**Q1)     Develop a basic Create, Read operation using Hibernate for a simple entity, such as Student.**

**Answer:**

**a). 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.demo</groupId>

<artifactId>School\_Management\_System</artifactId> <!-- Corrected typo in artifactId -->

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>School\_Management\_System</name> <!-- Corrected typo in name -->

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Hibernate dependencies -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.6.15.Final</version> <!-- Updated to the latest version -->

</dependency>

<!-- MySQL Connector dependency -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.30</version> <!-- Updated to the latest version -->

</dependency>

<!-- JPA API -->

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>javax.persistence-api</artifactId>

<version>2.2</version>

</dependency>

<!-- JUnit dependency for testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

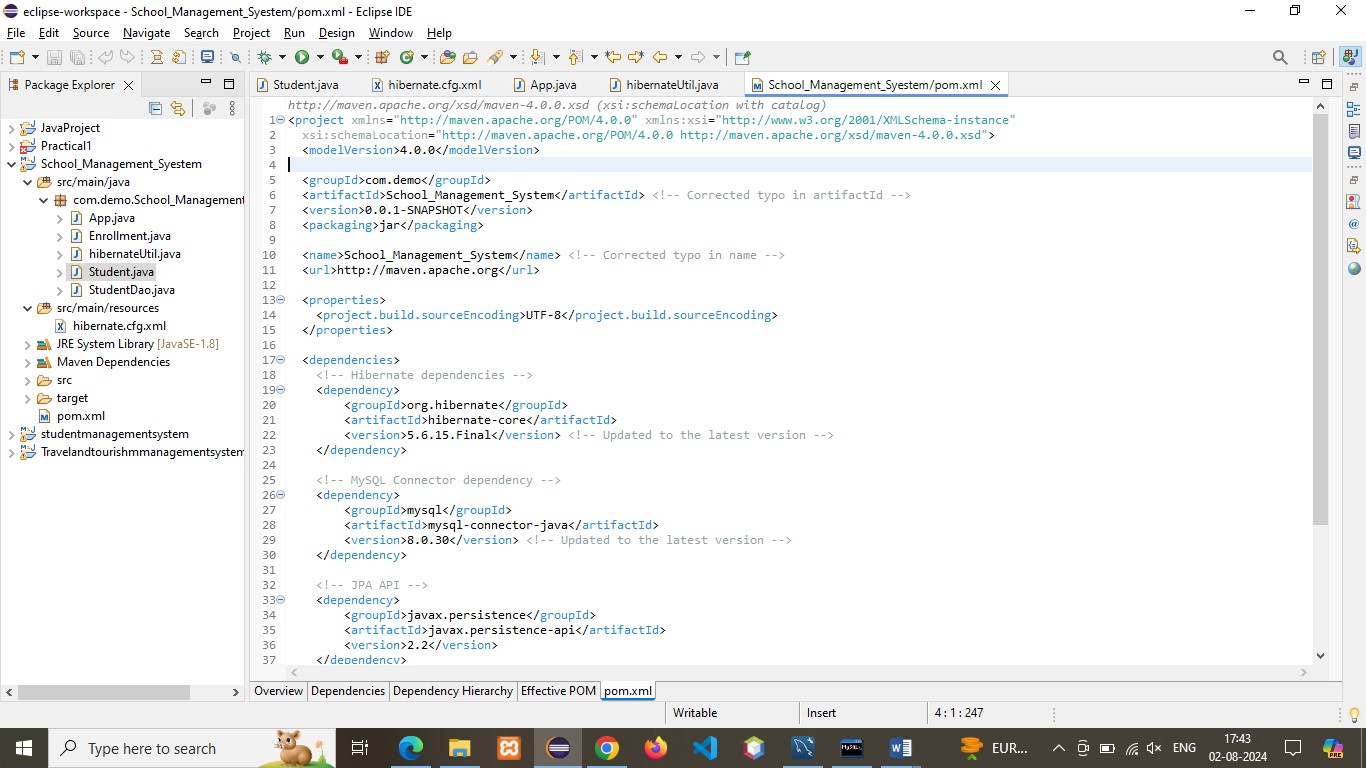
<version>4.13.2</version> <!-- Updated to a more recent version -->

<scope>test</scope>

</dependency>

</dependencies>

</project>



**b) . Hibernate.cfg.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/school\_management\_syestem</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>rinki</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQL8Dialect</property>

<property name=*"hibernate.hbm2ddl.auto"*>update</property>

<!-- Optional properties -->

<!-- JDBC connection pool settings -->

<property name=*"hibernate.c3p0.min\_size"*>5</property>

<property name=*"hibernate.c3p0.max\_size"*>20</property>

<property name=*"hibernate.c3p0.timeout"*>300</property>

<property name=*"hibernate.c3p0.max\_statements"*>50</property>

<property name=*"hibernate.c3p0.idle\_test\_period"*>3000</property>

<!-- SQL dialect -->

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<!-- Echo all executed SQL to stdout -->

<property name=*"hibernate.show\_sql"*>true</property>

<!-- Drop and re-create the database schema on startup -->

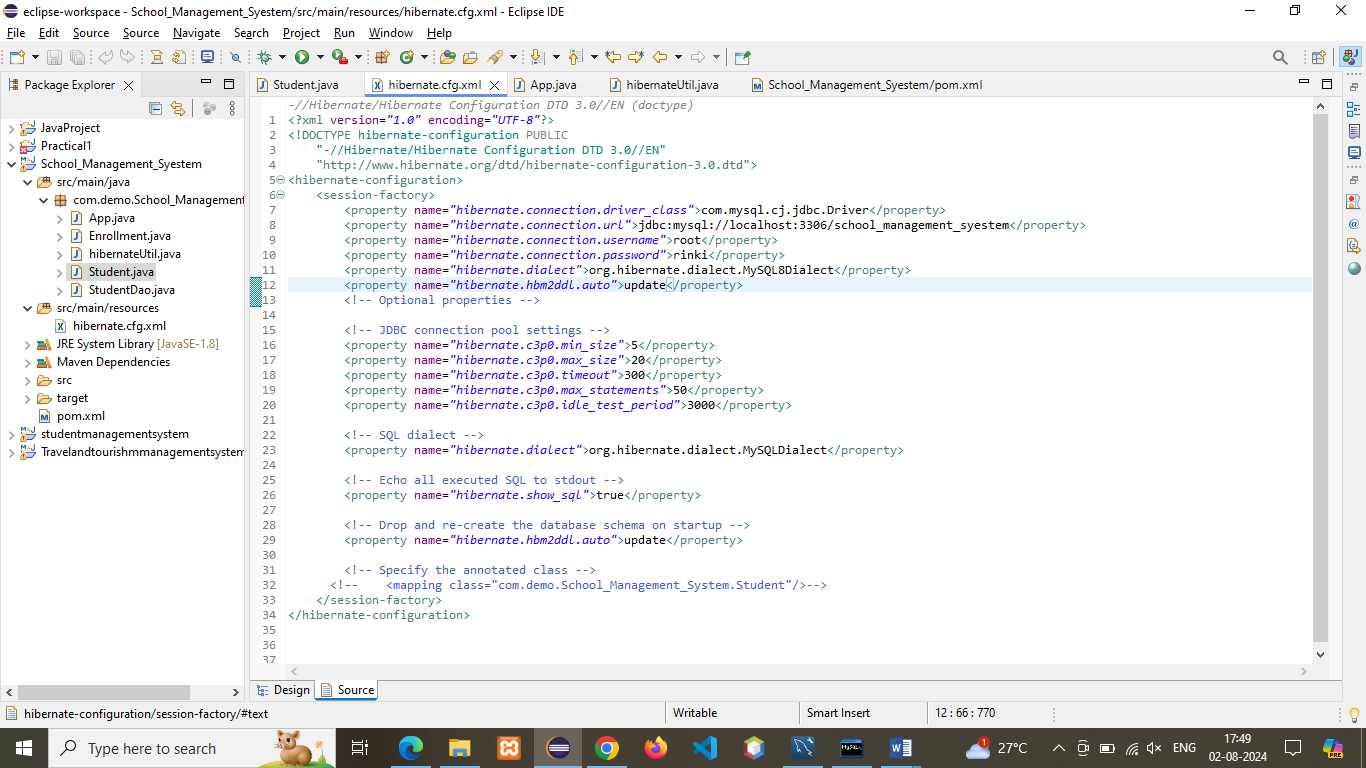
<property name=*"hibernate.hbm2ddl.auto"*>update</property>

<!-- Specify the annotated class -->

<!-- <mapping class="com.demo.School\_Management\_System.Student"/>-->

</session-factory>

</hibernate-configuration>



**C) .HibernateUtil.java**

**package** com.demo.college\_management\_system;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**public** **class** hibernateUtil {

**private** **static** **final** SessionFactory ***sessionFactory*** = *buildSessionFactory*();

**private** **static** SessionFactory buildSessionFactory() {

**try** {

**return** **new** Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Student.**class**)

.buildSessionFactory();

} **catch** (Throwable ex) {

**throw** **new** ExceptionInInitializerError(ex);

}

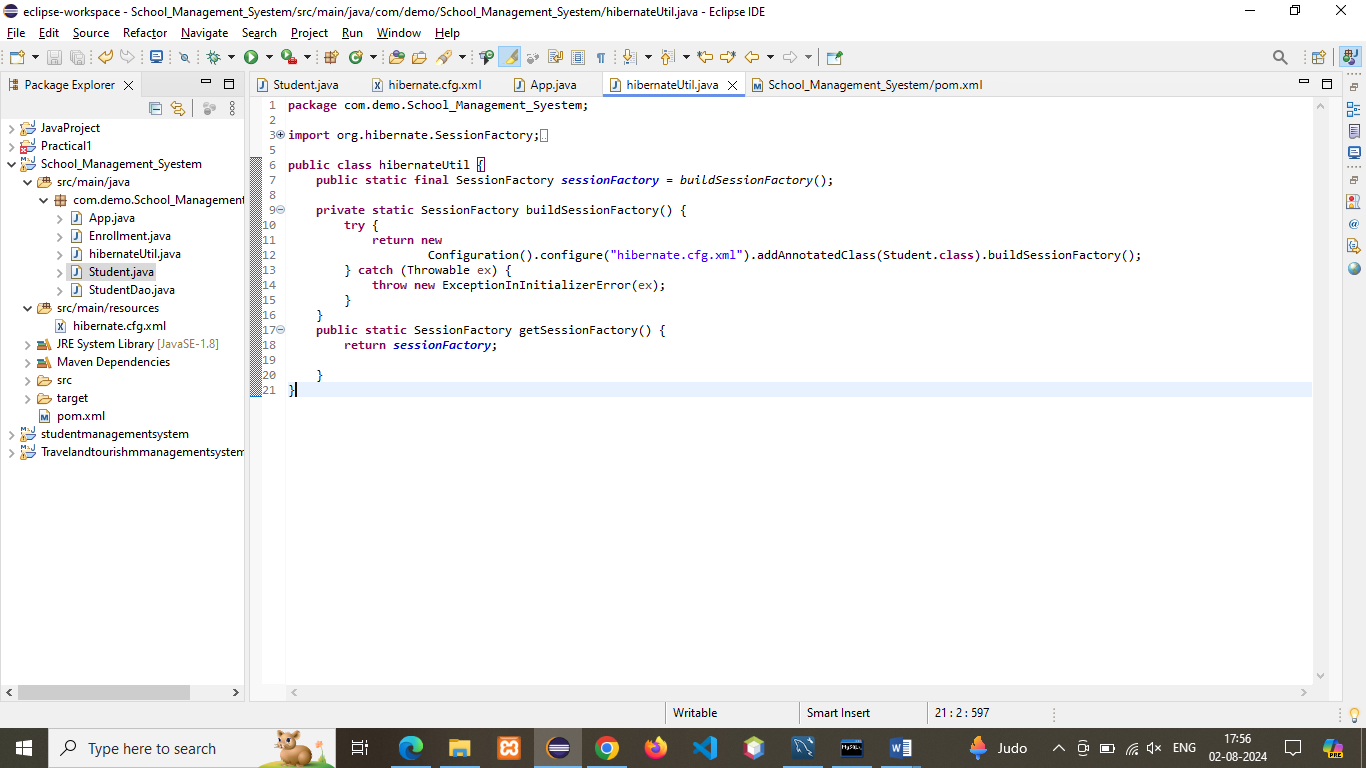
}

**public** **static** SessionFactory getSessionFactory() {

**return** ***sessionFactory***;

}

}



**D) .Student.java**

**package** com.demo.School\_Management\_Syestem;

**import** java.time.LocalDate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

/\*\*

\* Entity class representing a student in the School Management System.

\*/

@Entity

@Table(name = "Student")

**public** **class** Student {

@Id

@Column(name = "student\_id")

**private** String studentId;

@Column(name = "first\_name")

**private** String firstName;

@Column(name = "last\_name")

**private** String lastName;

@Column(name = "dob")

**private** LocalDate dob;

@Column(name = "email")

**private** String email;

@Column(name = "phone")

**private** String phone;

@Column(name = "address")

**private** String address;

// Default constructor

**public** Student() {}

// Parameterized constructor

**public** Student(String studentId, String firstName, String lastName, LocalDate dob, String email, String phone, String address) {

**this**.studentId = studentId;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.dob = dob;

**this**.email = email;

**this**.phone = phone;

**this**.address = address;

}

// Getters and setters

**public** String getStudentId() {

**return** studentId;

}

**public** **void** setStudentId(String studentId) {

**this**.studentId = studentId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** LocalDate getDob() {

**return** dob;

}

**public** **void** setDob(LocalDate dob) {

**this**.dob = dob;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

@Override

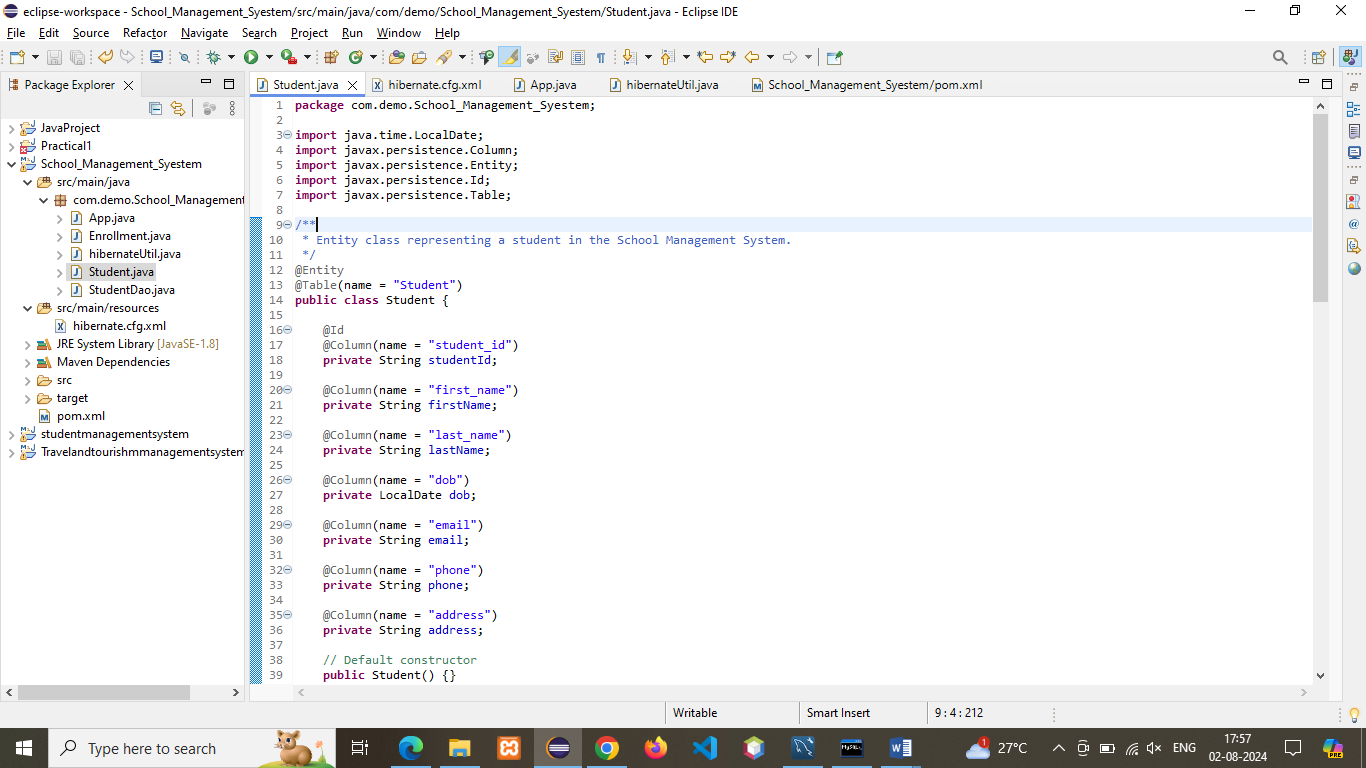
**public** String toString() {

**return** "Student [studentId=" + studentId + ", firstName=" + firstName + ", lastName=" + lastName + ", dob=" + dob

+ ", email=" + email + ", phone=" + phone + ", address=" + address + "]";

}

}



**E).StudentDAO.java**

**package** com.demo.college\_management\_system;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.query.Query;

**import** java.util.List;

**public** **class** StudentDAO {

**private** SessionFactory sessionFactory;

**public** StudentDAO(SessionFactory sessionFactory) {

**this**.sessionFactory = sessionFactory;

}

**public** Student createStudent(Student student) {

Transaction transaction = **null**;

**try** (Session session = sessionFactory.openSession()) {

transaction = session.beginTransaction();

session.save(student);

session.flush(); // Ensure all pending statements are executed

transaction.commit();

} **catch** (Exception e) {

**if** (transaction != **null**) {

transaction.rollback();

}

e.printStackTrace(); // Consider logging the exception

}

**return** student;

}

**public** Student getStudent(Long studentId) {

Student student = **null**;

**try** (Session session = sessionFactory.openSession()) {

student = session.get(Student.**class**, studentId);

} **catch** (Exception e) {

e.printStackTrace(); // Consider logging the exception

}

**return** student;

}

**public** List<Student> getAllStudents() {

List<Student> students = **null**;

**try** (Session session = sessionFactory.openSession()) {

Query<Student> query = session.createQuery("from Student", Student.**class**);

students = query.list();

} **catch** (Exception e) {

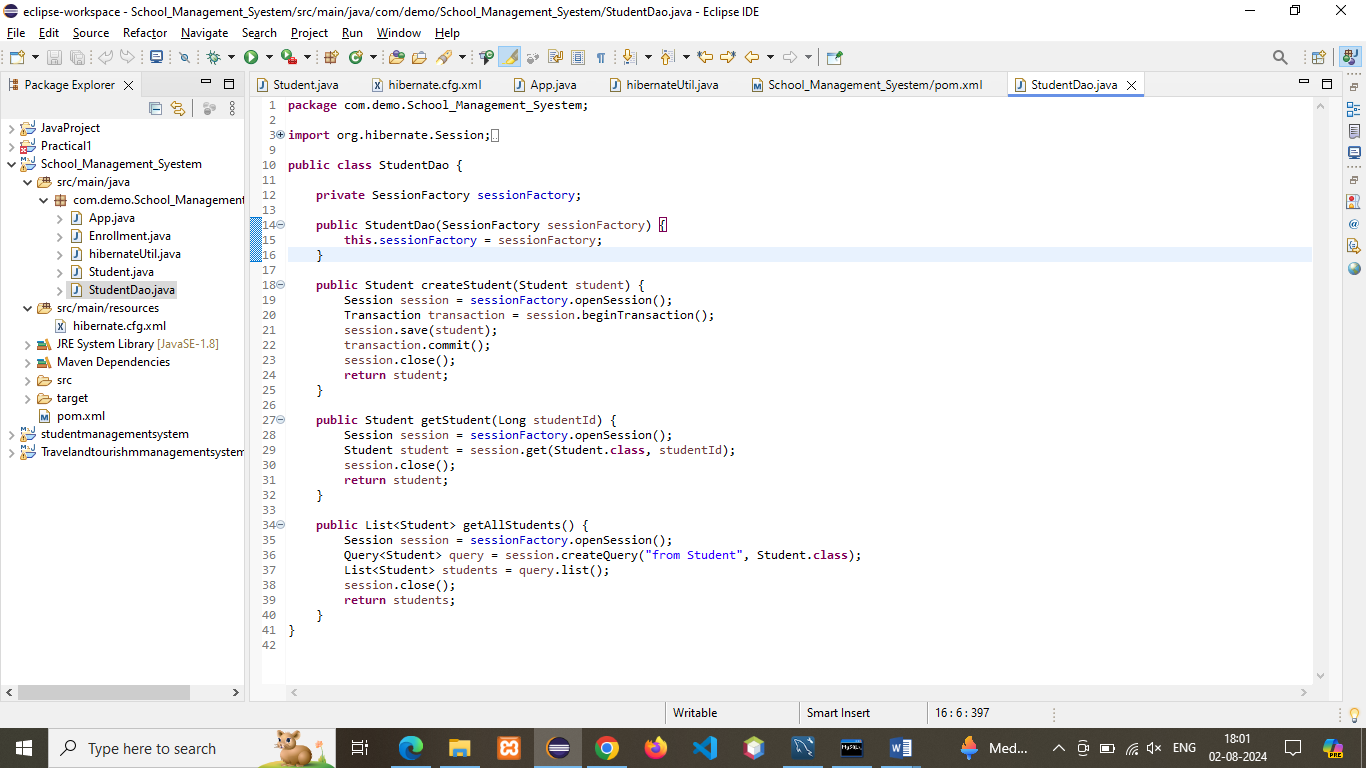
e.printStackTrace(); // Consider logging the exception

}

**return** students;

}

}



**F).App.java**

**package** com.demo.college\_management\_system;

**import** java.time.LocalDate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**public** **class** App {

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

// Obtain a Hibernate SessionFactory

SessionFactory factory = hibernateUtil.*getSessionFactory*();

// Create a new Student

LocalDate date1 = LocalDate.*of*(1988, 1, 13);

Student student1 = **new** Student("S111", "Oliver", "Henry", date1, "M", "oliver@gmail.com", "6742906745");

// Open a new session

Session session = factory.openSession();

// Begin a transaction

Transaction transaction = session.beginTransaction();

// Save the student to the database

session.save(student1);

// Commit the transaction

transaction.commit();

// Close the session

session.close();

// Close the session factory

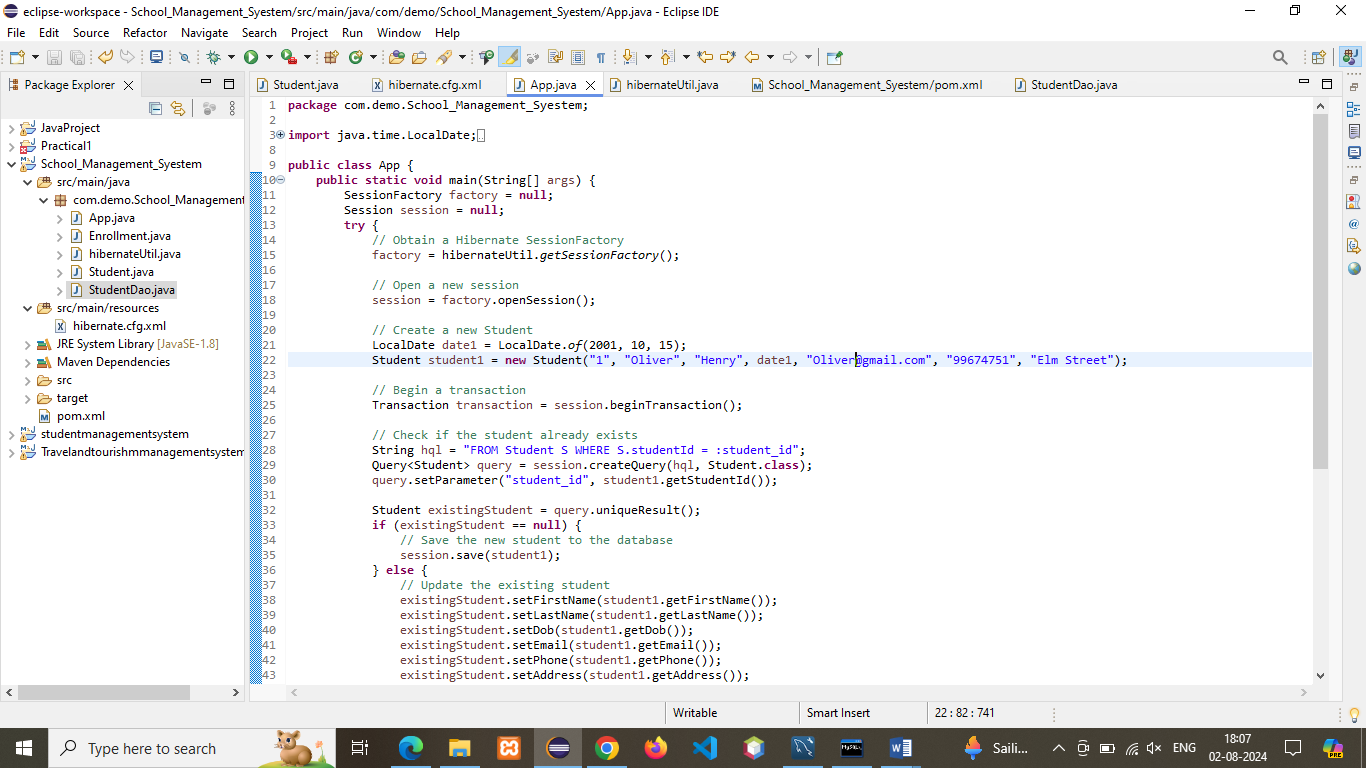
factory.close();

// Output the saved student details

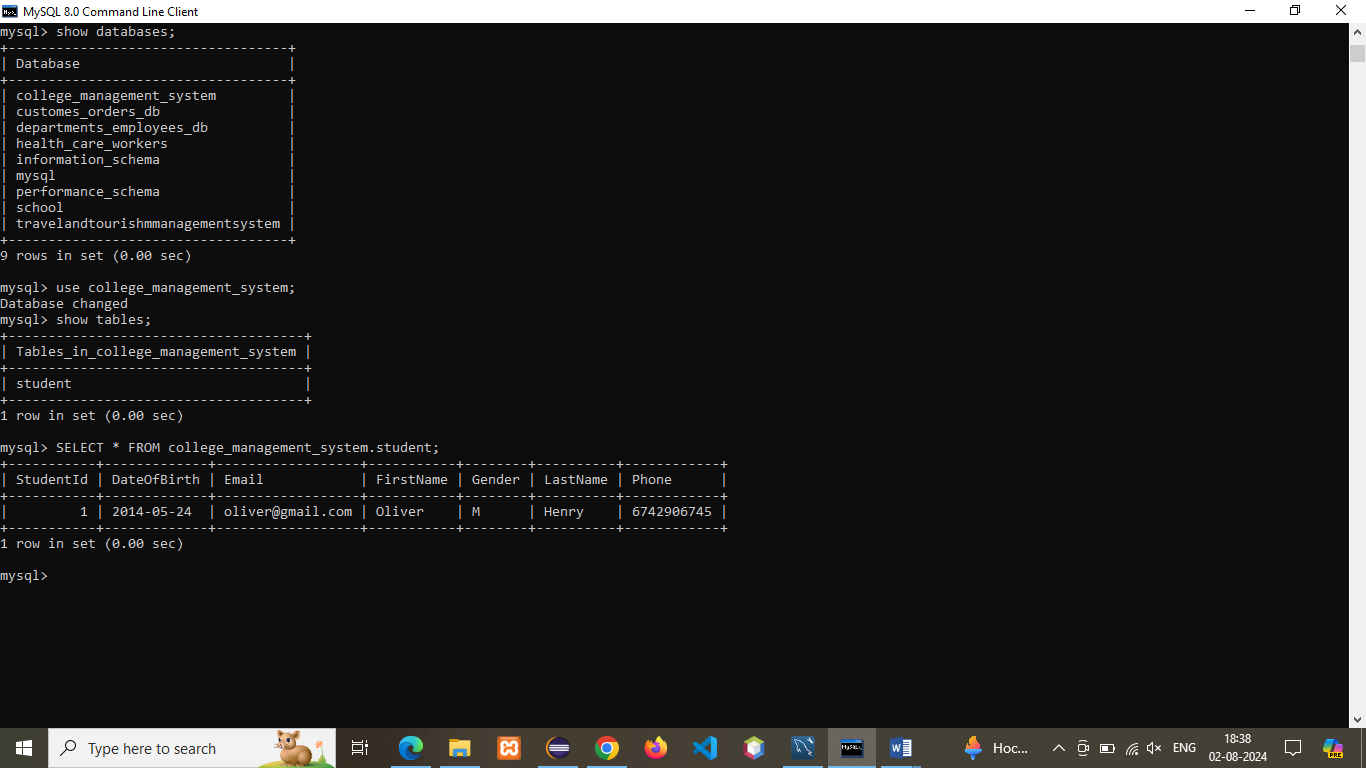
System.***out***.println("Saved student: " + student1);

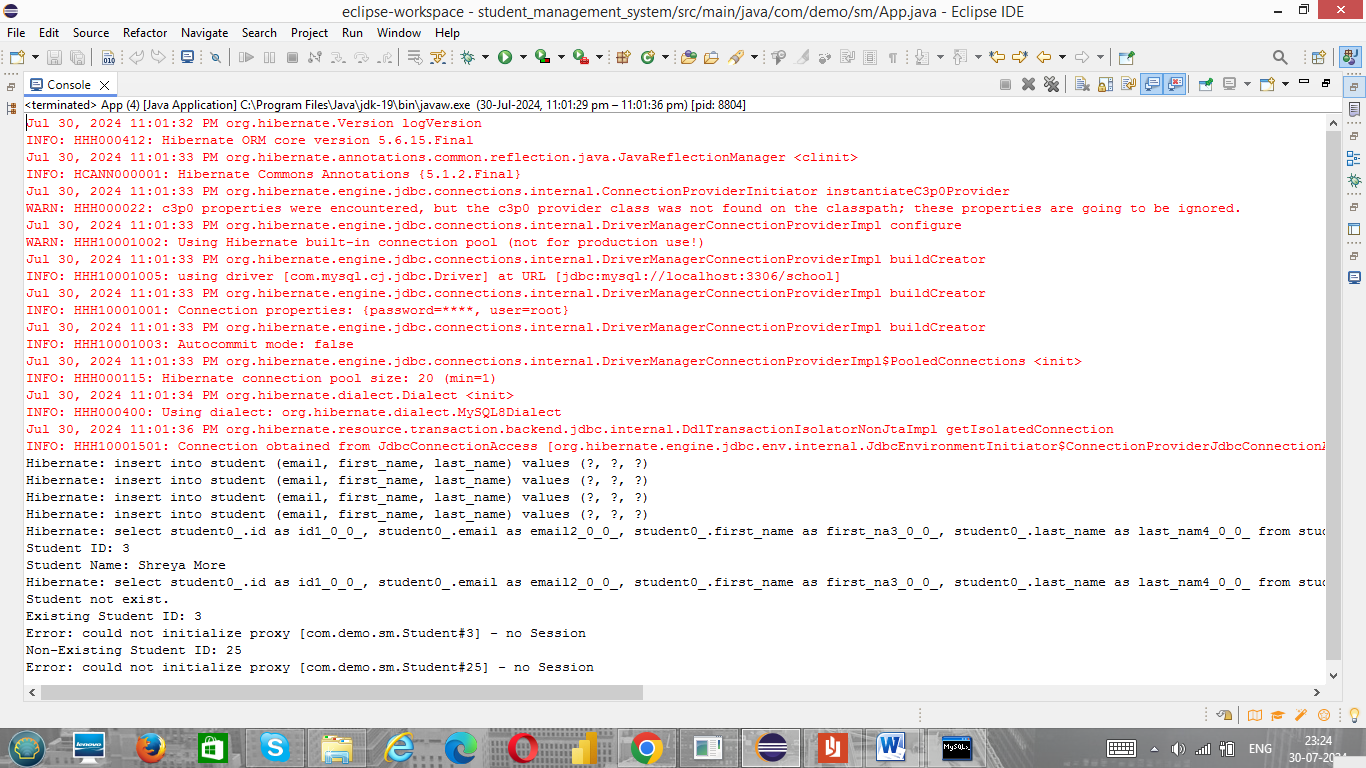
}

}



**Output:**





**Q2)     Use get() method to fetch a student object with an ID that doesn't exist in the database. What will be the result, and how would you handle it?**

**a)RetrieveRecord.java**

**package** com.demo.college\_management\_system;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**public** **class** RetrievingRecord {

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

// Obtain a Hibernate SessionFactory

SessionFactory factory = hibernateUtil.*getSessionFactory*();

Session session = **null**;

Transaction transaction = **null**;

**try** {

// Open a new session

session = factory.openSession();

// Begin a transaction

transaction = session.beginTransaction();

// Retrieve the object using the primary key

Student student = session.get(Student.**class**, "S111");

// Display data using toString() method

System.***out***.println(student);

// Commit the transaction

transaction.commit();

} **catch** (Exception e) {

**if** (transaction != **null**) {

transaction.rollback(); // Rollback transaction on error

}

e.printStackTrace();

} **finally** {

// Close the session

**if** (session != **null**) {

session.close();

}

// Close the session factory

**if** (factory != **null**) {

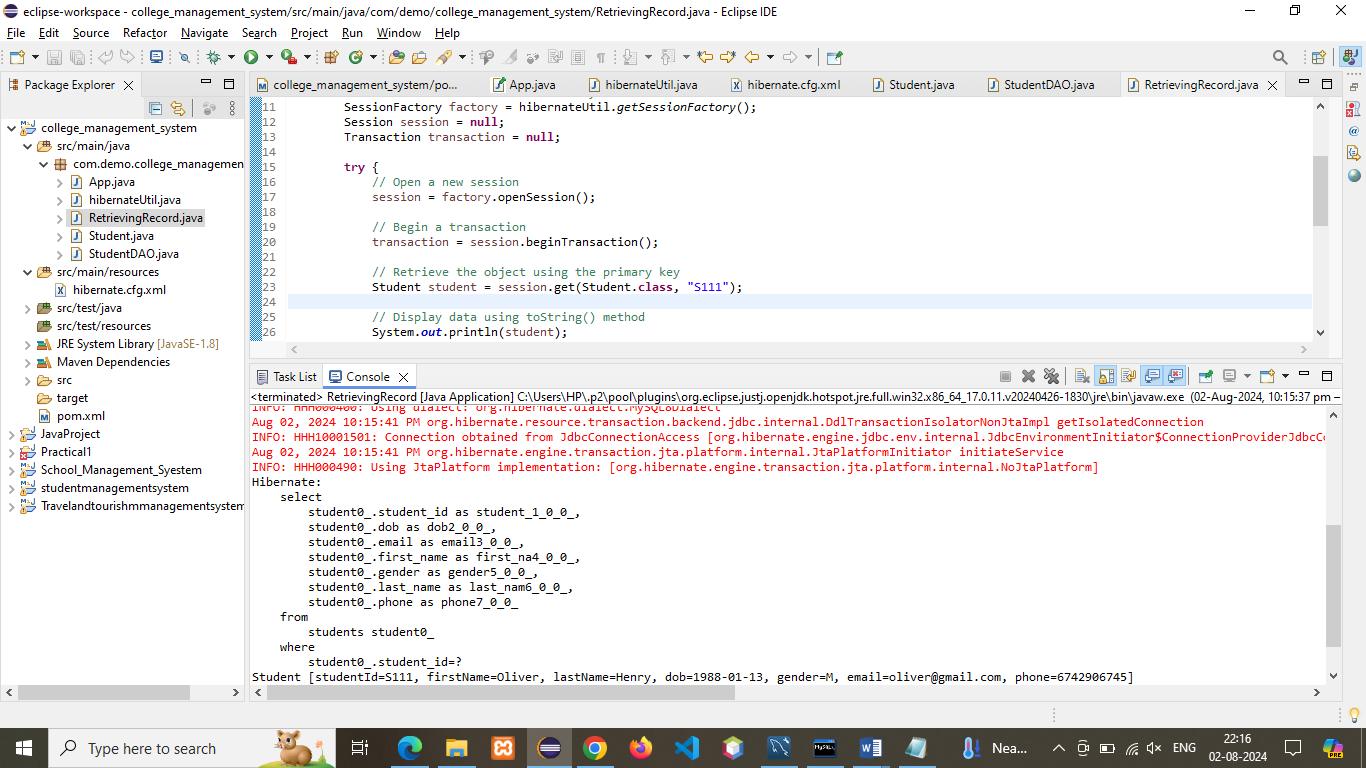
factory.close();

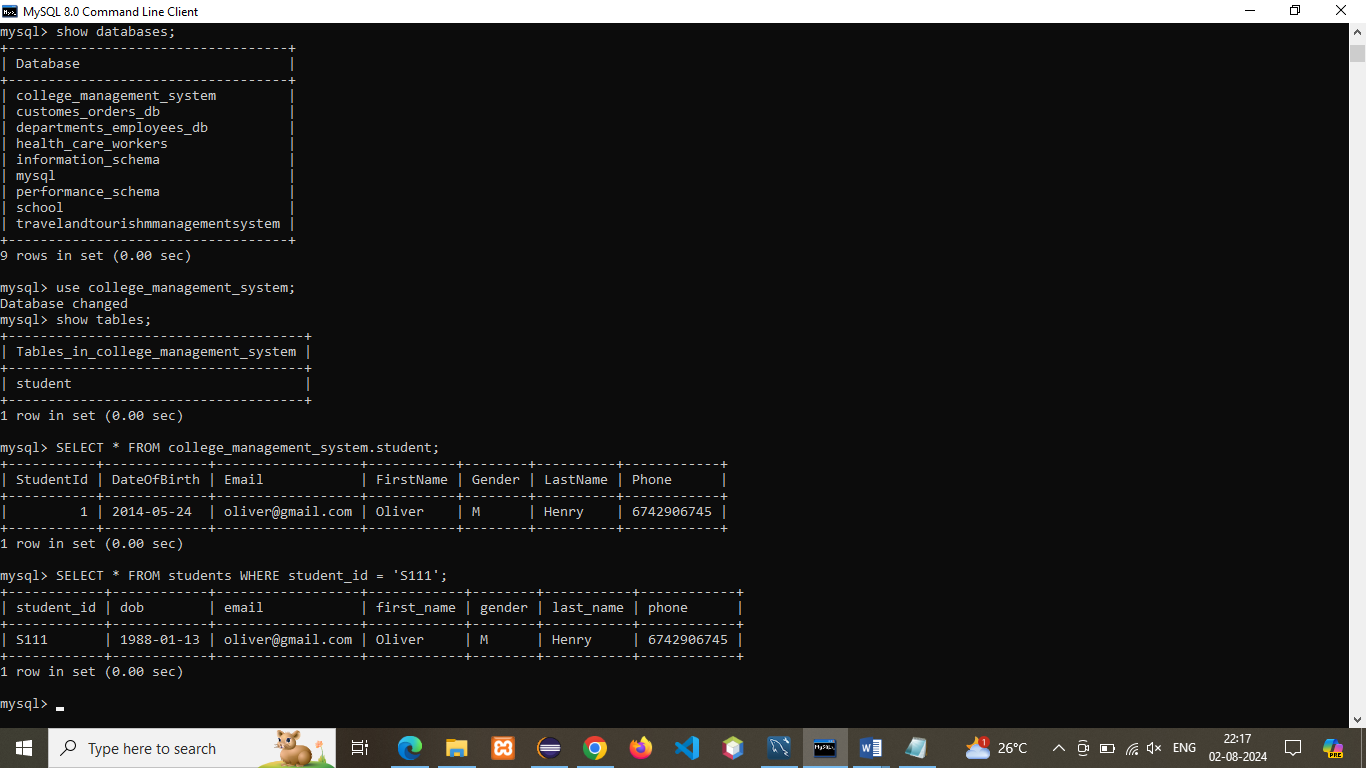
}

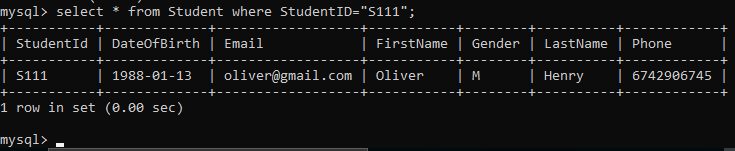
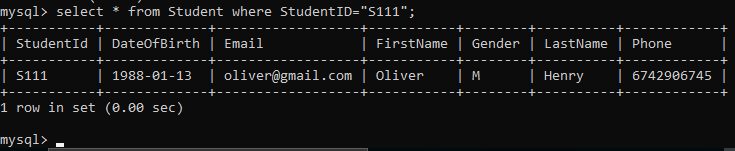
}

}

}







**Q3)     Also demonstrate use of load() method.**

**RetrievingRecordload**

**package** com.demo.college\_management\_system;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**public** **class** RetrievingRecordload {

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

// Obtain a Hibernate SessionFactory

SessionFactory factory = hibernateUtil.*getSessionFactory*();

Session session = **null**;

Transaction transaction = **null**;

**try** {

// Open a new session

session = factory.openSession();

// Begin a transaction

transaction = session.beginTransaction();

// Retrieve the object using the primary key

Student student = session.load(Student.**class**, "S111");

// Display data using toString() method

System.***out***.println(student);

// Commit the transaction

transaction.commit();

} **catch** (Exception e) {

**if** (transaction != **null**) {

transaction.rollback(); // Rollback transaction on error

}

e.printStackTrace();

} **finally** {

// Close the session

**if** (session != **null**) {

session.close();

}

// Close the session factory

**if** (factory != **null**) {

factory.close();

}

}

}

}

