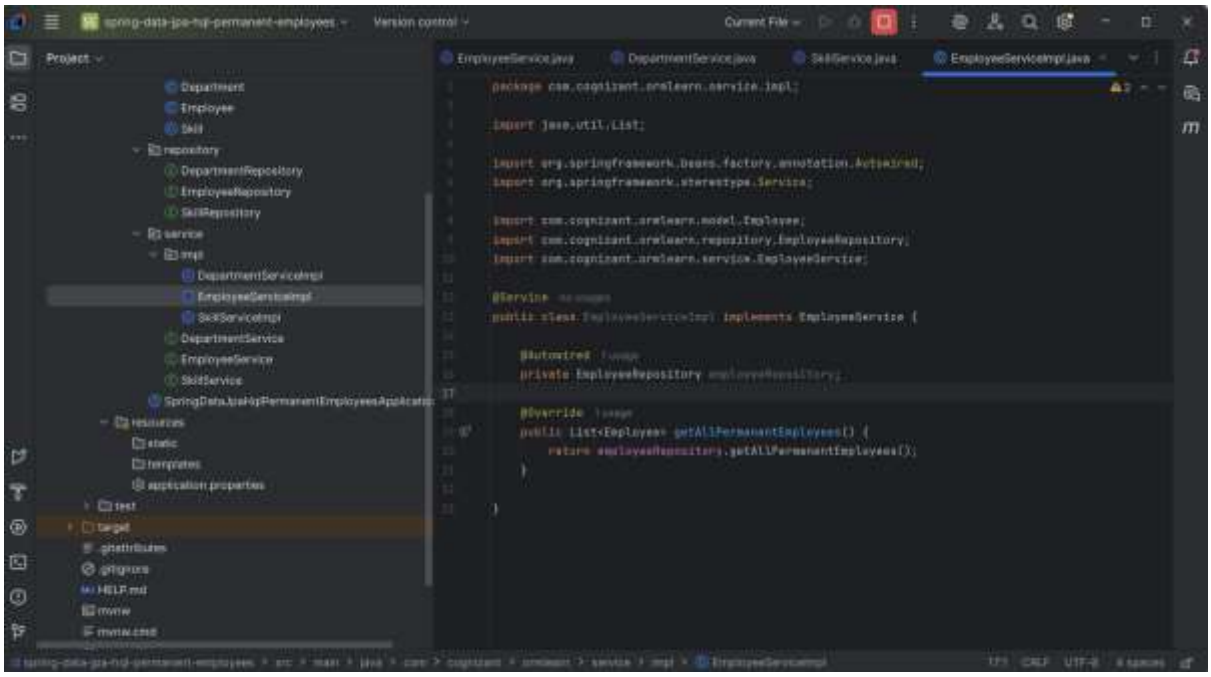
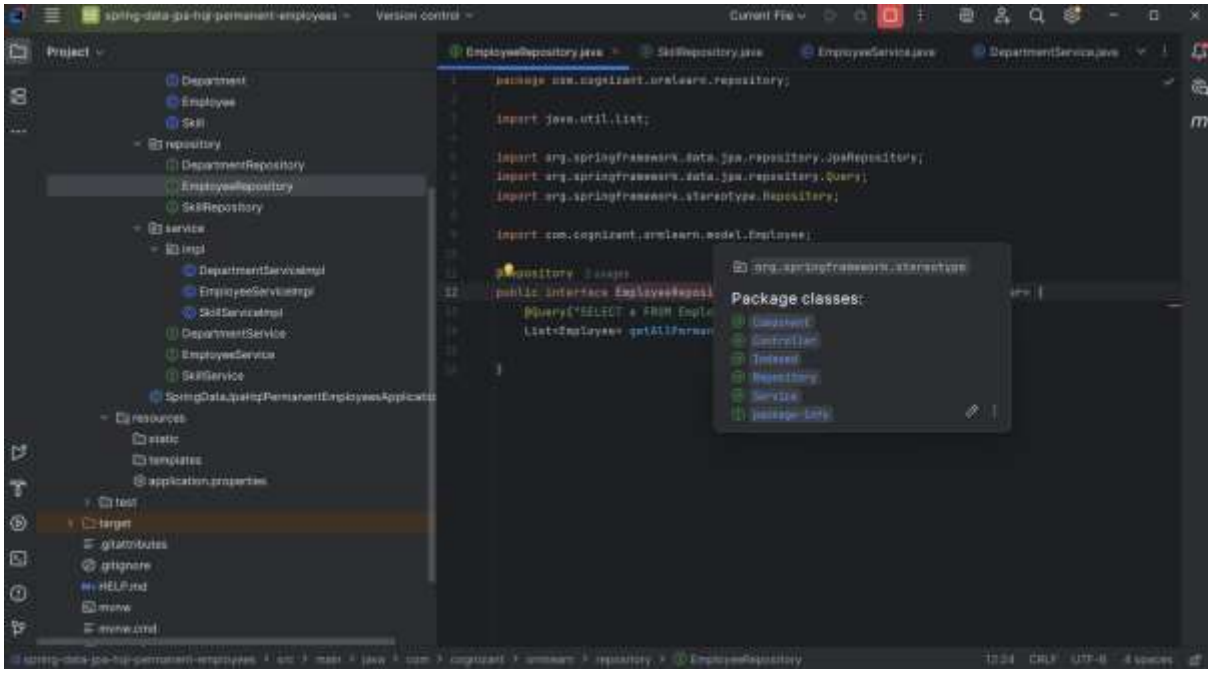
Hands on 2

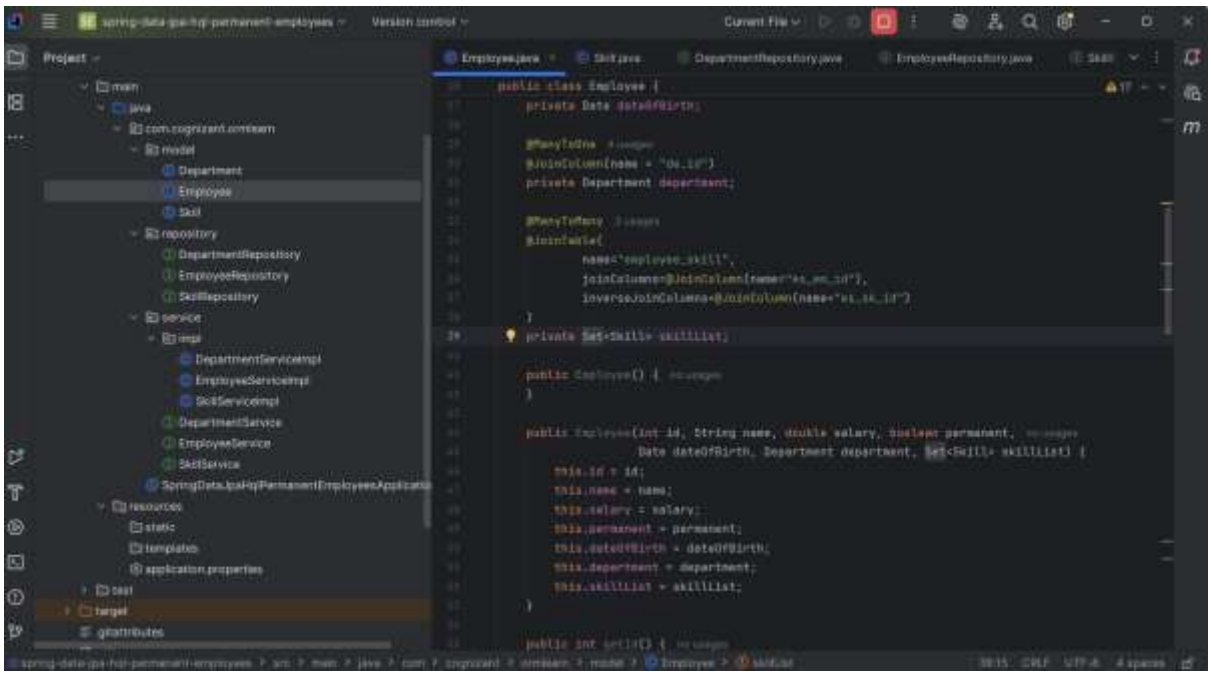
Get all permanent employees using HQL

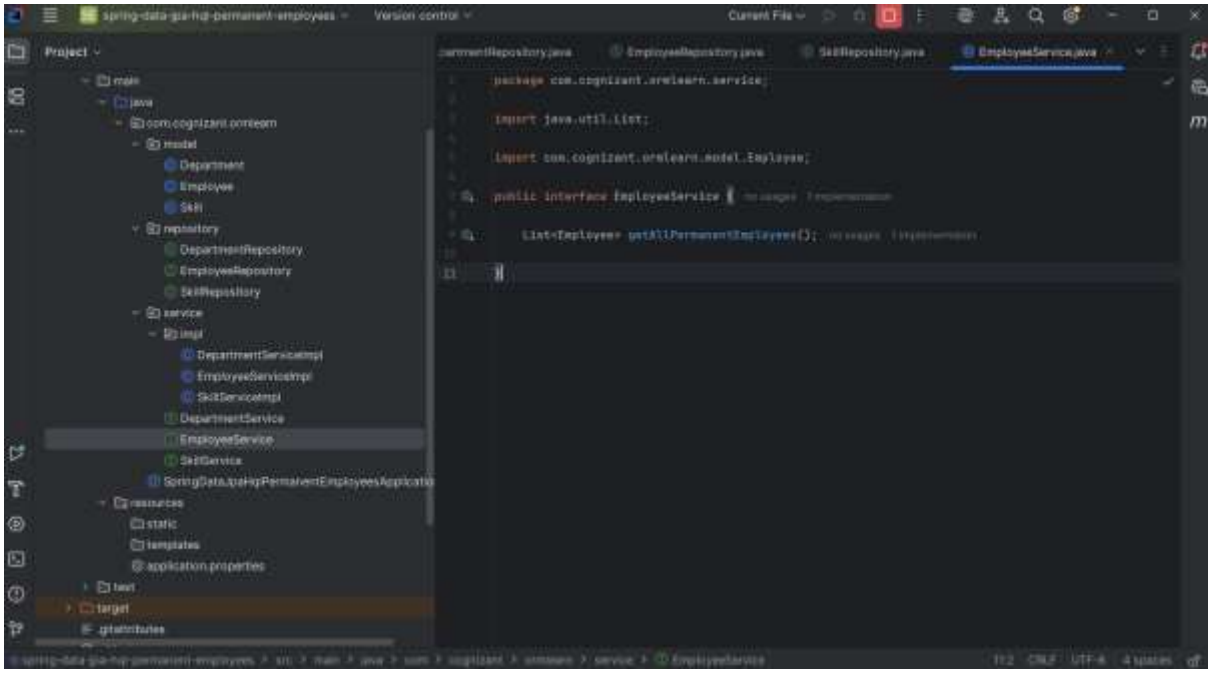
Step 1: Basic HQL query



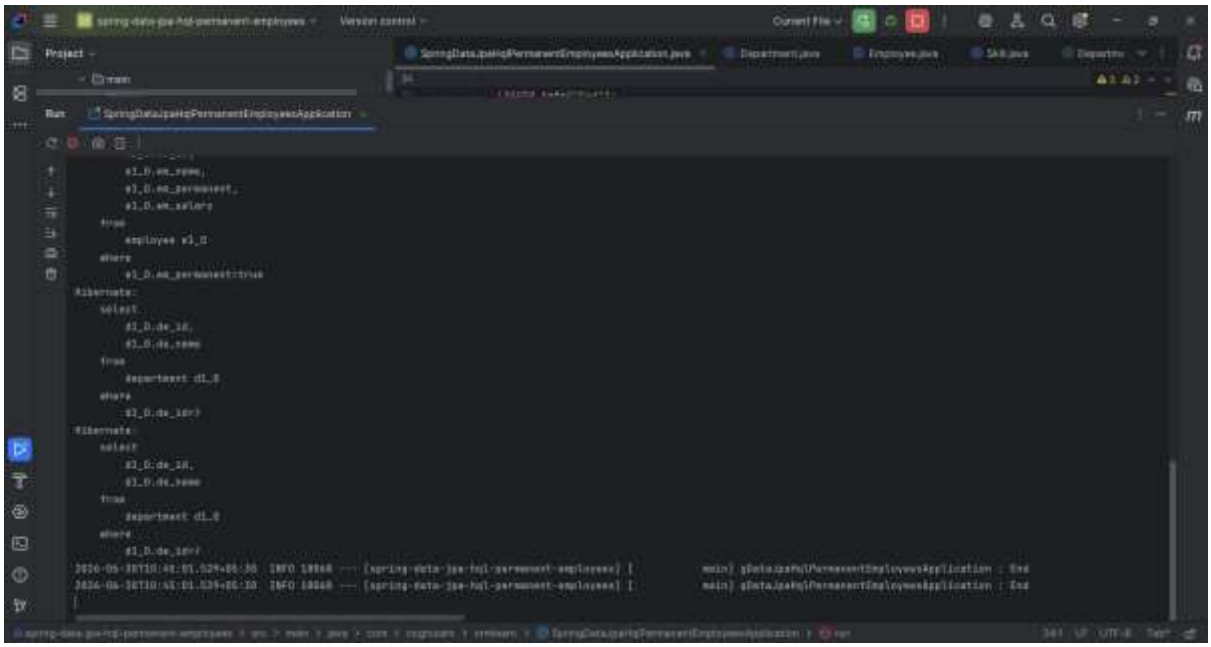



spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
EmployeeRepository.java
SkillRepository.java
EmployeeService.java
DepartmentService.java
package com.cognizant.orelearn.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import com.cognizant.orelearn.model.Employee;
@Repository
public interface EmployeeRepos
@Query("SELECT e FROM Empl
List<Employee> getAllPerman
}
org.springframework.stereotype
Package classes:
Component
Controller
Indexed
Repository
Service
package-info

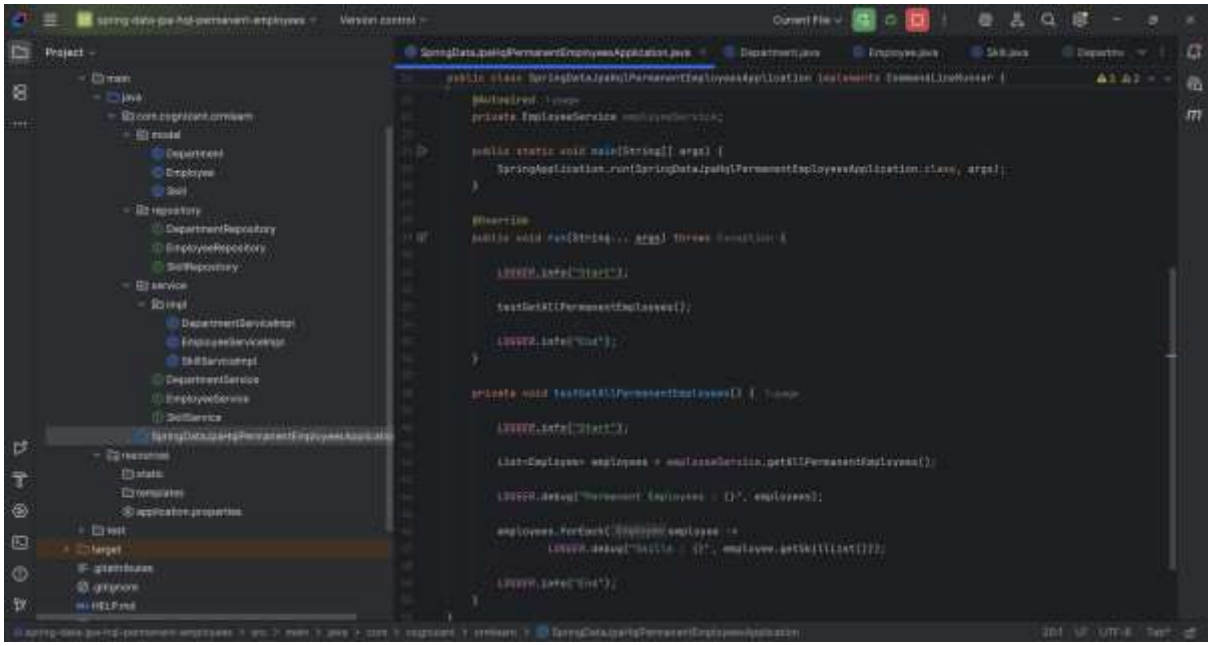

spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
main
java
com.cognizant.orelearn
model
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
resources
static
templates
application.properties
test
target
.gitattributes
Employee.java
Skill.java
DepartmentRepository.java
EmployeeRepository.java
public class Employee {
private Date dateOfBirth;
@ManyToOne
@JoinColumn(name = "de_id")
private Department department;
@ManyToMany
@JoinTable(
name="employee_skill",
joinColumns=@JoinColumn(name="es_em_id"),
inverseJoinColumns=@JoinColumn(name="es_sk_id")
)
private Set<Skill> skillList;
public Employee() {
}
public Employee(int id, String name, double salary, boolean permanent,
Date dateOfBirth, Department department, Set<Skill> skillList) {
this.id = id;
this.name = name;
this.salary = salary;
this.permanent = permanent;
this.dateOfBirth = dateOfBirth;
this.department = department;
this.skillList = skillList;
}
public int getId() {

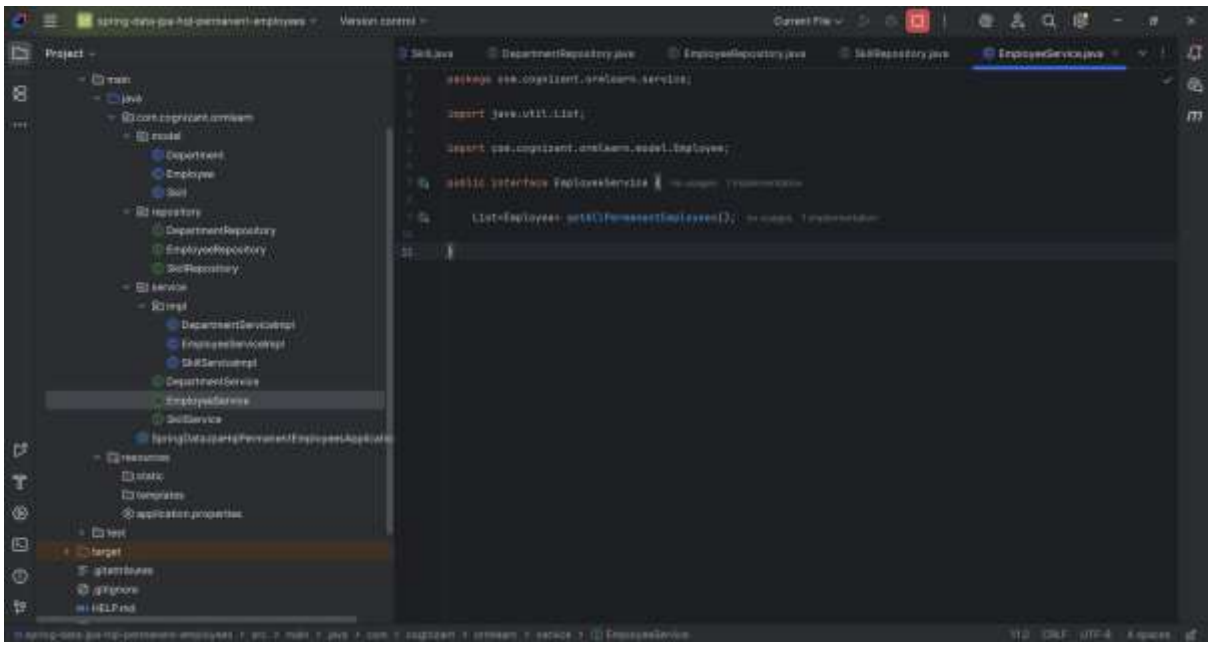


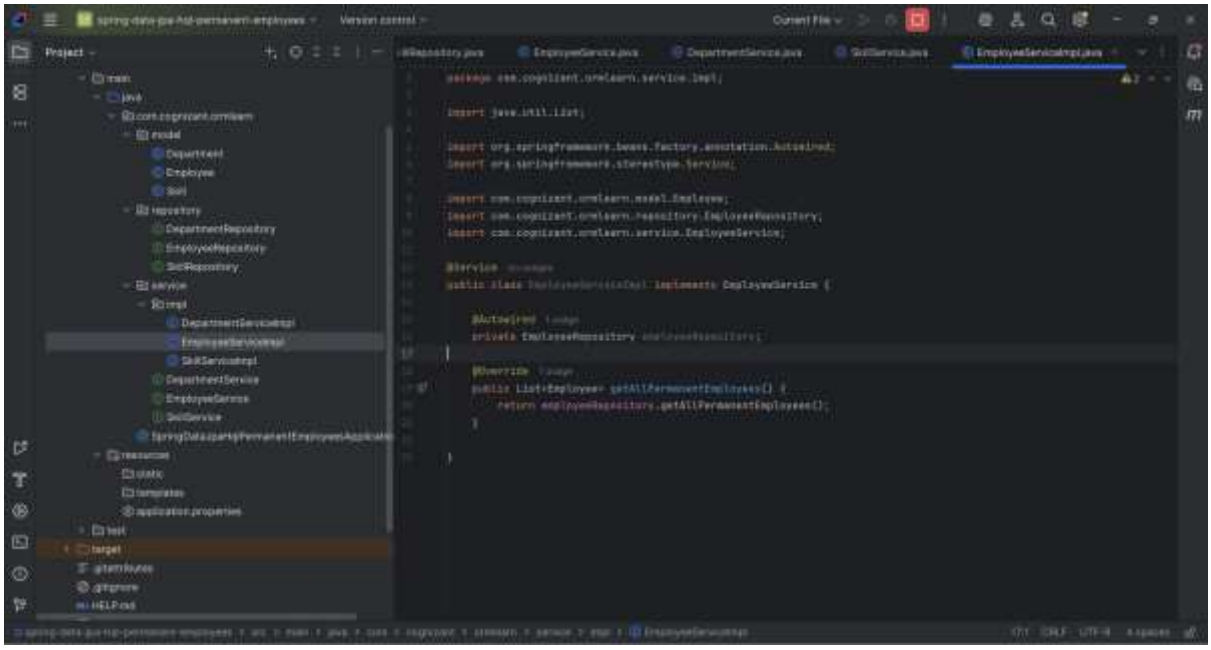
OUTPUT:

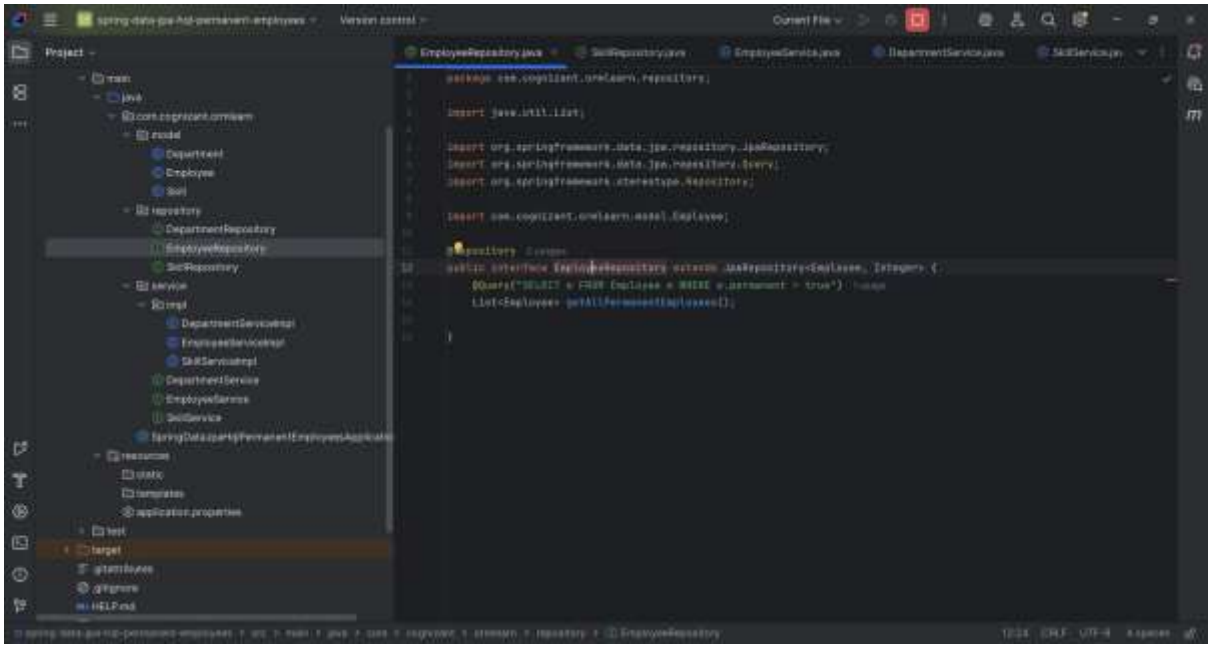


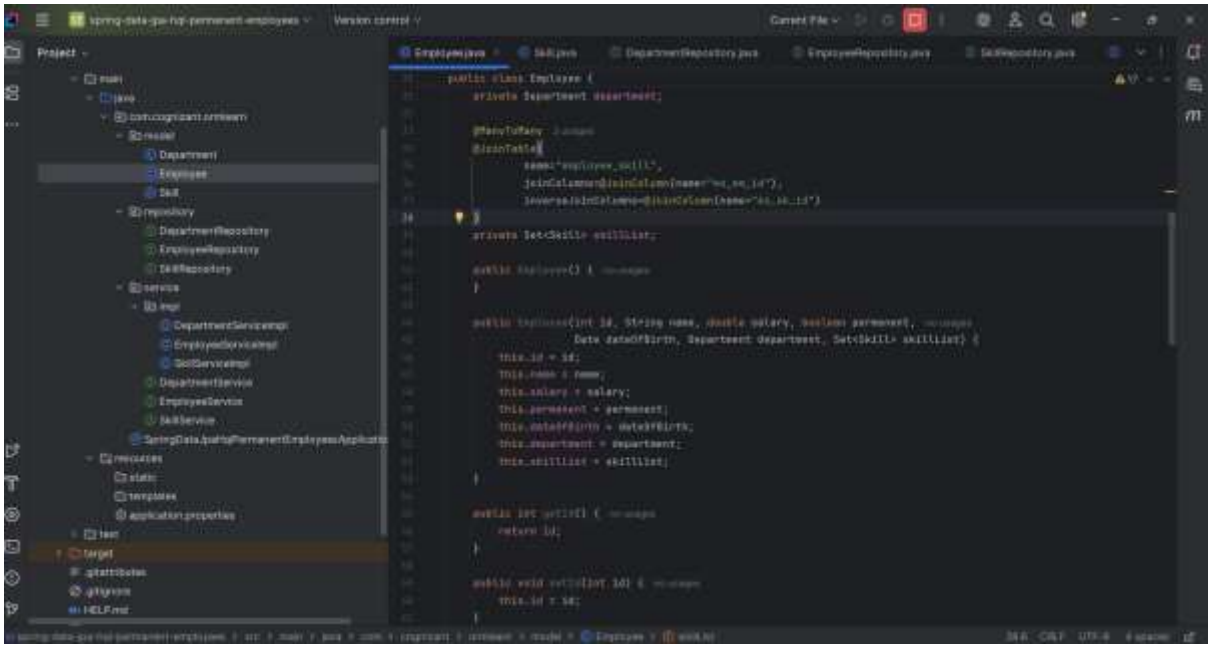
Step 2: Remove EAGER fetch configuration

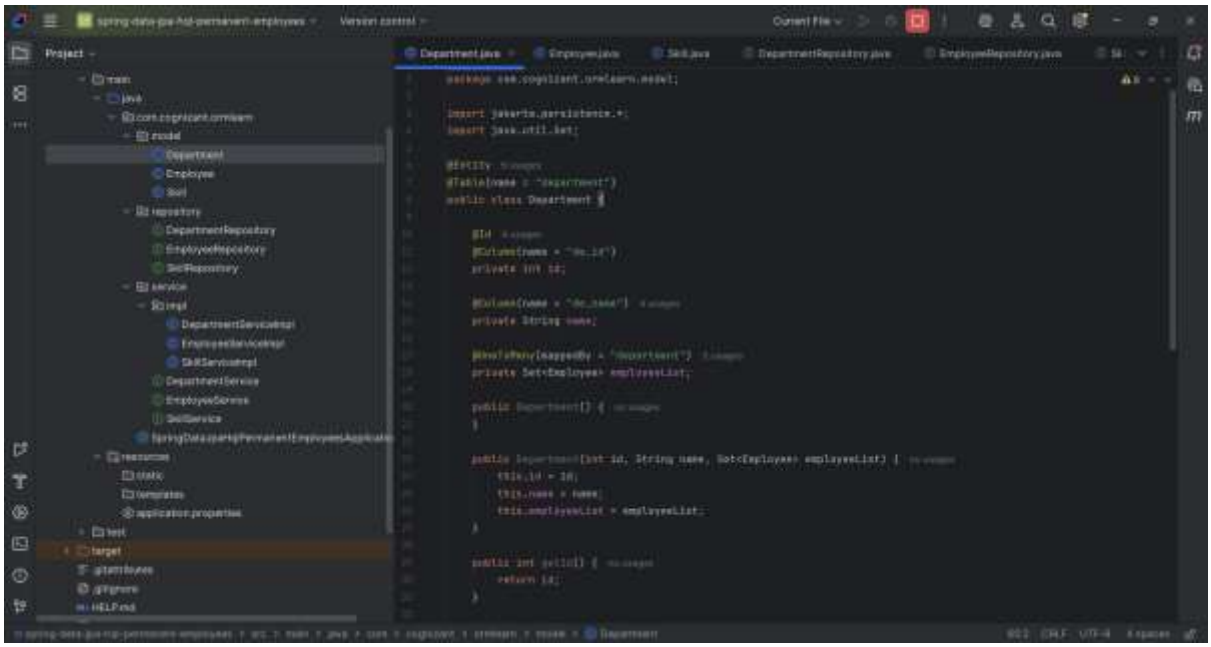


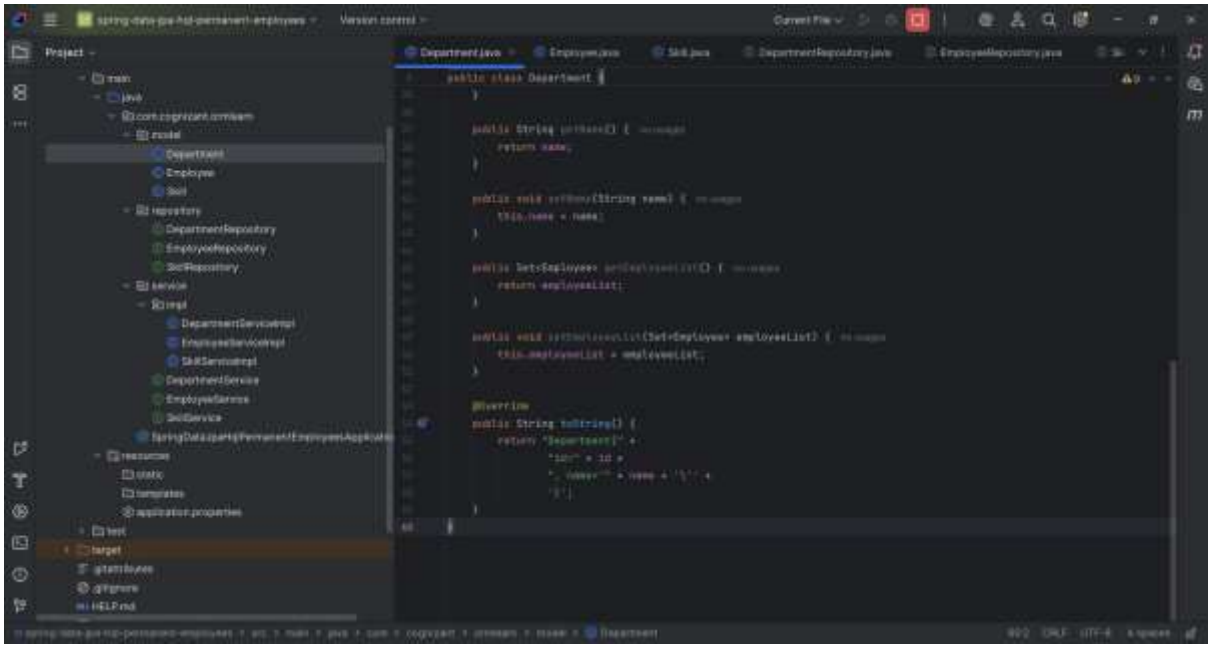


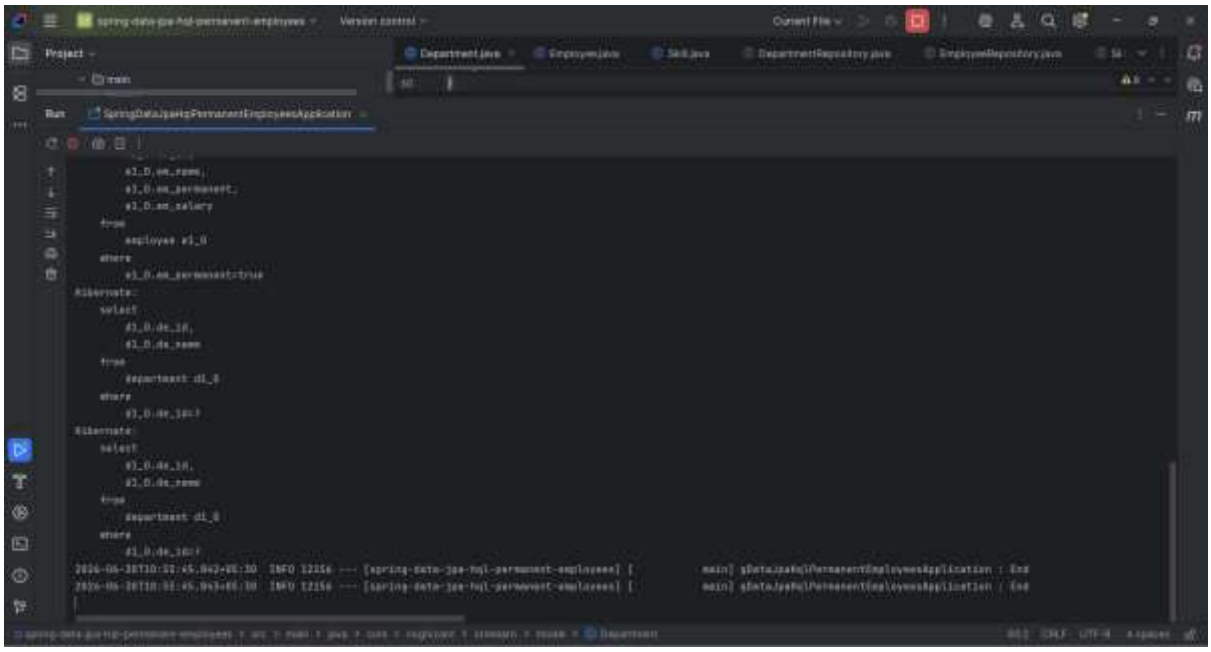

spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
EmployeeRepository.java
EmployeeService.java
DepartmentService.java
SkillService.java
EmployeeServiceImpl.java
main
java
com.cognizant.onelearn
model
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
SpringDataJpaHqlPermanentEmployeesApplicatio
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
package com.cognizant.onelearn.service.impl;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.cognizant.onelearn.model.Employee;
import com.cognizant.onelearn.repository.EmployeeRepository;
import com.cognizant.onelearn.service.EmployeeService;
@Service
public class EmployeeServiceImpl implements EmployeeService {
@Autowired
private EmployeeRepository employeeRepository;
@Override
public List<Employee> getAllPermanentEmployees() {
return employeeRepository.getAllPermanentEmployees();
}
}


spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
EmployeeRepository.java
SkillRepository.java
EmployeeService.java
DepartmentService.java
SkillService.java
main
java
com.cognizant.onelearn
model
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
SpringDataJpaHqlPermanentEmployeesApplicatio
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
package com.cognizant.onelearn.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import com.cognizant.onelearn.model.Employee;
@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
@Query("SELECT e FROM Employee e WHERE e.permanent = true")
List<Employee> getAllPermanentEmployees();
}

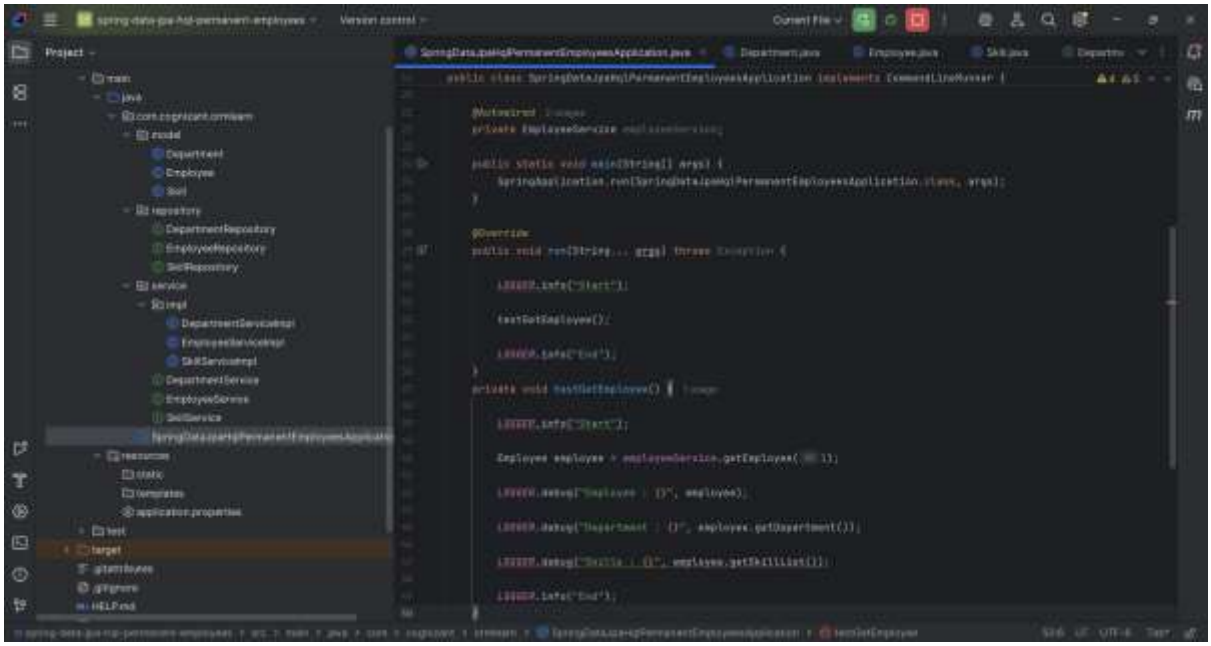


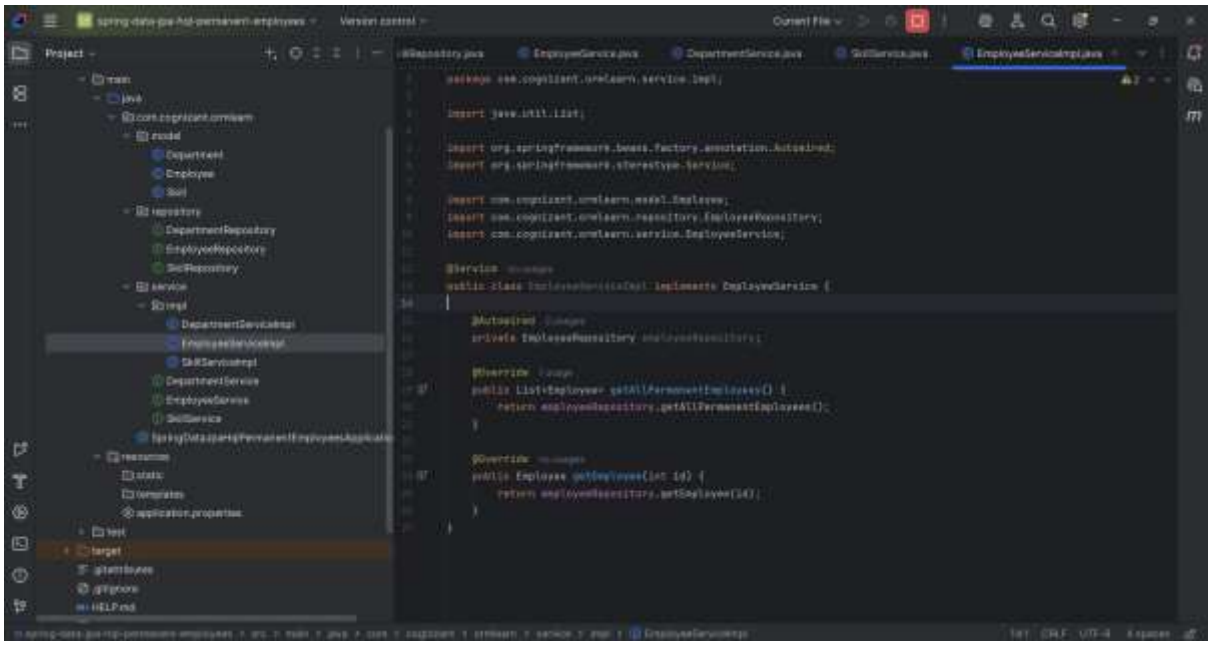


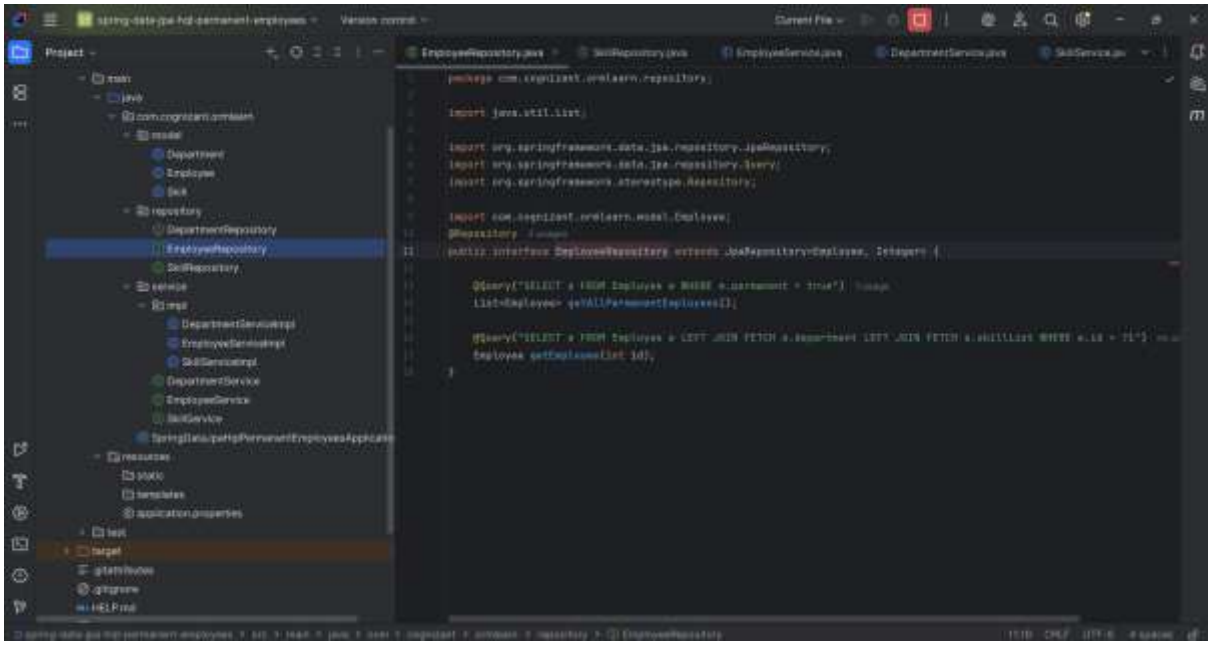


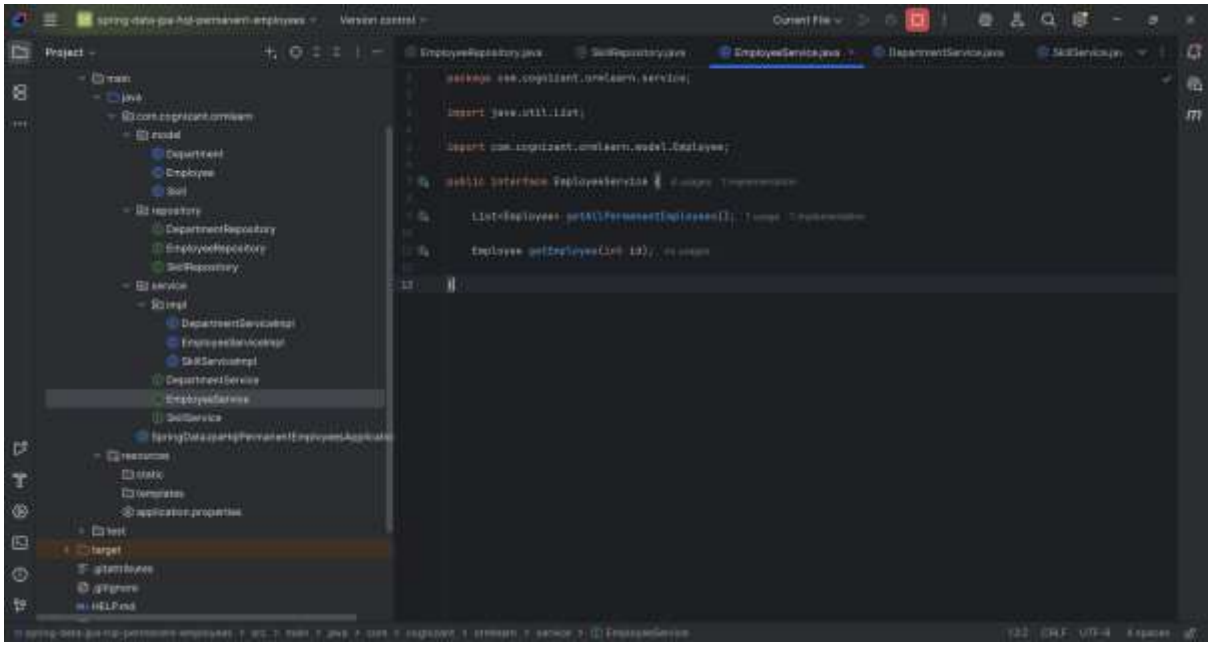


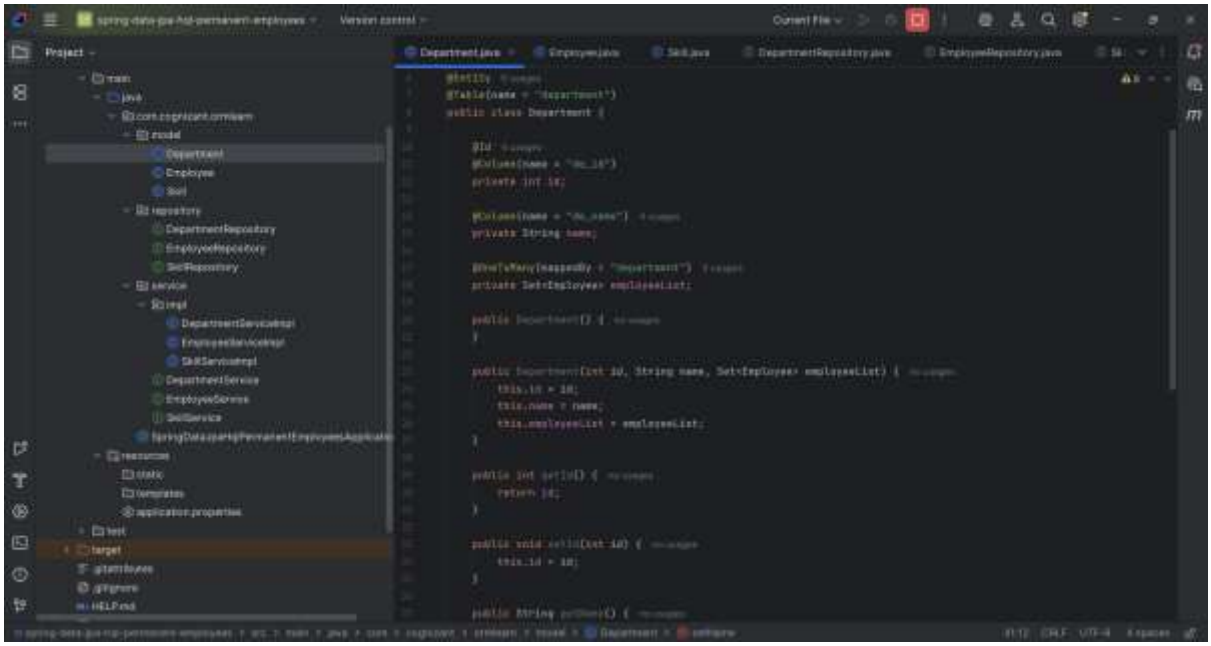
Step 3: Optimize using LEFT JOIN FETCH

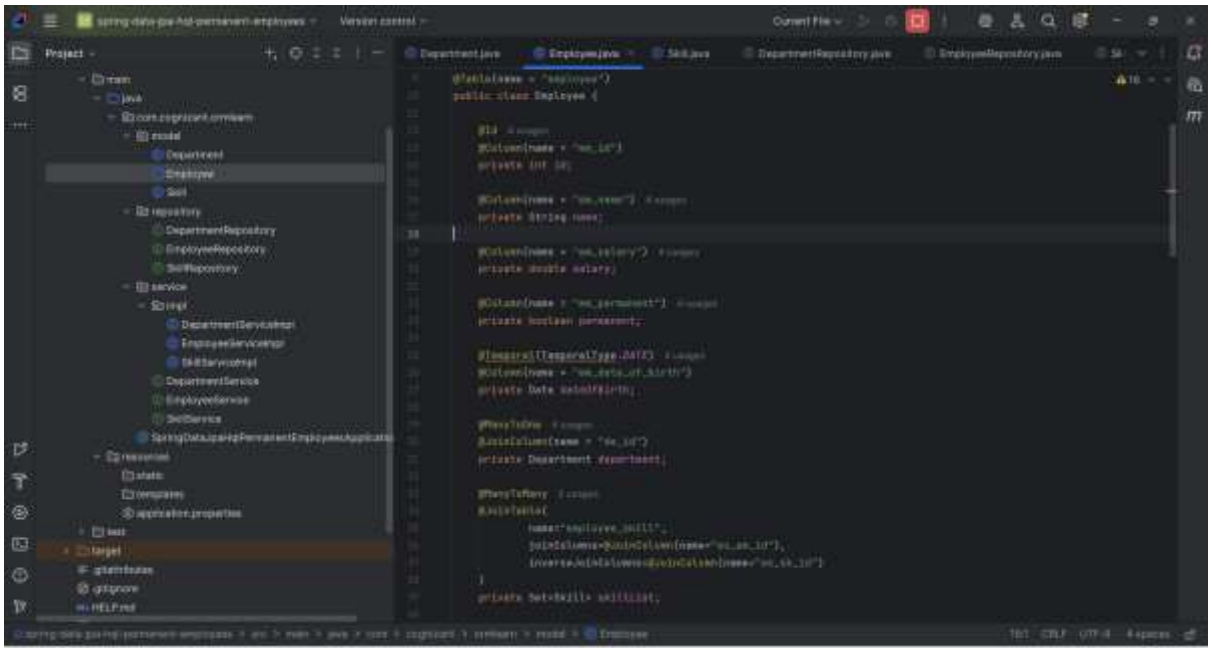


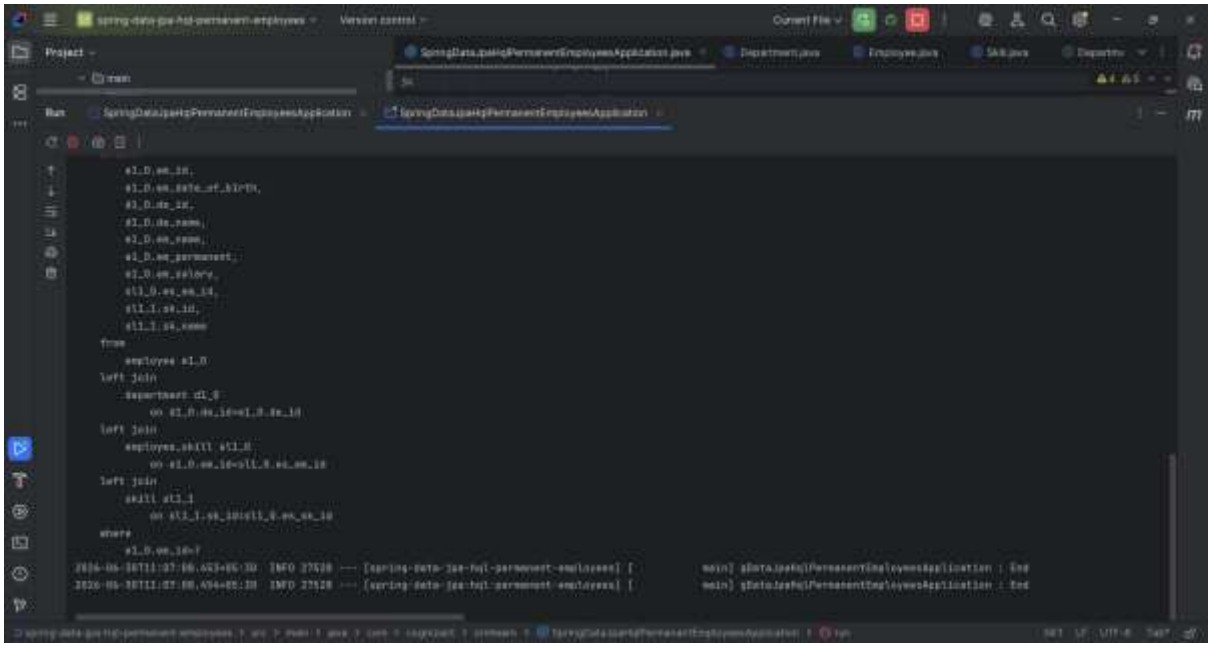



spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
EmployeeRepository.java
SkillRepository.java
EmployeeService.java
DepartmentService.java
SkillService.java
main
java
com.cognizant.orglearn
model
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
SpringData.JpaHqlPermanentEmployeesApplicatio
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
package com.cognizant.orglearn.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import com.cognizant.orglearn.model.Employee;
@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
@Query("SELECT e FROM Employee e WHERE e.permanent = true")
List<Employee> getAllPermanentEmployees();
@Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department LEFT JOIN FETCH e.skillList WHERE e.id = ?1")
Employee getEmployee(int id);
}


spring-data-jpa-hql-permanent-employees
Version control
Current File
Project
EmployeeRepository.java
SkillRepository.java
EmployeeService.java
DepartmentService.java
SkillService.java
main
java
com.cognizant.orglearn
model
Department
Employee
Skill
repository
DepartmentRepository
EmployeeRepository
SkillRepository
service
impl
DepartmentServiceImpl
EmployeeServiceImpl
SkillServiceImpl
DepartmentService
EmployeeService
SkillService
SpringData.JpaHqlPermanentEmployeesApplicatio
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
package com.cognizant.orglearn.service;
import java.util.List;
import com.cognizant.orglearn.model.Employee;
public interface EmployeeService {
List<Employee> getAllPermanentEmployees();
Employee getEmployee(int id);
}

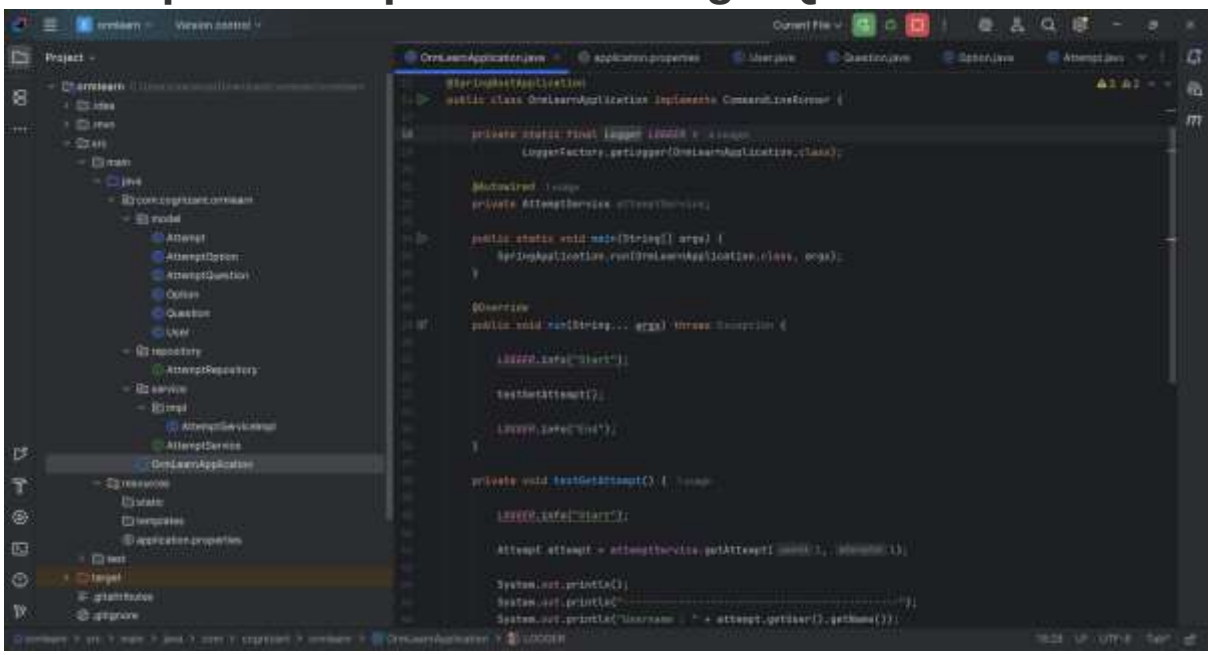


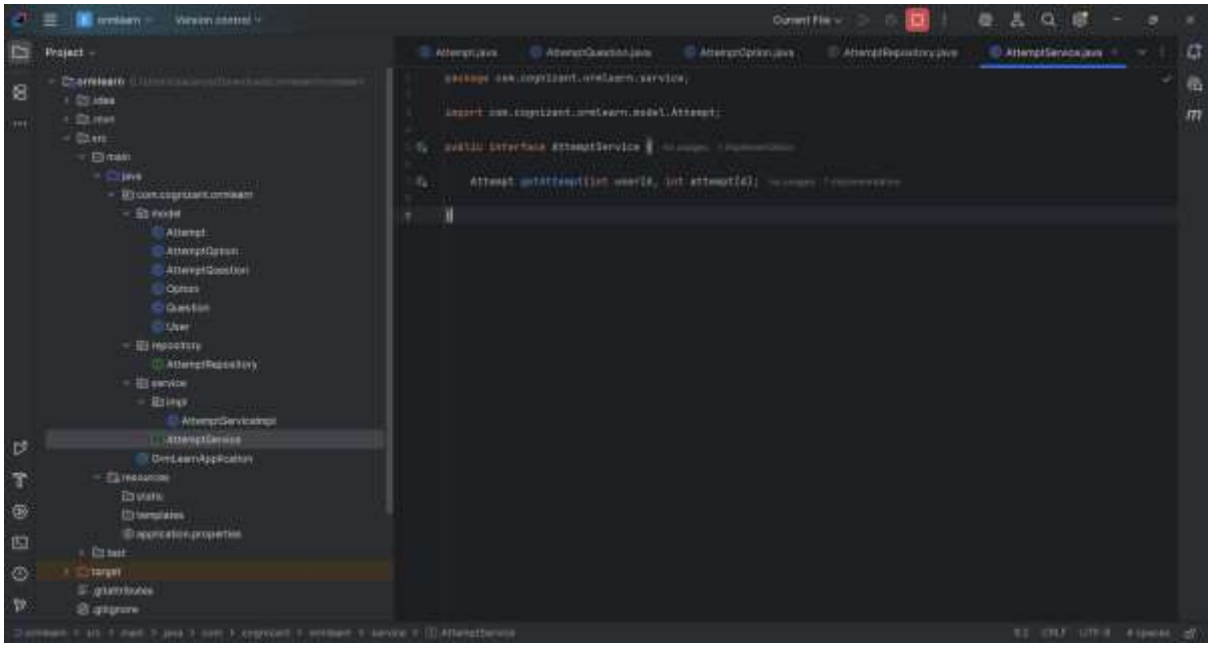


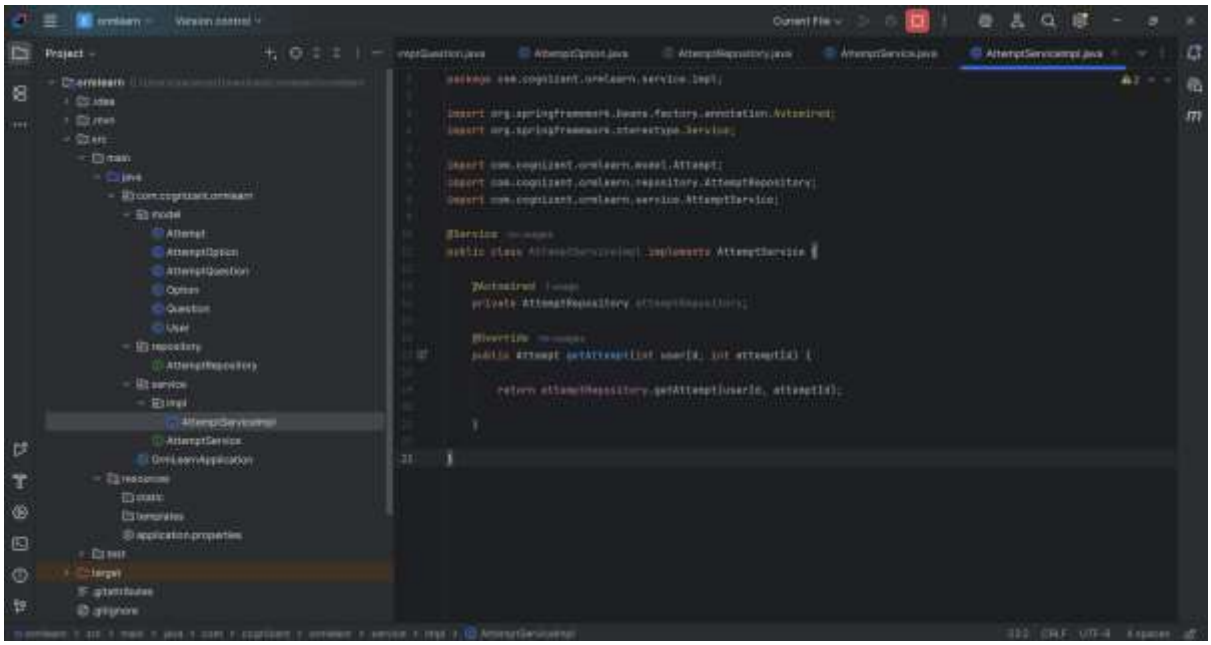


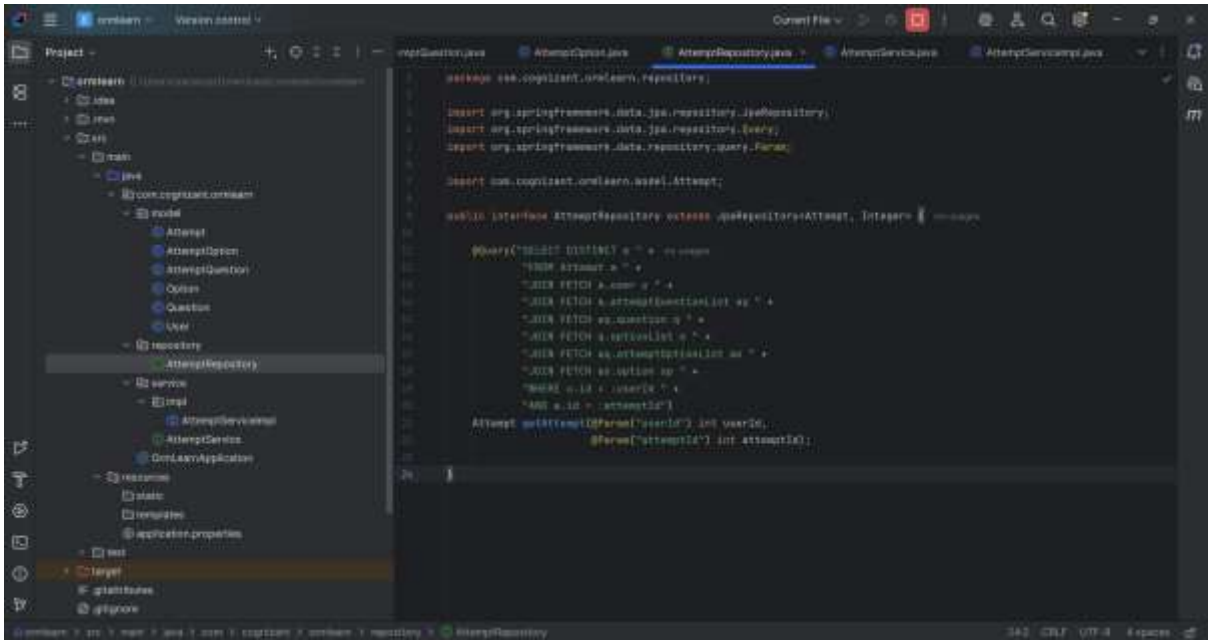
Hands on 3

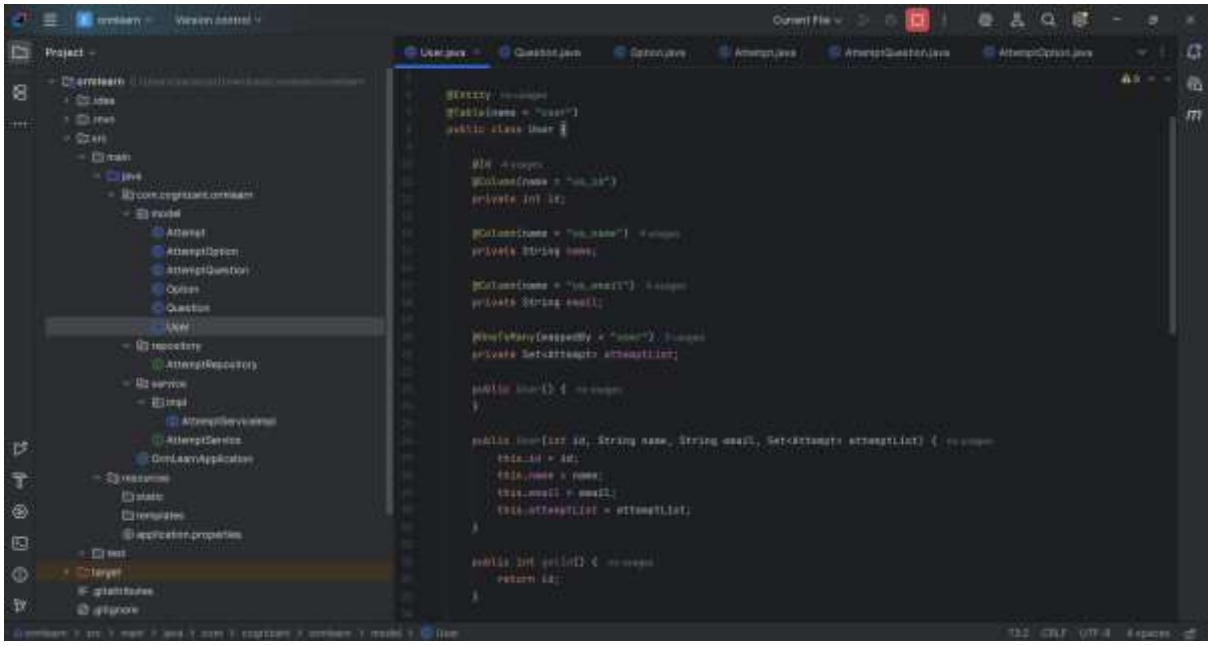
Fetch quiz attempt details using HQL

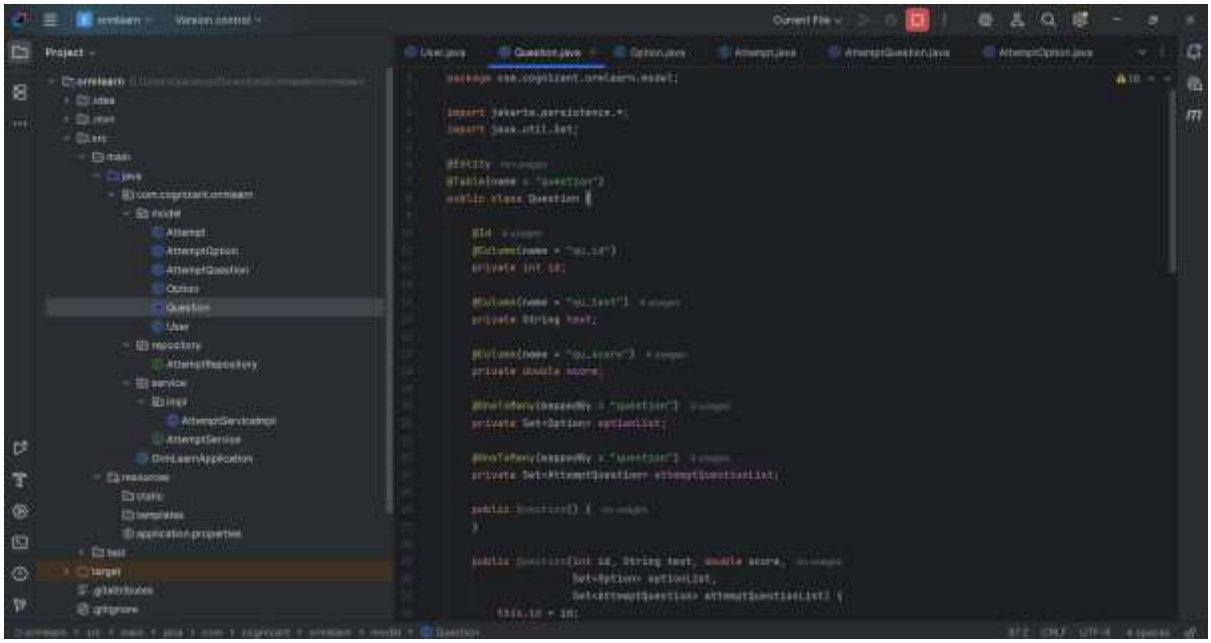


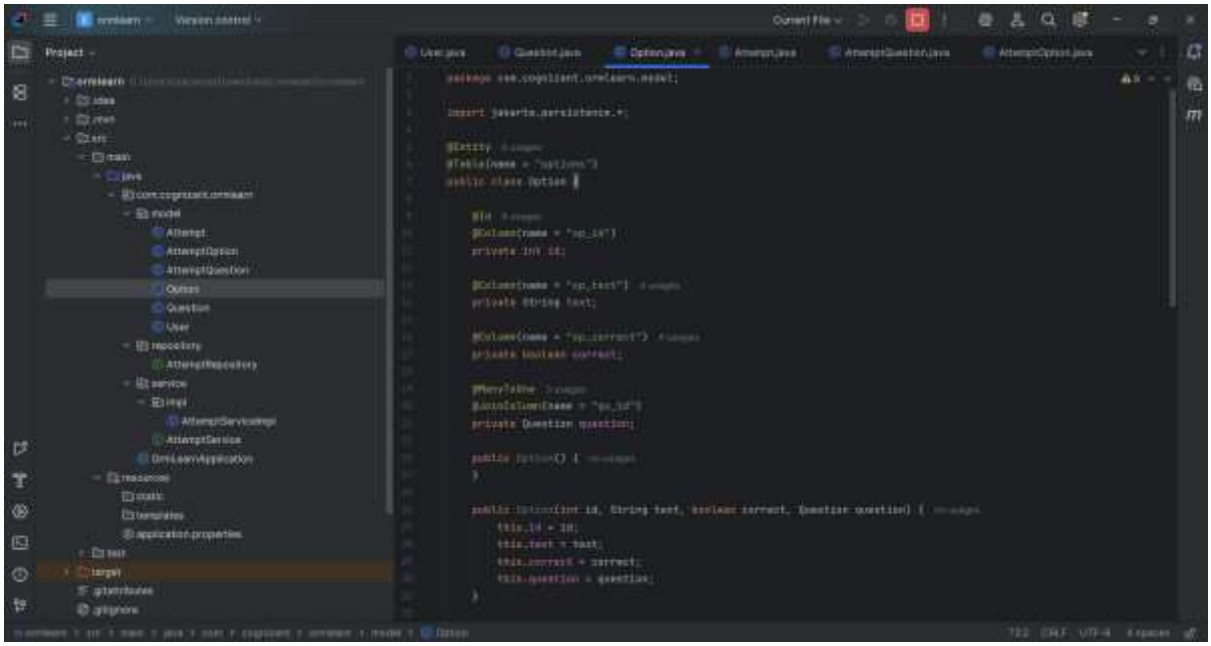

Attempt.java
AttemptQuestion.java
AttemptOption.java
AttempRepository.java
AttemptService.java
package com.cognizant.omlearn.service;
import com.cognizant.omlearn.model.Attempt;
public interface AttemptService {
Attempt getAttempt(int userId, int attemptId);
}

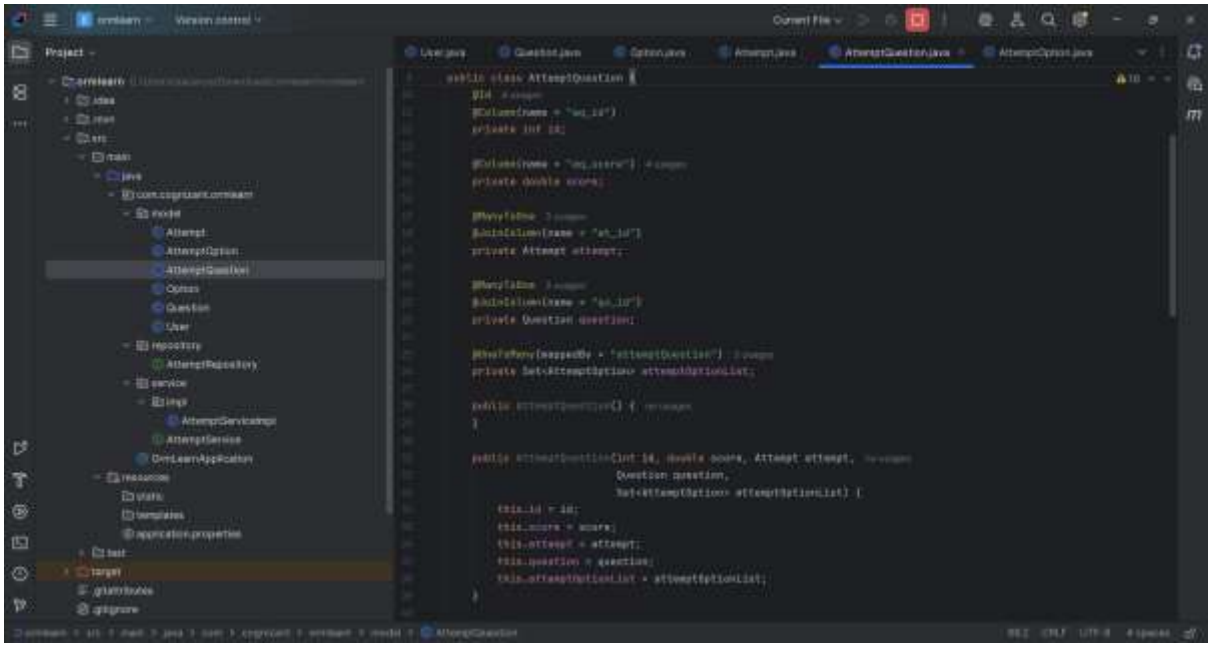

AttemptOption.java
AttempRepository.java
AttemptService.java
AttemptServiceImpl.java
package com.cognizant.omlearn.service.impl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.cognizant.omlearn.model.Attempt;
import com.cognizant.omlearn.repository.AttemptRepository;
import com.cognizant.omlearn.service.AttemptService;
@Service
public class AttemptServiceImpl implements AttemptService {
@Autowired
private AttemptRepository attemptRepository;
@Override
public Attempt getAttempt(int userId, int attemptId) {
return attemptRepository.getAttempt(userId, attemptId);
}
}

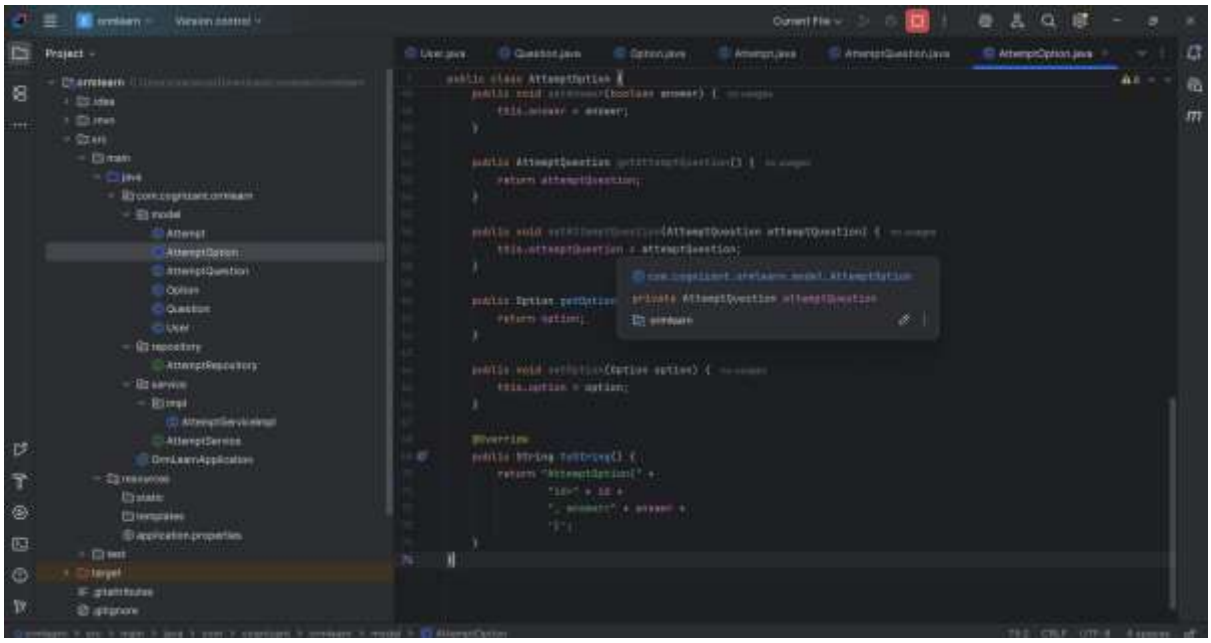

Project
AttemptRepository.java
AttemptOption.java
AttemptService.java
AttemptServiceImpl.java
package com.cognizant.onlearn.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import com.cognizant.onlearn.model.Attempt;
public interface AttemptRepository extends JpaRepository<Attempt, Integer> {
@Query("SELECT DISTINCT a " +
"FROM Attempt a " +
"JOIN FETCH a.user u " +
"JOIN FETCH a.attemptQuestionList aq " +
"JOIN FETCH aq.question q " +
"JOIN FETCH a.optionList o " +
"JOIN FETCH aq.attemptOptionList ao " +
"JOIN FETCH ao.option op " +
"WHERE u.id = :userId " +
"AND a.id = :attemptId")
Attempt getAttempt(@Param("userId") int userId,
@Param("attemptId") int attemptId);
}

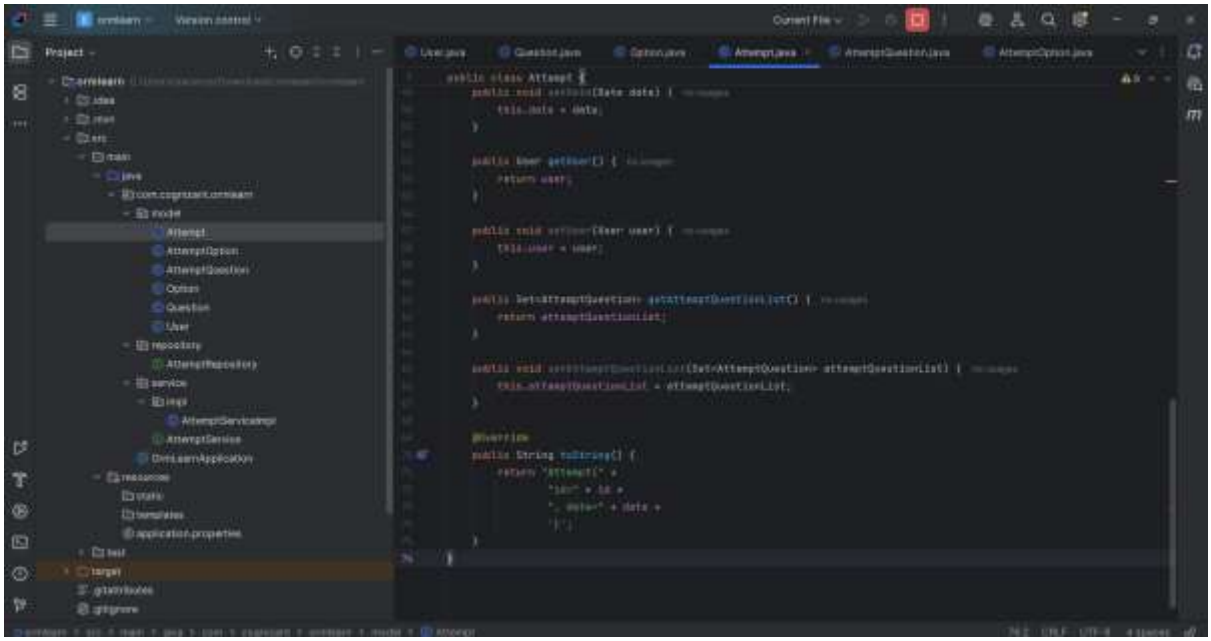

Project
User.java
Question.java
Option.java
Attempt.java
AttemptQuestion.java
AttemptOption.java
@Entity
@Table(name = "user")
public class User {
@Id
@Column(name = "us_id")
private int id;
@Column(name = "us_name")
private String name;
@Column(name = "us_email")
private String email;
@OneToMany(mappedBy = "user")
private Set<Attempt> attemptList;
public User() {
}
public User(int id, String name, String email, Set<Attempt> attemptList) {
this.id = id;
this.name = name;
this.email = email;
this.attemptList = attemptList;
}
public int getId() {
return id;
}



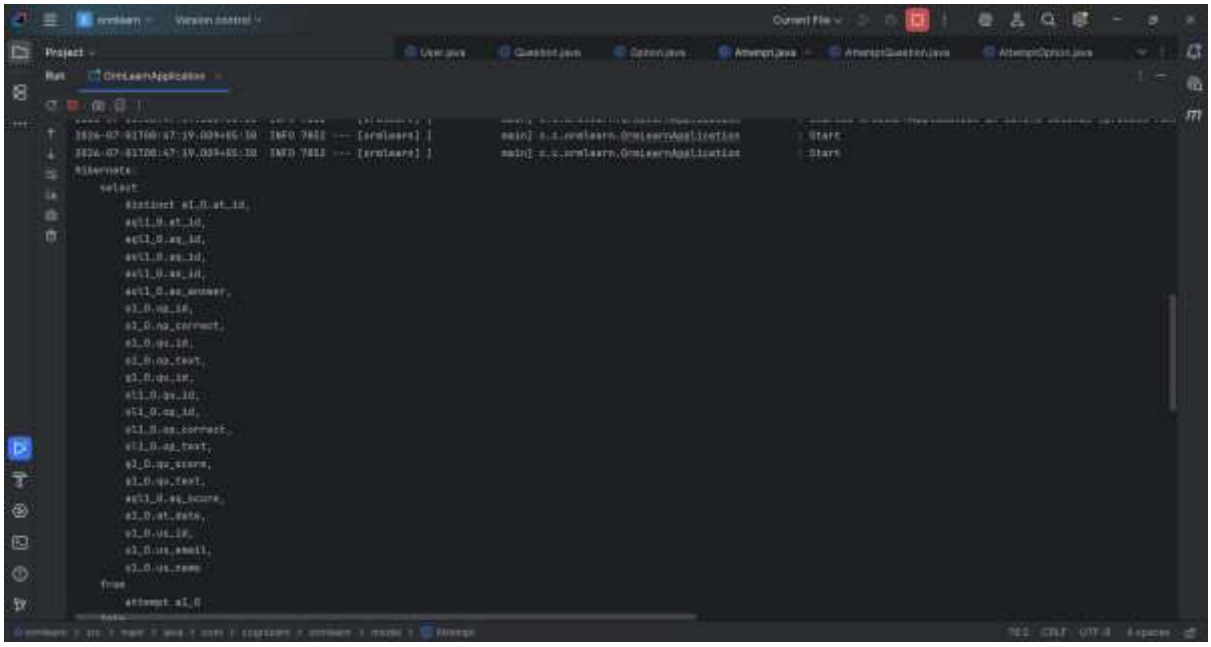


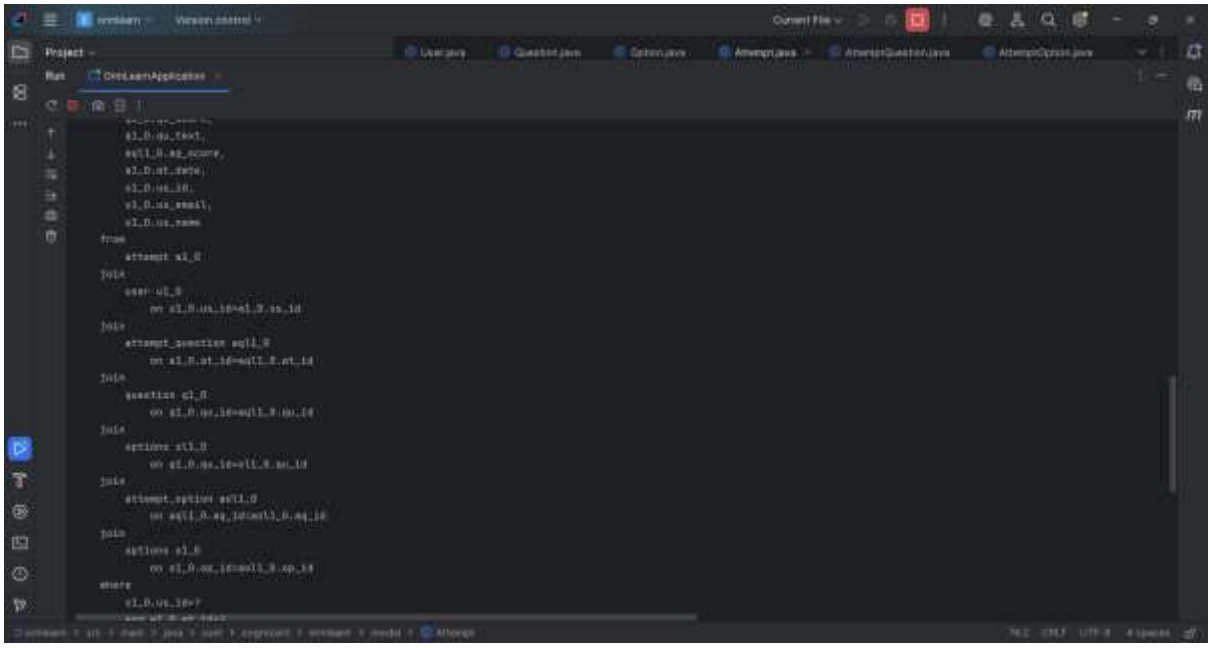


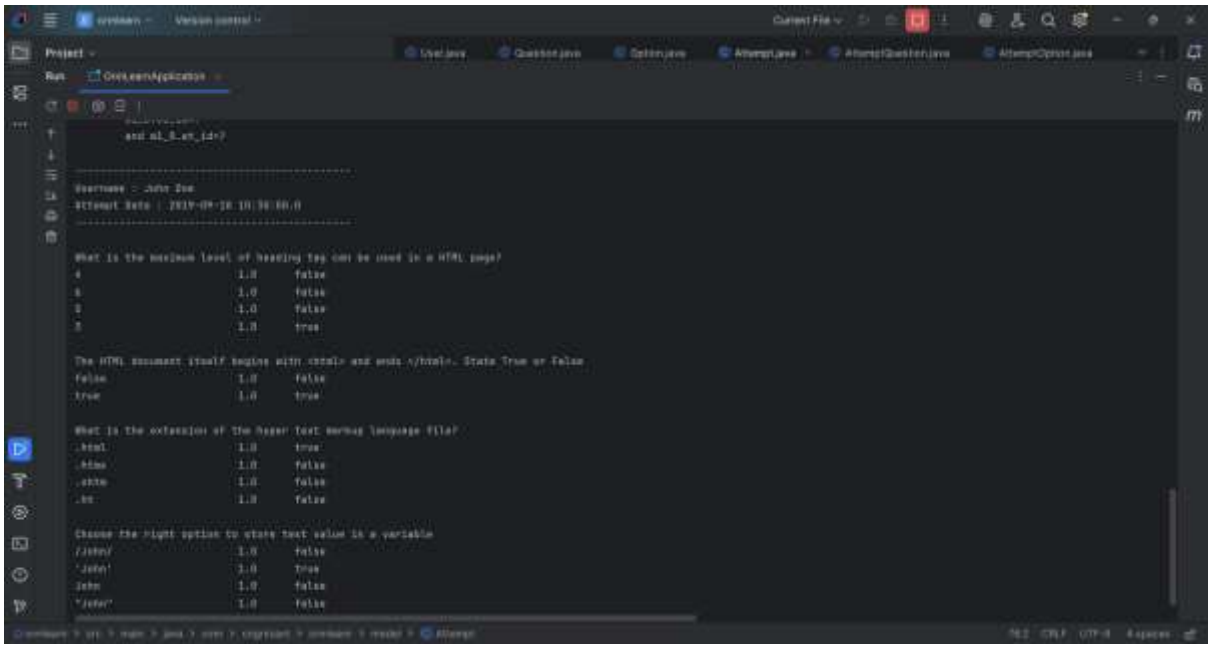




OUTPUT:



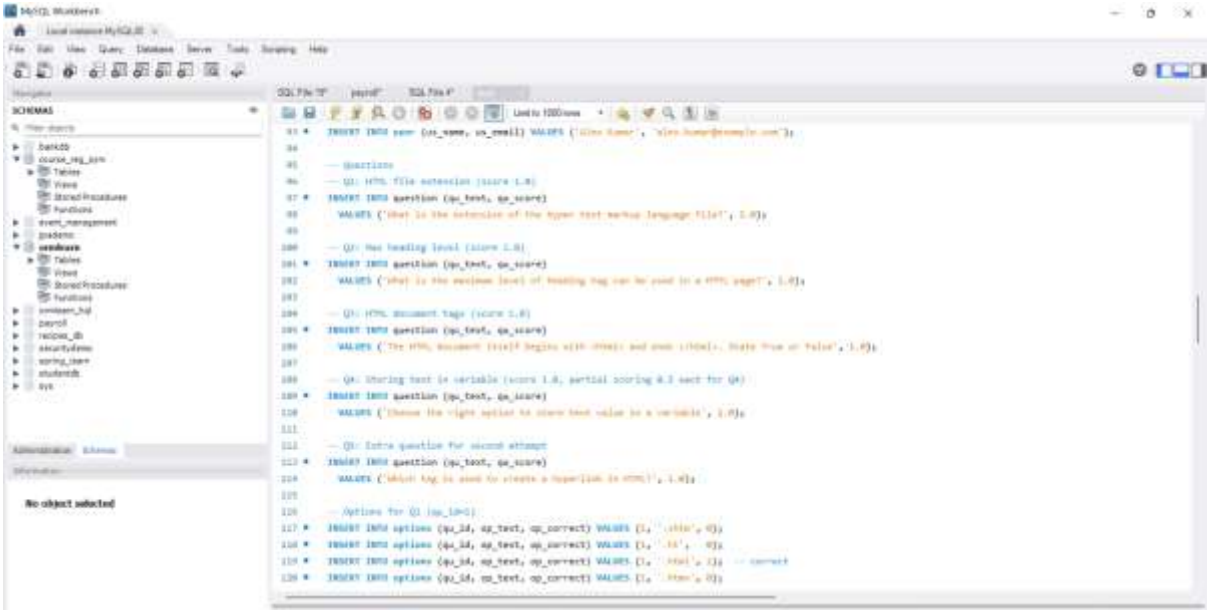


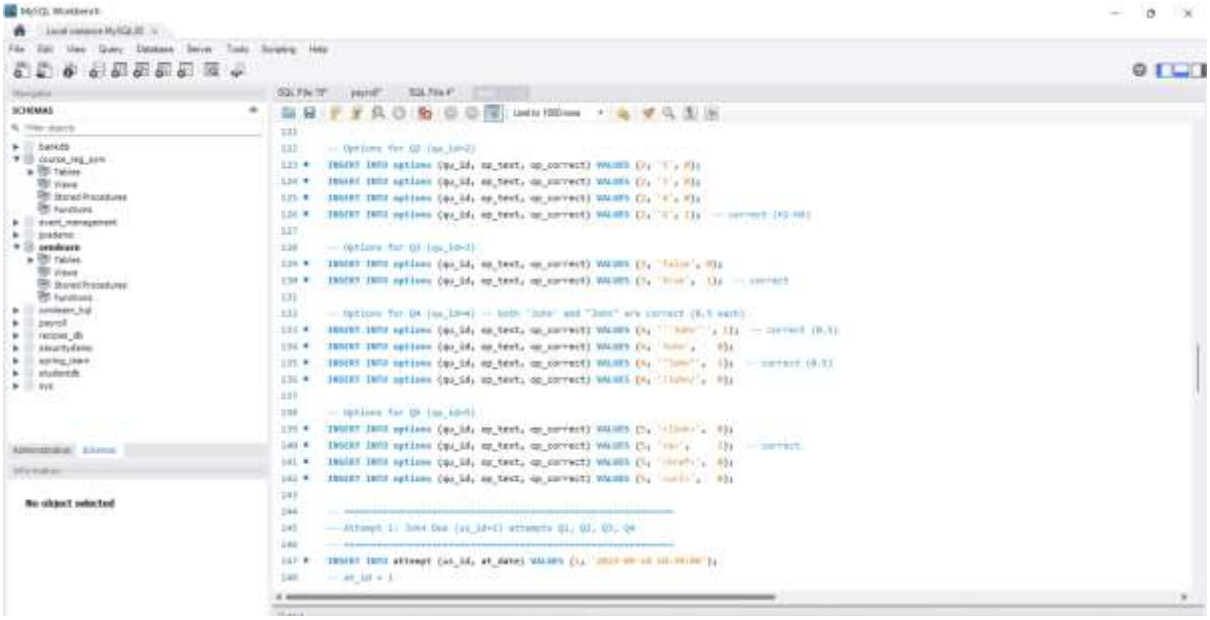


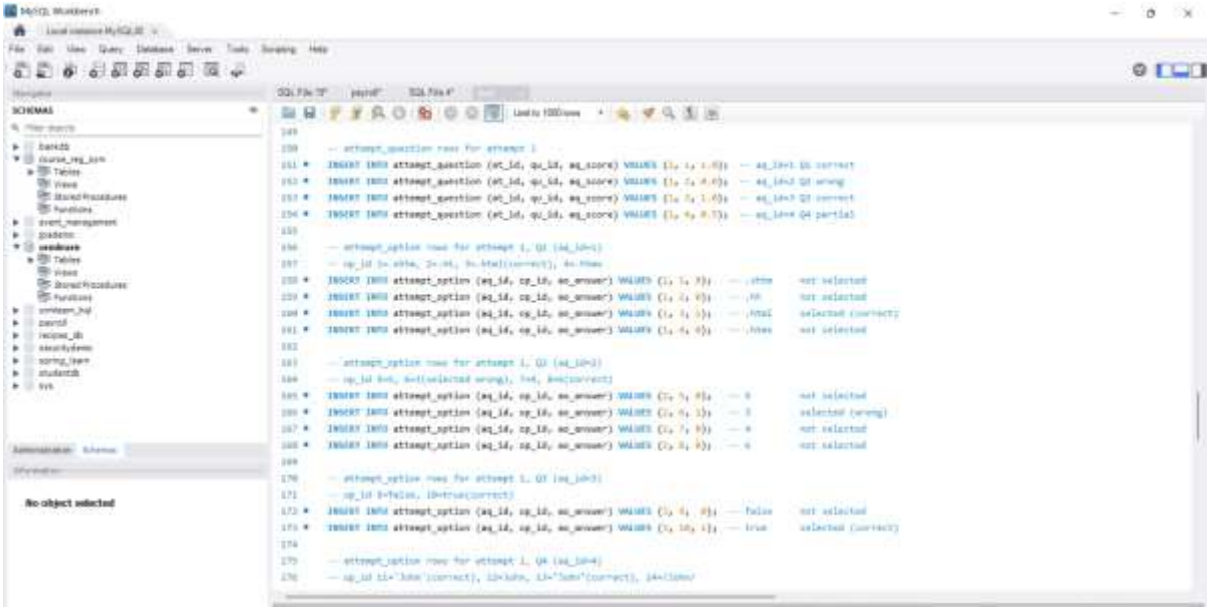
```
-- Drop tables in reverse FK order
DROP TABLE IF EXISTS attempt_option;
DROP TABLE IF EXISTS attempt_question;
DROP TABLE IF EXISTS attempt;
DROP TABLE IF EXISTS options;
DROP TABLE IF EXISTS question;
DROP TABLE IF EXISTS user;

-- ----------------------------------------------------------
-- Table: user
-- ----------------------------------------------------------
CREATE TABLE user (
  us_id      INT          NOT NULL AUTO_INCREMENT,
  us_name    VARCHAR(100) NOT NULL,
  us_email   VARCHAR(100) NOT NULL,
  PRIMARY KEY (us_id)
);

-- ----------------------------------------------------------
-- Table: question
-- ----------------------------------------------------------
CREATE TABLE question (
  qu_id      INT          NOT NULL AUTO_INCREMENT,
  qu_text    VARCHAR(500) NOT NULL,
  qu_score   DECIMAL(4,1) NOT NULL DEFAULT 1.0,
  PRIMARY KEY (qu_id)
);
```

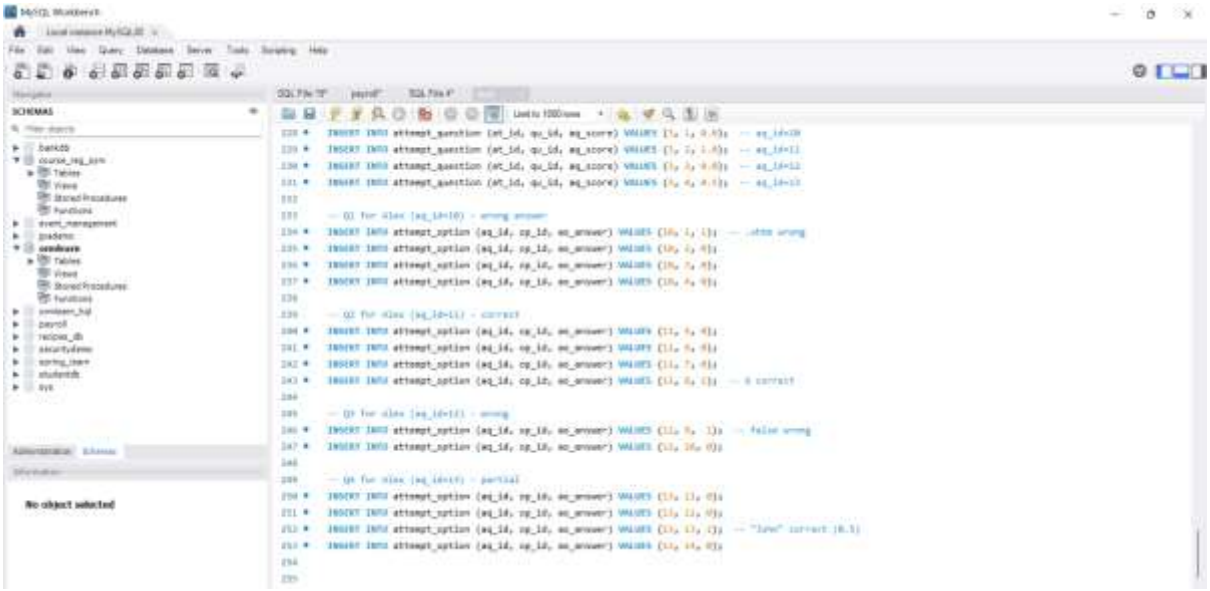
```
  op_id        INT          NOT NULL AUTO_INCREMENT,
  qu_id        INT          NOT NULL,
  op_text      VARCHAR(300) NOT NULL,
  op_correct   TINYINT(1)   NOT NULL DEFAULT 0,
  PRIMARY KEY (op_id),
  CONSTRAINT fk_opt_question FOREIGN KEY (qu_id) REFERENCES question (qu_id)
);

-- ----------------------------------------------------------
-- Table: attempt
-- ----------------------------------------------------------
CREATE TABLE attempt (
  at_id        INT      NOT NULL AUTO_INCREMENT,
  us_id        INT      NOT NULL,
  at_date      DATETIME NOT NULL,
  PRIMARY KEY (at_id),
  CONSTRAINT fk_att_user FOREIGN KEY (us_id) REFERENCES user (us_id)
);

-- ----------------------------------------------------------
-- Table: attempt_question
-- ----------------------------------------------------------
CREATE TABLE attempt_question (
  aq_id     INT          NOT NULL AUTO_INCREMENT,
  at_id     INT          NOT NULL,
  qu_id     INT          NOT NULL,
  aq_score  DECIMAL(4,1) NOT NULL DEFAULT 0.0,
  PRIMARY KEY (aq_id),
```

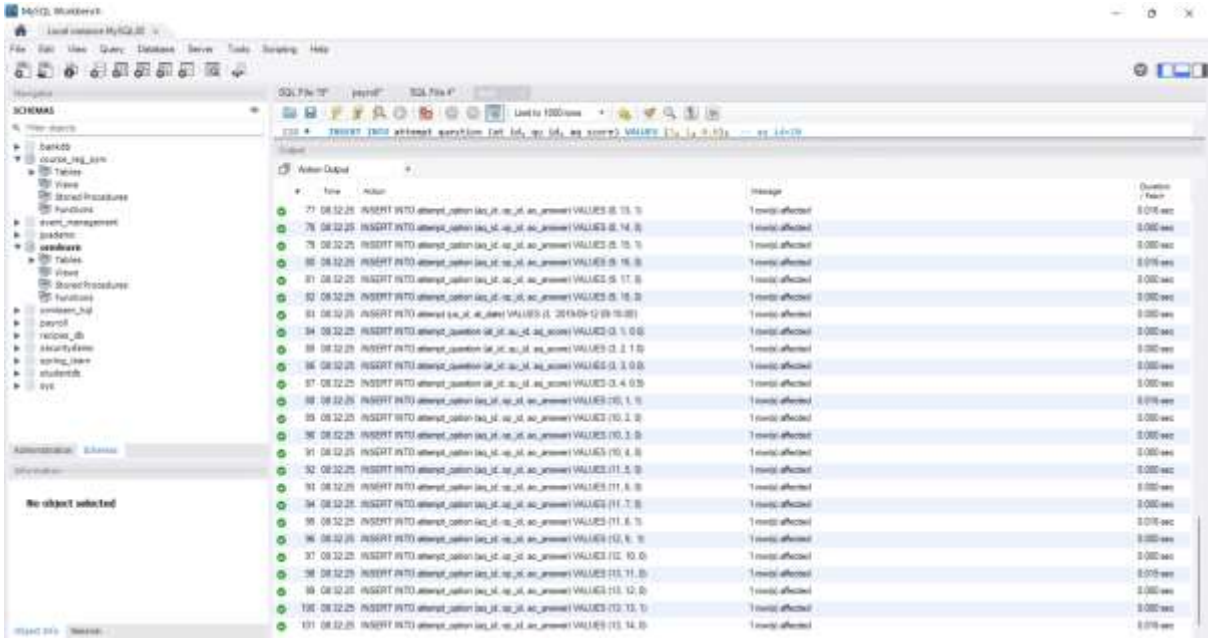
```
-- ----------------------------------------------------------
-- Table: attempt_option
-- ----------------------------------------------------------
CREATE TABLE attempt_option (
  ao_id      INT          NOT NULL AUTO_INCREMENT,
  aq_id      INT          NOT NULL,
  op_id      INT          NOT NULL,
  ao_answer  TINYINT(1)   NOT NULL DEFAULT 0,
  PRIMARY KEY (ao_id),
  CONSTRAINT fk_ao_aq     FOREIGN KEY (aq_id) REFERENCES attempt_question (aq_id),
  CONSTRAINT fk_ao_option FOREIGN KEY (op_id) REFERENCES options (op_id)
);

-- ==========================================================
-- Seed Data
-- ==========================================================

-- Users
INSERT INTO user (us_name, us_email) VALUES ('John Doe',   'john.doe@example.com');
INSERT INTO user (us_name, us_email) VALUES ('Jane Smith', 'jane.smith@example.com');
INSERT INTO user (us_name, us_email) VALUES ('Alex Kumar', 'alex.kumar@example.com');

-- Questions
-- Q1: HTML file extension (score 1.0)
INSERT INTO question (qu_text, qu_score)
  VALUES ('What is the extension of the hyper text markup language file?', 1.0);

-- Q2: Max heading level (score 1.0)
```





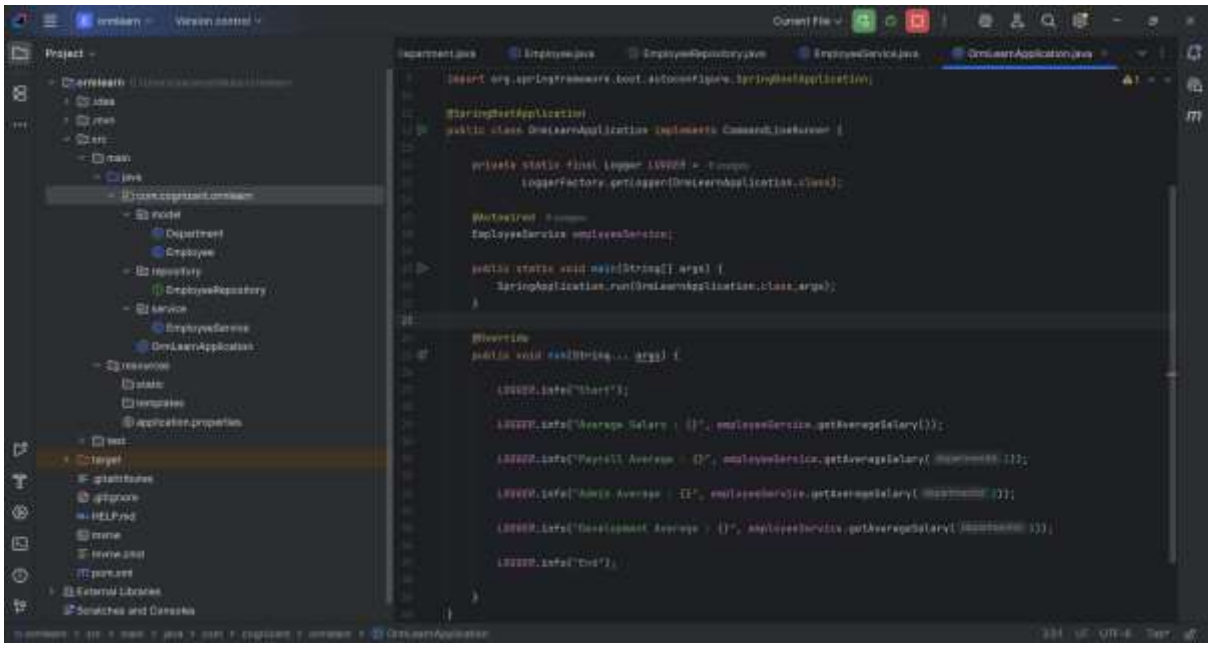


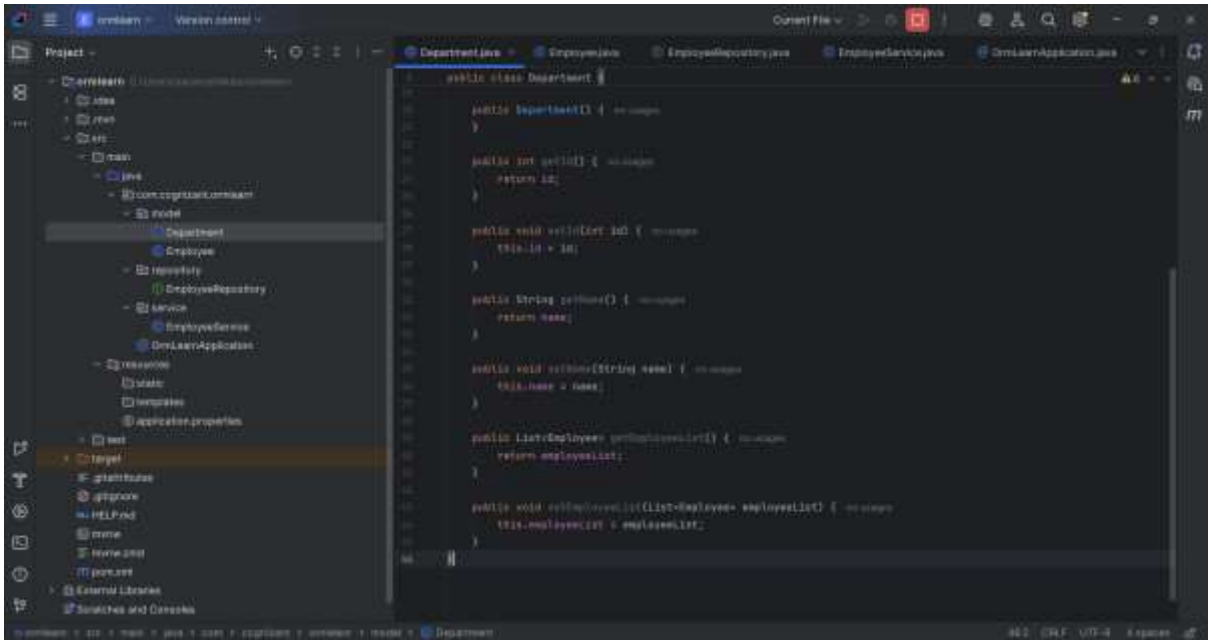


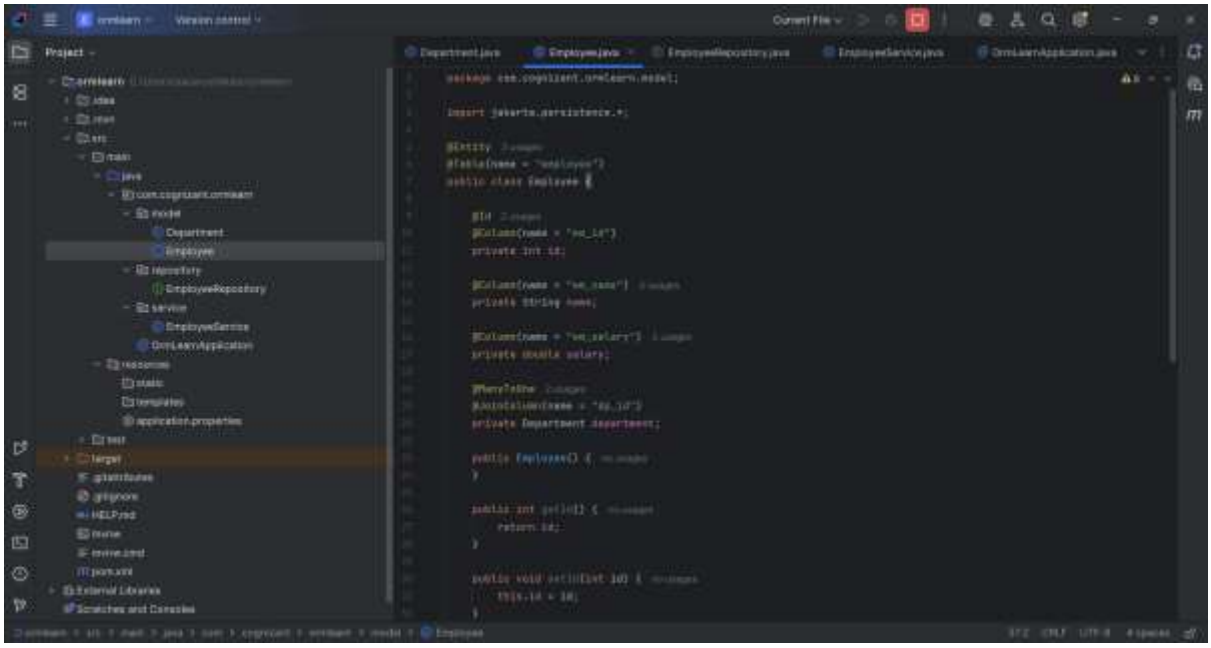


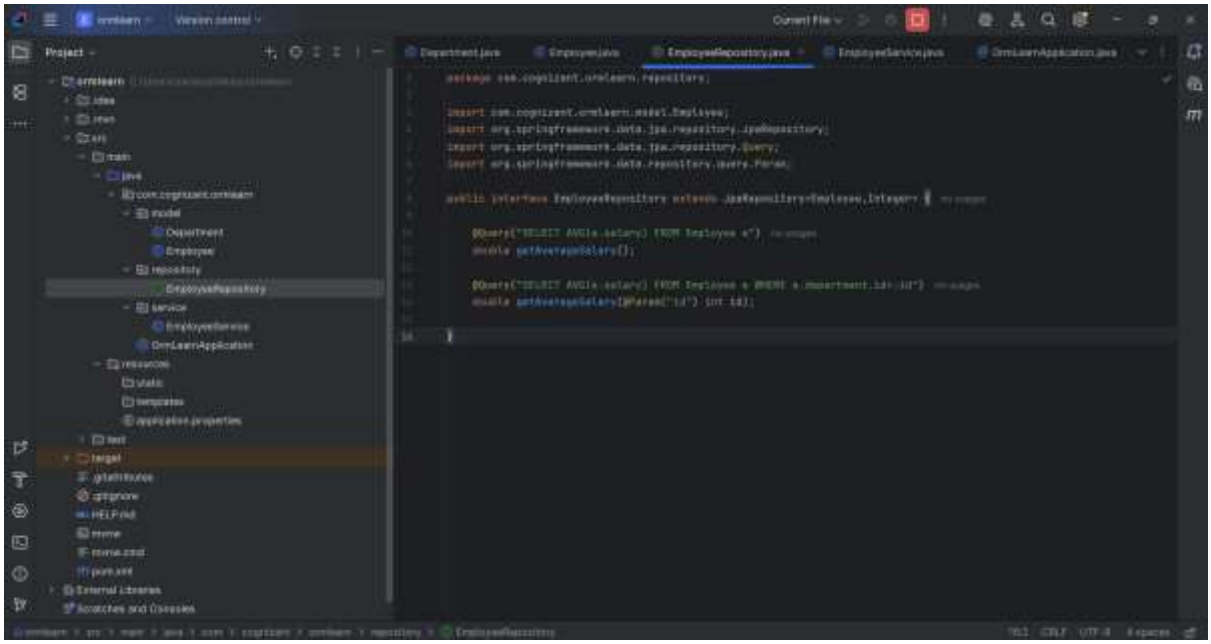
Hands on 4

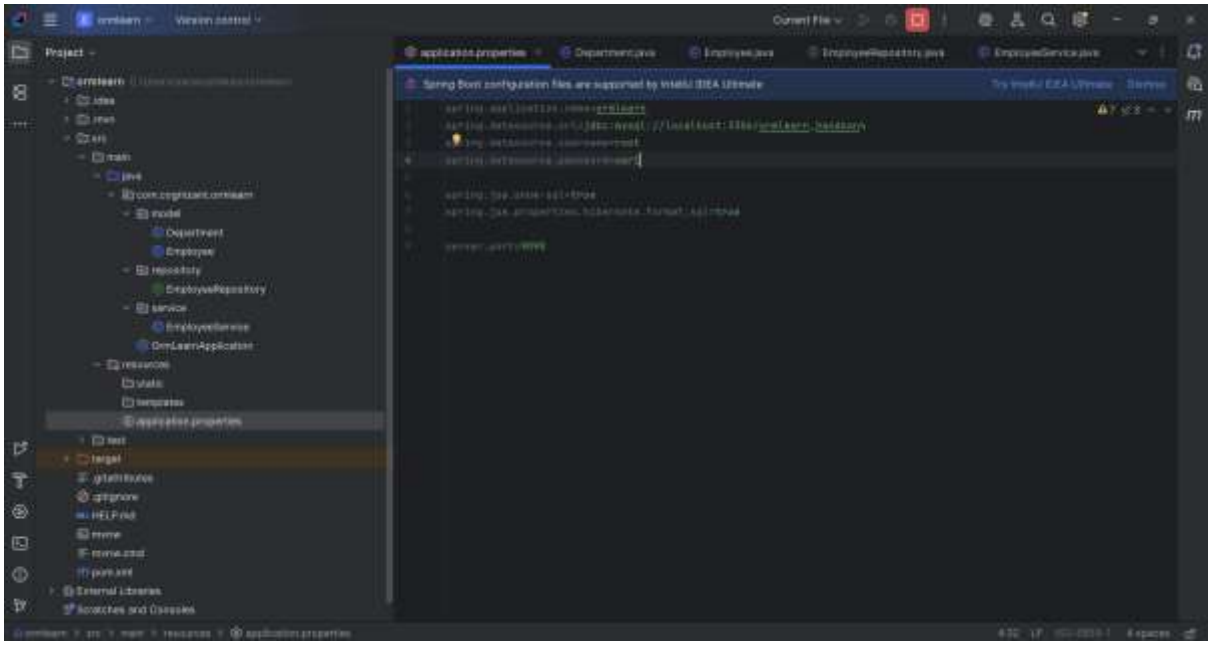
Get average salary using HQL



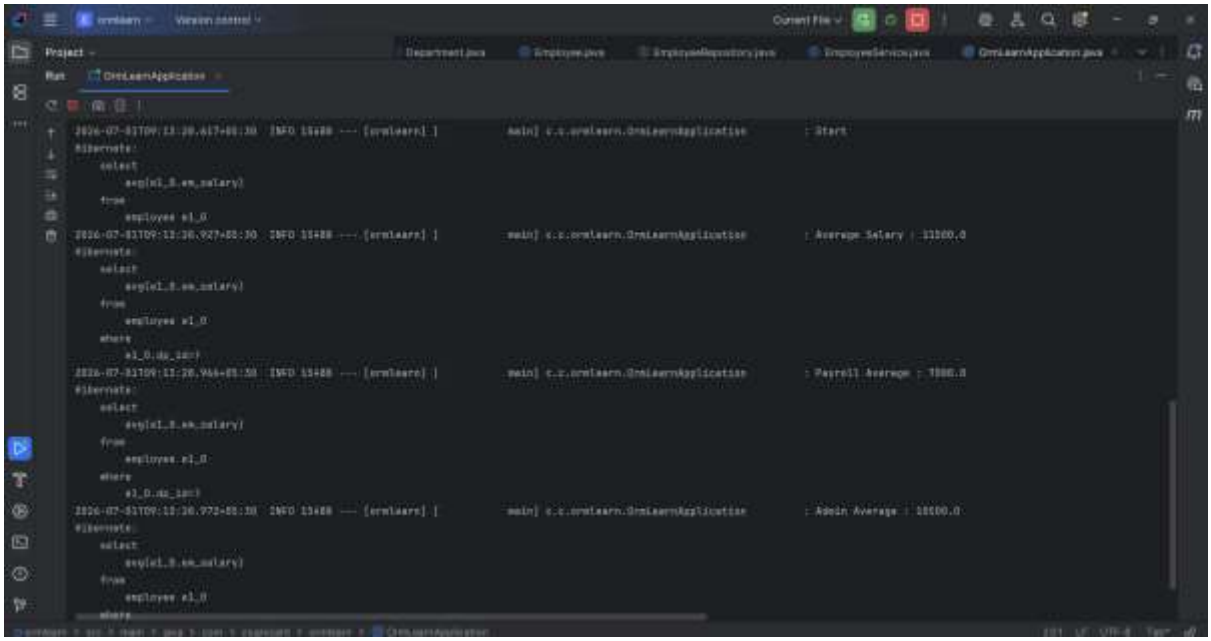


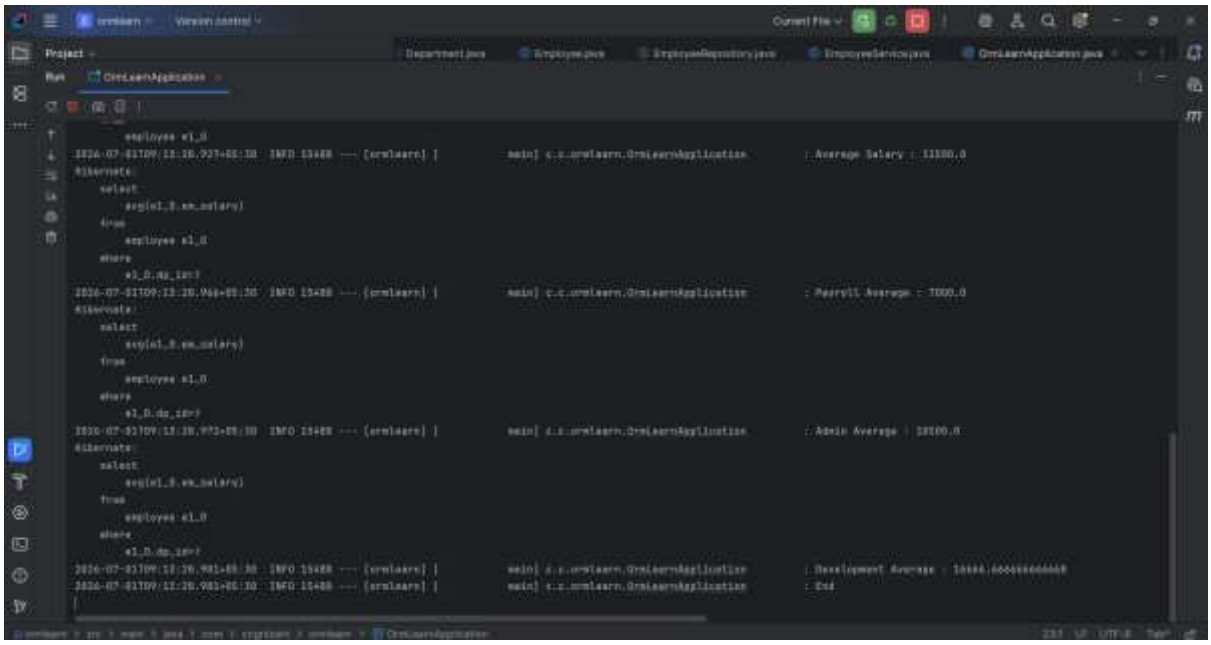






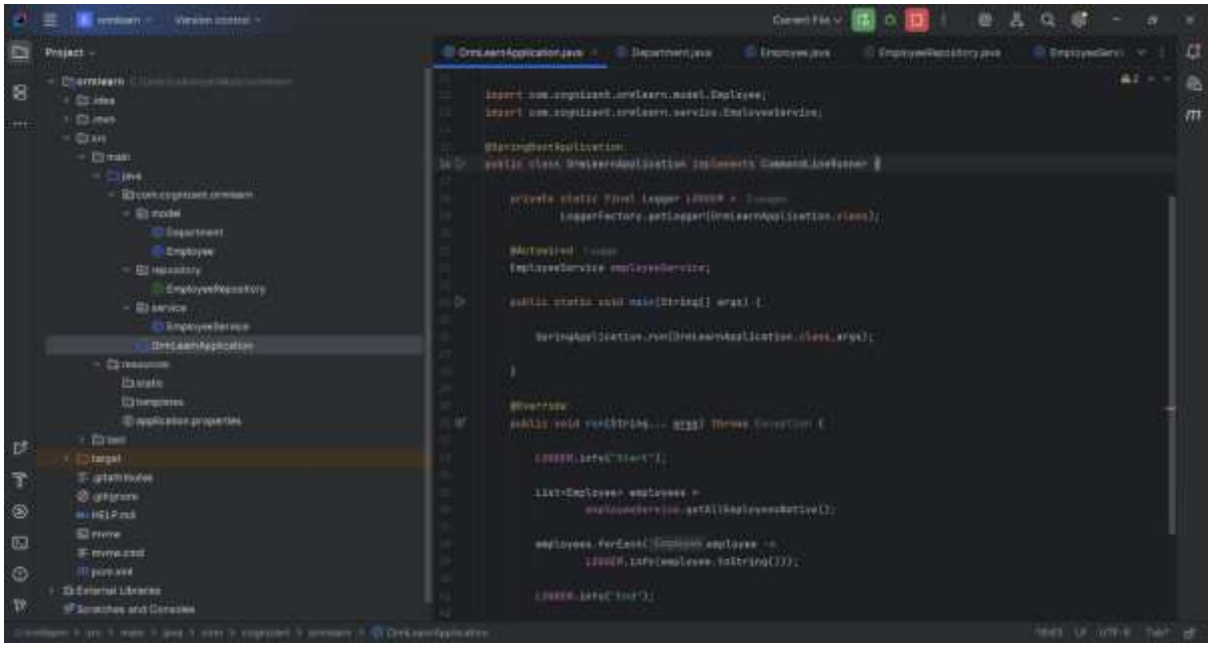
OUTPUT:

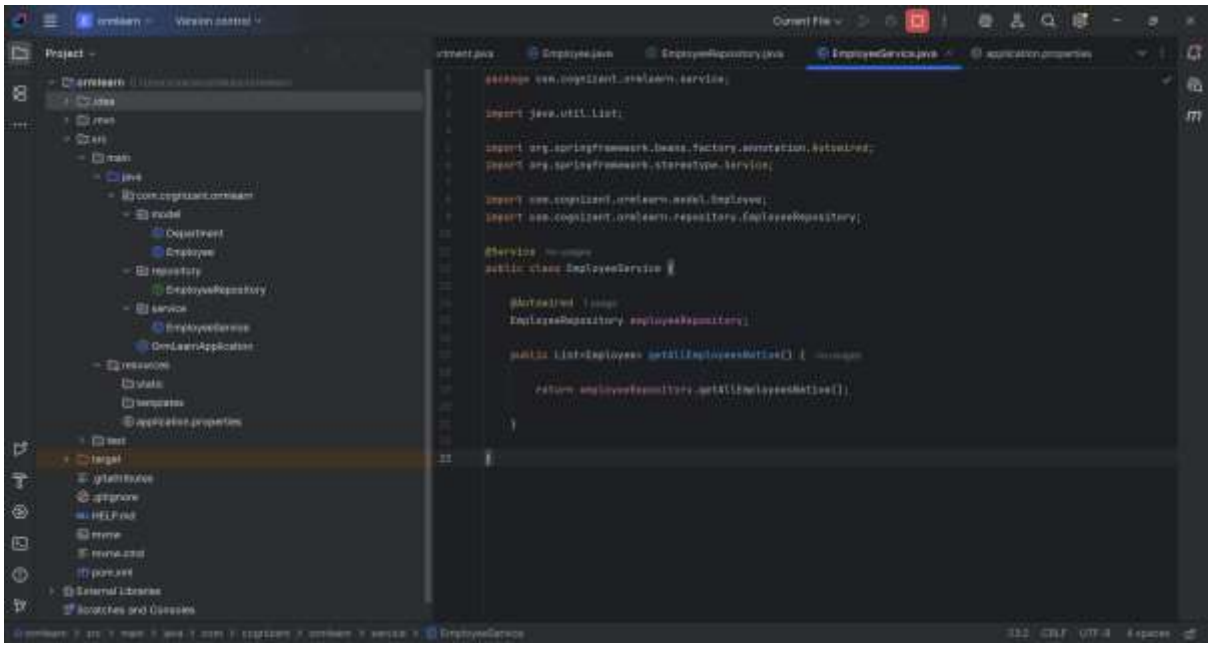


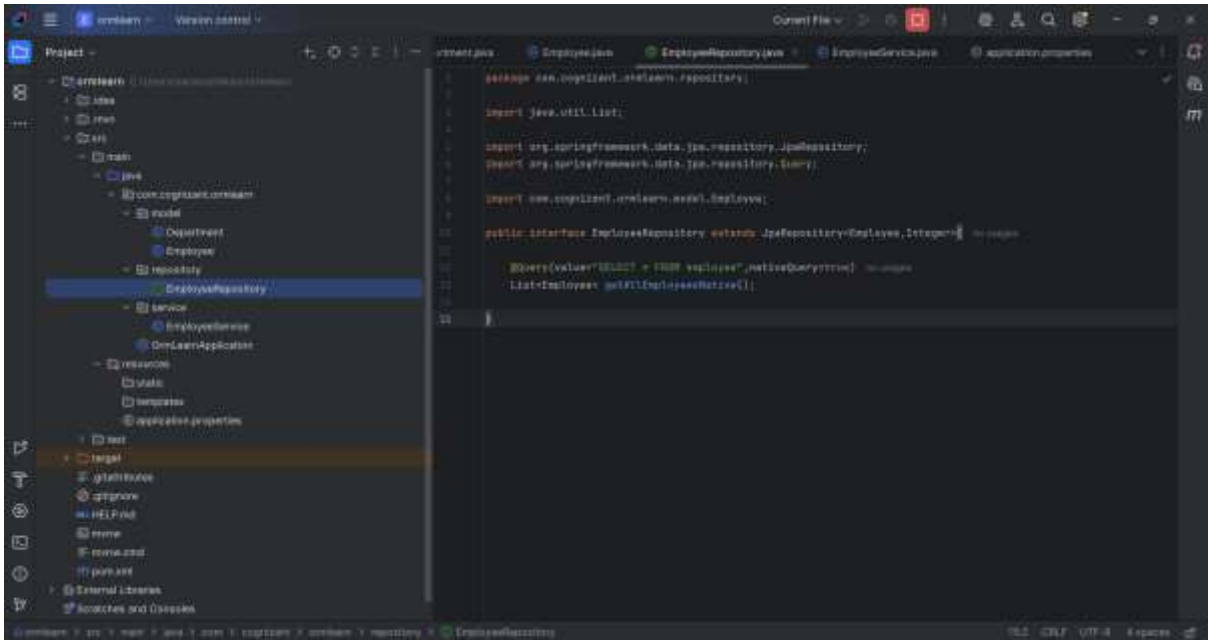


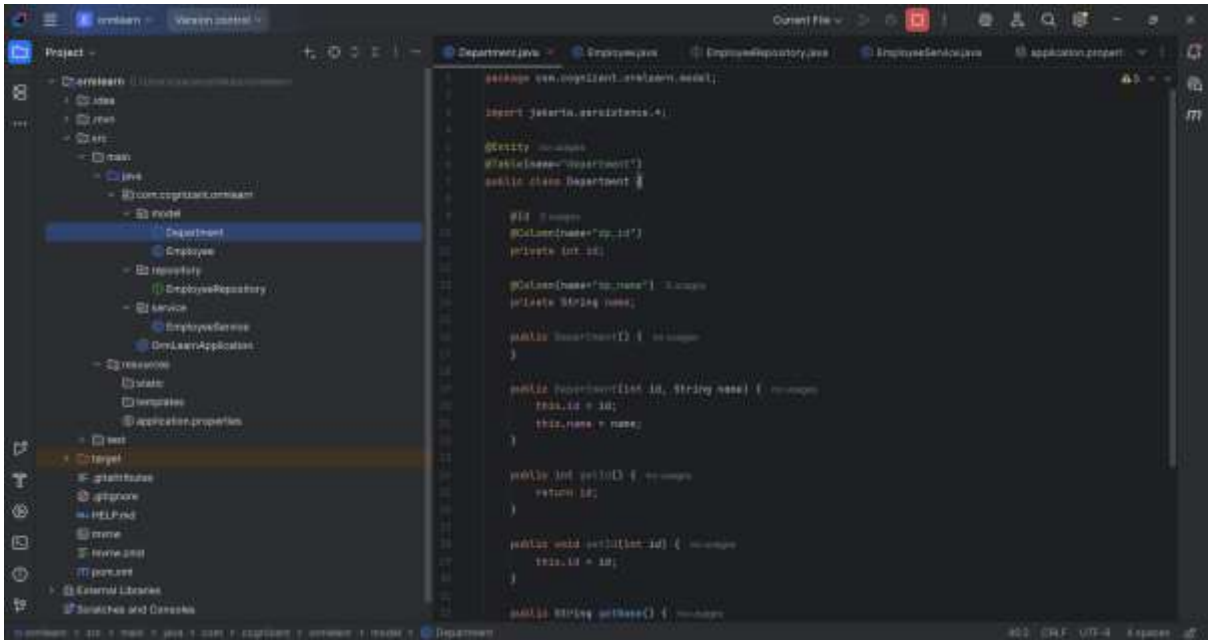
Hands on 5

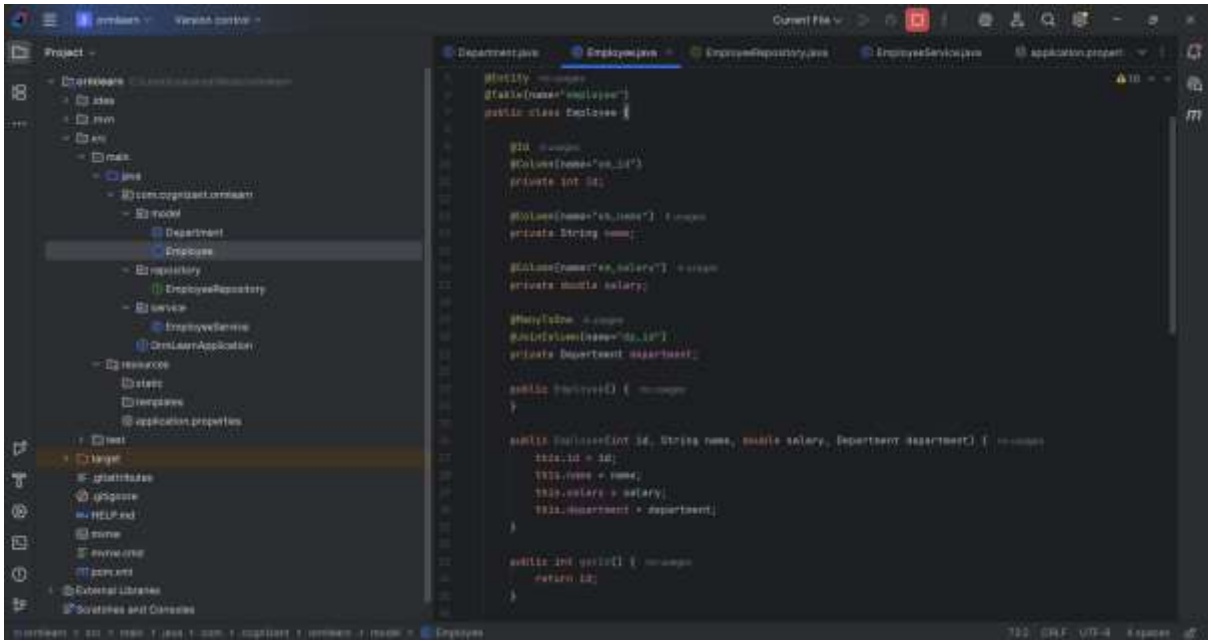
Get all employees using Native Query

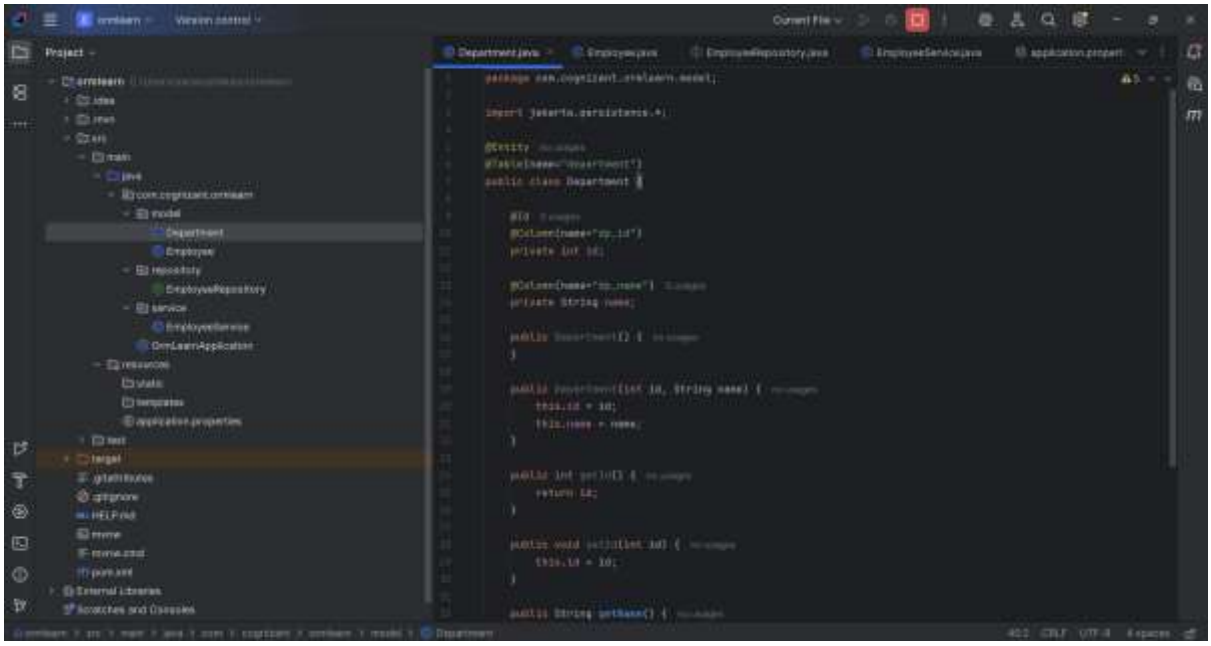


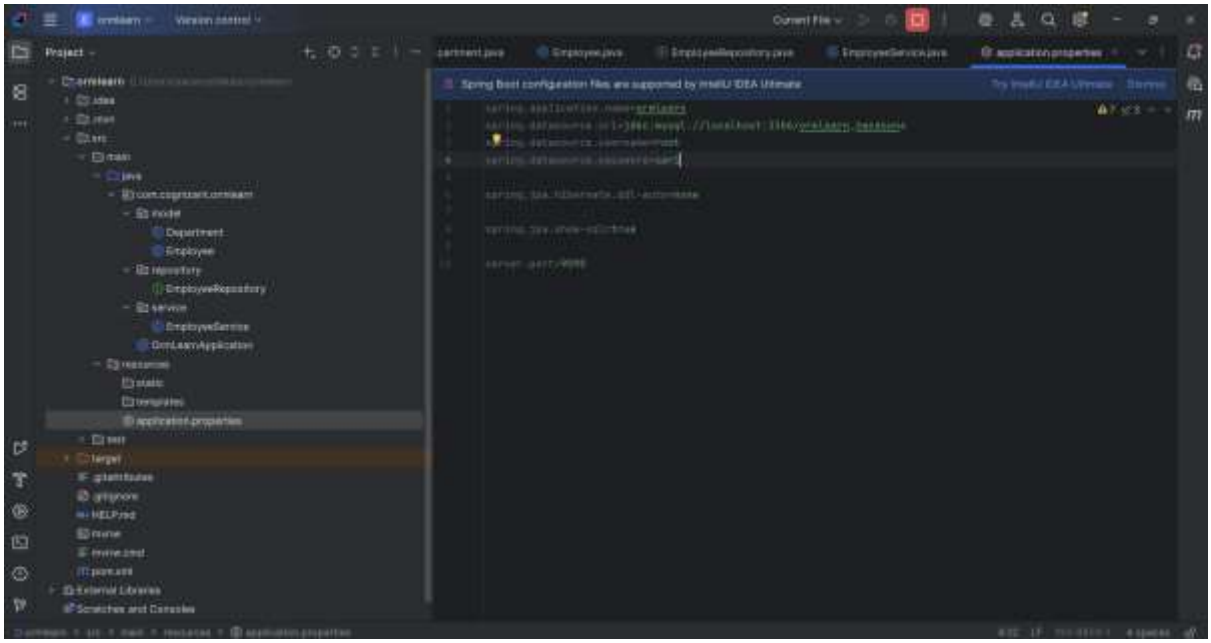




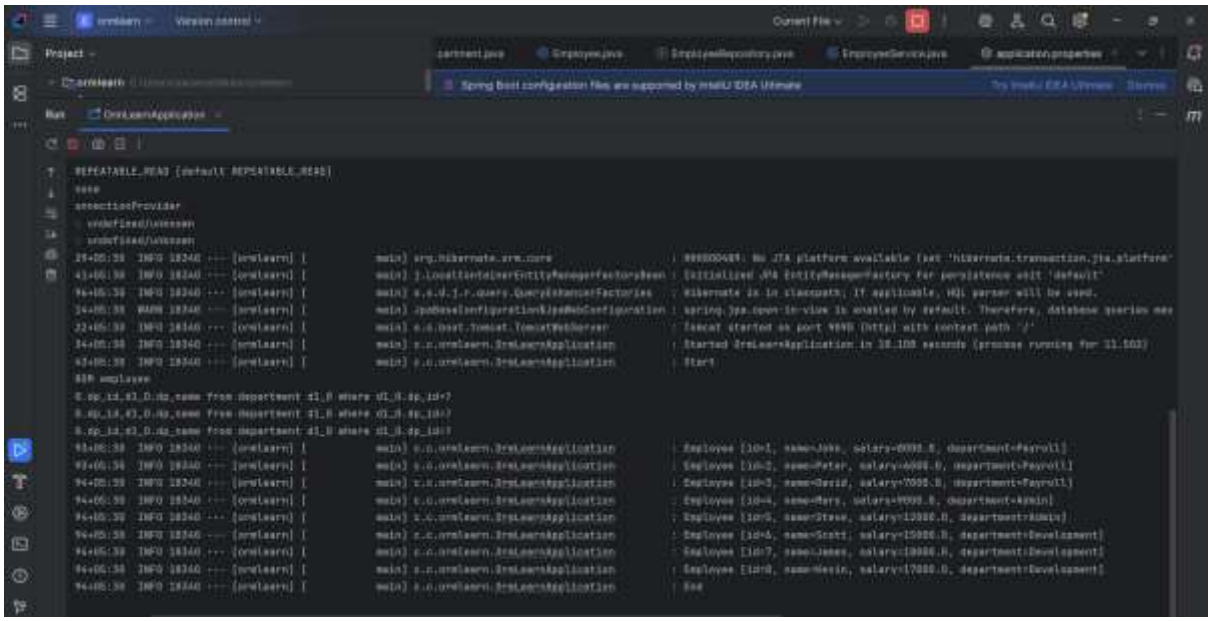






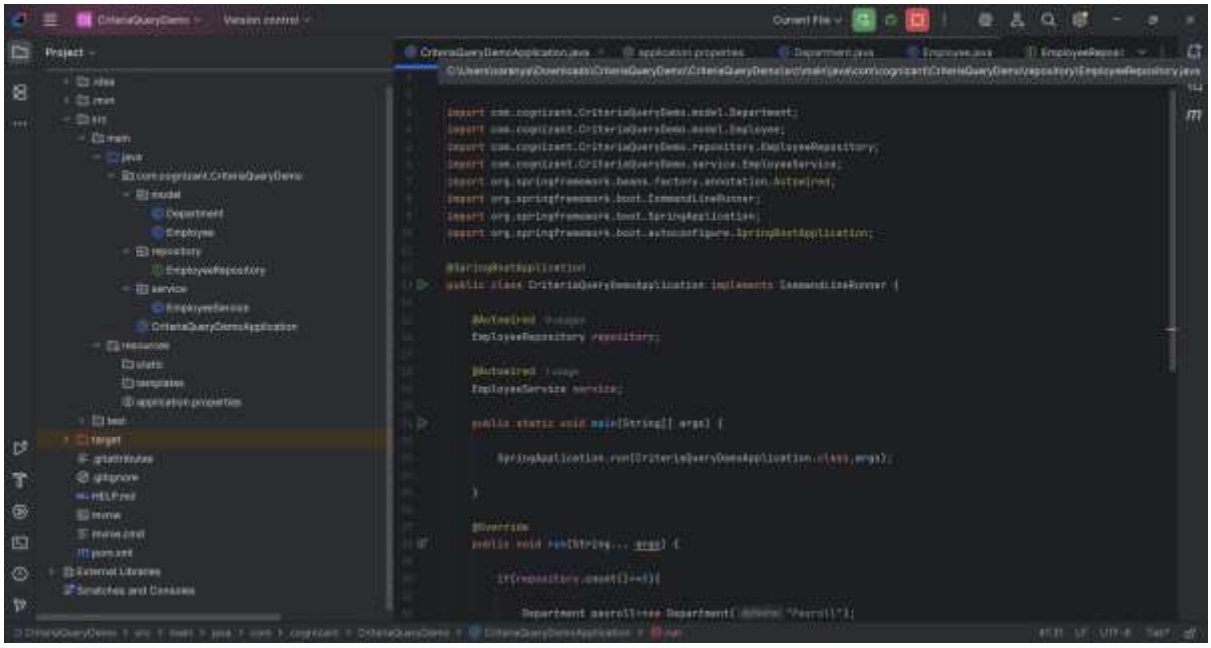


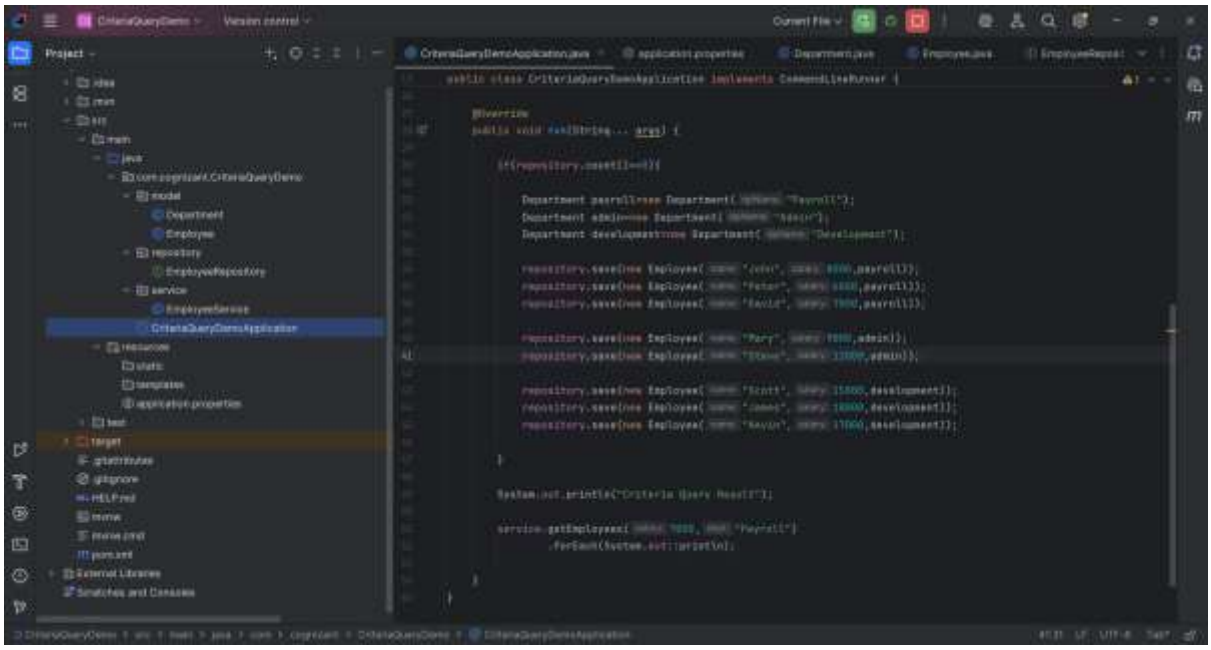
OUTPUT:



Hands on 6

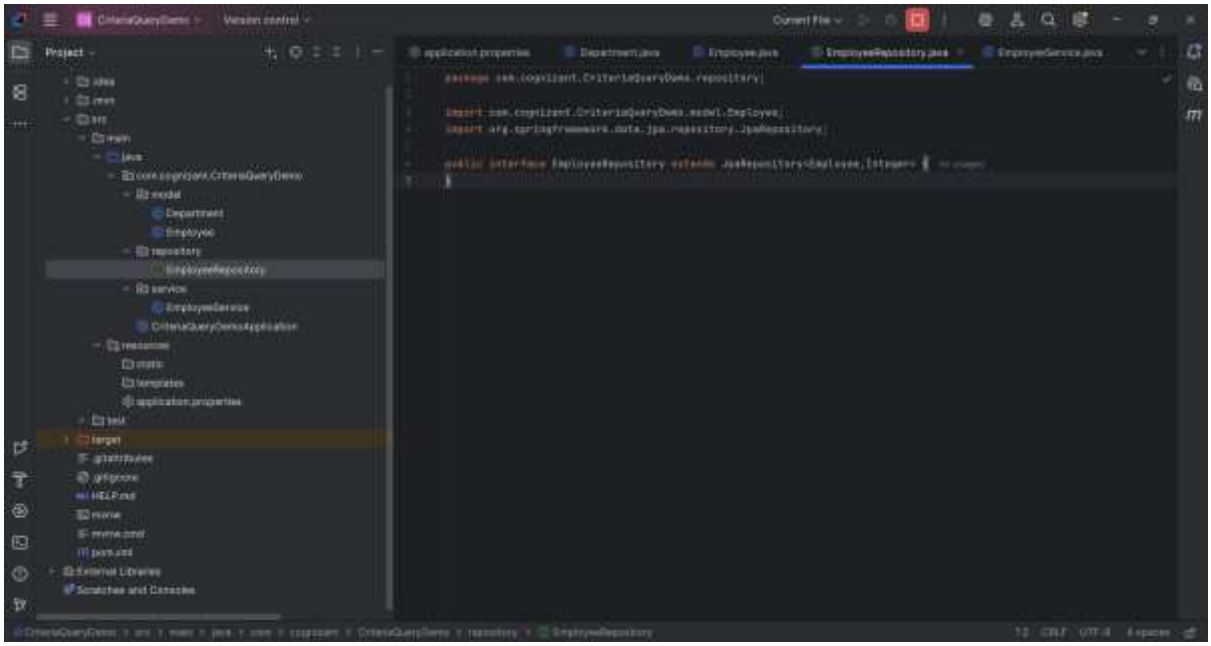
Criteria Query

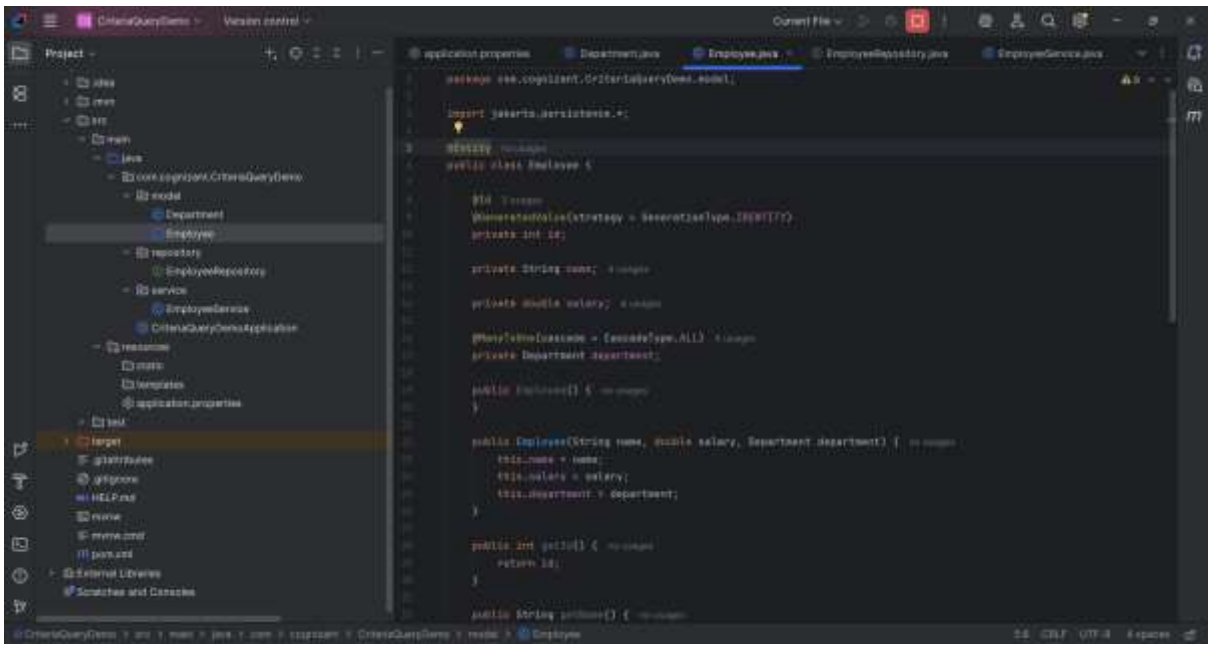


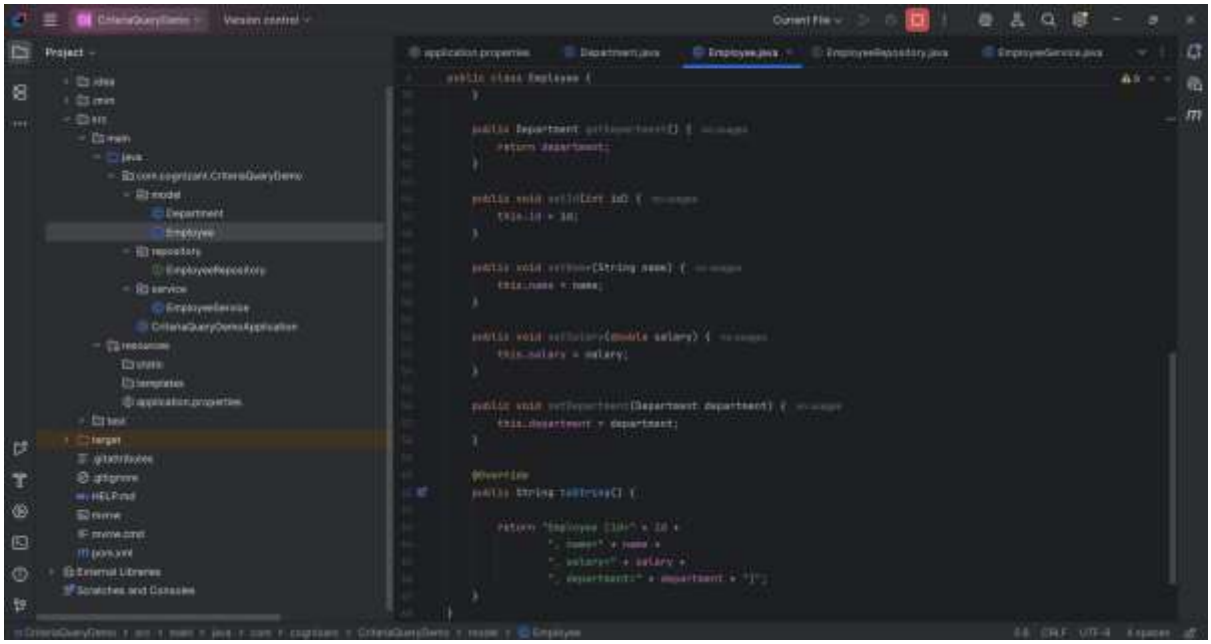


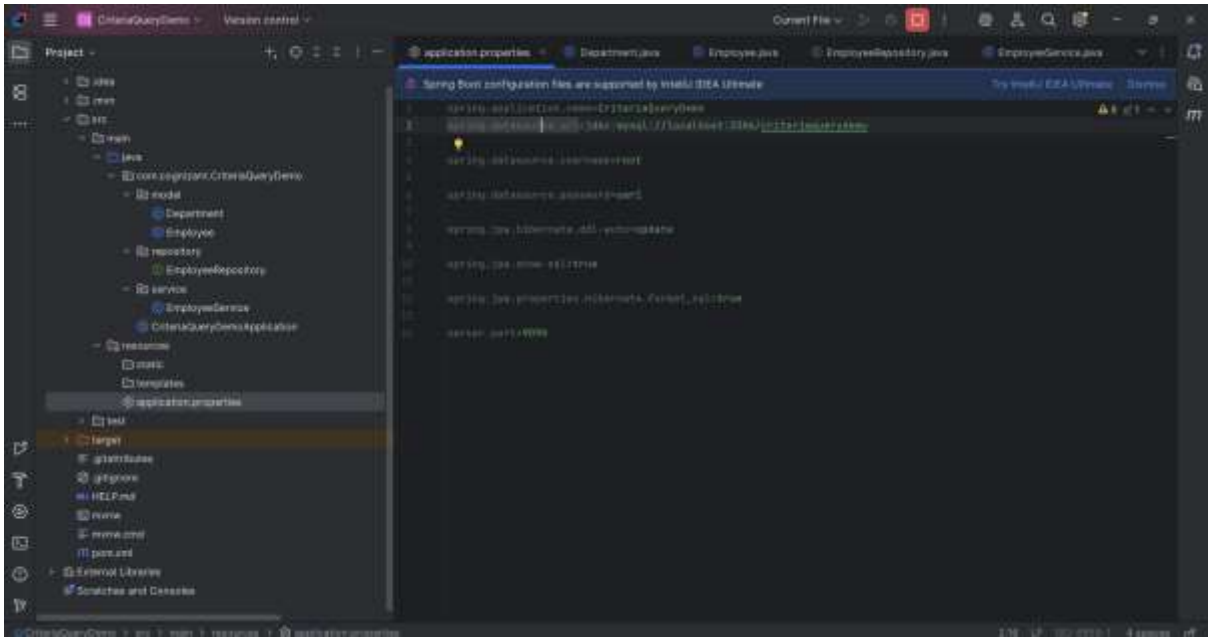
```
import com.cognizant.CriteriaQueryDemo.model.Employee;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.criteria.CriteriaBuilder;
import jakarta.persistence.criteria.CriteriaQuery;
import jakarta.persistence.criteria.Predicate;
import jakarta.persistence.criteria.Root;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    @PersistenceContext
    private EntityManager entityManager;

    public List<Employee> getEmployees(double salary, String dept){

        CriteriaBuilder cb = entityManager.getCriteriaBuilder();

        CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);

        Root<Employee> employee = cq.from(Employee.class);

        Predicate salaryPredicate =
                cb.greaterThan(employee.get("salary"),salary);

        Predicate deptPredicate =
                cb.equal(employee.get("department").get("dname"),dept);
```

```
    public class EmployeeService {

        @PersistenceContext
        private EntityManager entityManager;

        public List<Employee> getEmployees(double salary, String dept){

            CriteriaBuilder cb = entityManager.getCriteriaBuilder();

            CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);

            Root<Employee> employee = cq.from(Employee.class);

            Predicate salaryPredicate =
                    cb.greaterThan(employee.get("salary"),salary);

            Predicate deptPredicate =
                    cb.equal(employee.get("department").get("dname"),dept);

            cq.select(employee)
                    .where(cb.and(salaryPredicate,deptPredicate));

            return entityManager.createQuery(cq).getResultList();

        }

    }
```


CriteriaQueryDemo
Version control
Current File
Project
application.properties
Department.java
Employee.java
EmployeeRepository.java
EmployeeService.java
package com.cognizant.CriteriaQueryDemo.repository;
import com.cognizant.CriteriaQueryDemo.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
public interface EmployeeRepository extends JpaRepository<Employee,Integer> {
}
com.cognizant.CriteriaQueryDemo
model
Department
Employee
repository
EmployeeRepository
service
EmployeeService
CriteriaQueryDemoApplication
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles


CriteriaQueryDemo
Version control
Current File
Project
application.properties
Department.java
Employee.java
EmployeeRepository.java
EmployeeService.java
package com.cognizant.CriteriaQueryDemo.model;
import jakarta.persistence.*;
@Entity
public class Employee {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
private String name;
private double salary;
@ManyToOne(cascade = {CascadeType.ALL})
private Department department;
public Employee() {
}
public Employee(String name, double salary, Department department) {
this.name = name;
this.salary = salary;
this.department = department;
}
public int getId() {
return id;
}
public String getName() {
com.cognizant.CriteriaQueryDemo.model.Employee
model
Department
Employee
repository
EmployeeRepository
service
EmployeeService
CriteriaQueryDemoApplication
resources
static
templates
application.properties
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles





OUTPUT:

```
2026-07-01T10:01:27.720+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
    Database JDBC URL [jdbc:mysql://localhost:3306/criteriaquerydemo]
    Database driver: MySQL Connector/J
    Database dialect: MySQL8Dialect
    Database version: 8.0.44
    Default catalog/schema: criteriaquerydemo/undefined
    Autocommit mode: undefined/unknown
    Isolation level: REPEATABLE_READ [default REPEATABLE_READ]
    JDBC fetch size: none
    Pool: DataSourceConnectionProvider
    Minimum pool size: undefined/unknown
    Maximum pool size: undefined/unknown
2026-07-01T10:01:29.558+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] org.hibernate.orm.core                   : HHH000489: No JTA platform available (set 'hibern
2026-07-01T10:01:29.720+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persist
2026-07-01T10:01:29.914+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] o.s.d.j.r.query.QueryEnhancerFactories   : Hibernate is in classpath; If applicable, HQL pa
2026-07-01T10:01:30.181+05:30  WARN 13444 --- [CriteriaQueryDemo] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. T
2026-07-01T10:01:30.995+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] o.s.boot.tomcat.TomcatWebServer          : Tomcat started on port 9090 (http) with context
2026-07-01T10:01:31.010+05:30  INFO 13444 --- [CriteriaQueryDemo] [           main] c.c.C.CriteriaQueryDemoApplication       : Started CriteriaQueryDemoApplication in 10.579 s
Hibernate:
    select
        count(*)
    from
        employee e1_0
Criteria Query Result
Hibernate:
    select
        e1_0.id,
        e1_0.department_dp_id,
```

```
        count(*)
    from
        employee e1_0
Criteria Query Result
Hibernate:
    select
        e1_0.id,
        e1_0.department_dp_id,
        e1_0.name,
        e1_0.salary
    from
        employee e1_0
    join
        department d1_0
            on e1_0.dp_id=d1_0.department_dp_id
    where
        e1_0.salary>?
        and d1_0.dp_name=?
Hibernate:
    select
        d1_0.dp_id,
        d1_0.dp_name
    from
        department d1_0
    where
        d1_0.dp_id=?
Employee [id=1, name=John, salary=8000.0, department=Payroll]
```