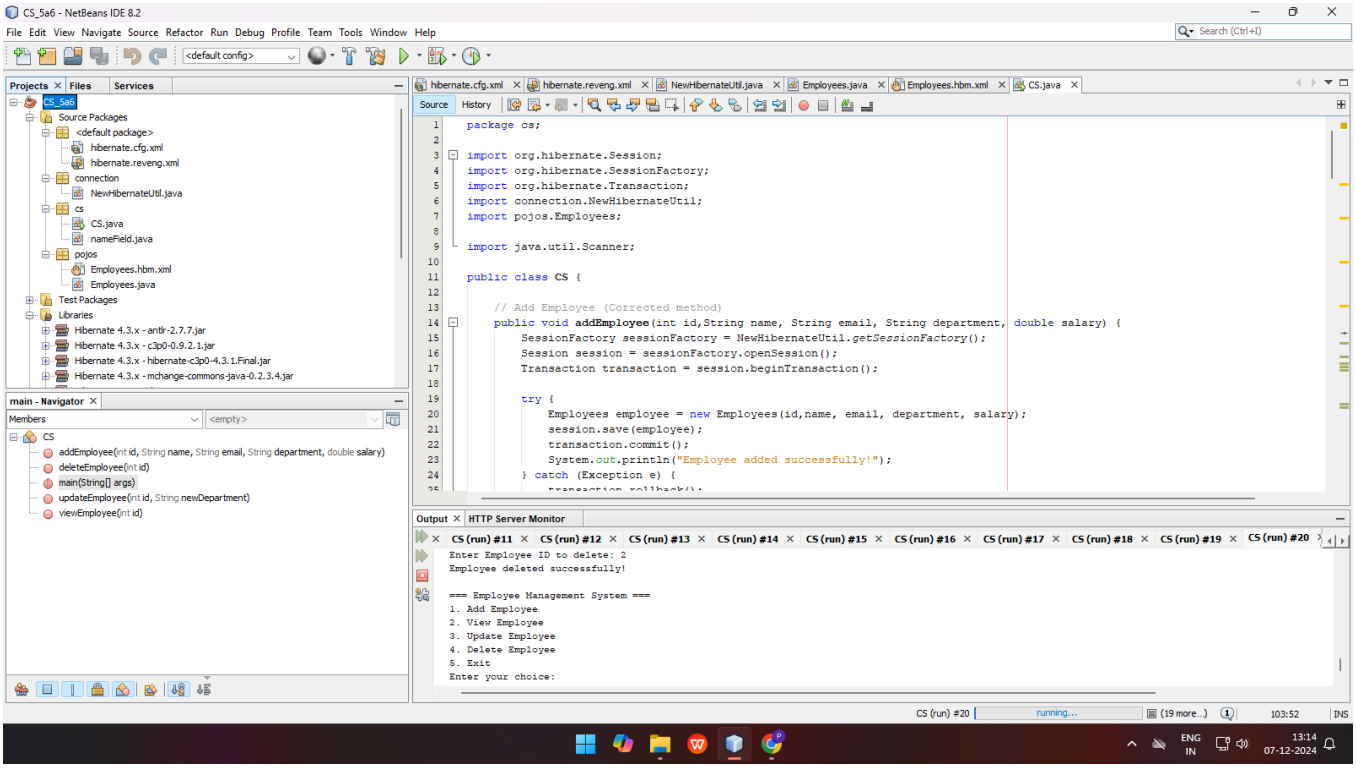
AIM: **Implement an employee management system using hibernate for orm(object relational mapping) and hql for quering the database.add,update,delete and view employee details**

[The below picture i.e. the screenshot contains the name as the name of the project itself. Please check below]



Cs.java

```
package cs;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import connection.NewHibernateUtil;
import pojos.Employees;

import java.util.Scanner;

public class CS {

    // Add Employee (Corrected method)
    public void addEmployee(int id,String name, String email, String department,
double salary) {
```

```
    SessionFactory sessionFactory = NewHibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction transaction = session.beginTransaction();

    try {
        Employees employee = new Employees(id,name, email, department, salary);
        session.save(employee);
        transaction.commit();
        System.out.println("Employee added successfully!");
    } catch (Exception e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
}

// View Employee
public void viewEmployee(int id) {
    SessionFactory sessionFactory = NewHibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    try {
        Employees employee = (Employees) session.get(Employees.class, id);
        if (employee != null) {
            System.out.println("Employee Details: " + employee);
        } else {
            System.out.println("No employee found with ID " + id);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        session.close();
    }
}

// Update Employee
public void updateEmployee(int id, String newDepartment) {
    SessionFactory sessionFactory = NewHibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction transaction = session.beginTransaction();
    try {
        Employees employee = (Employees) session.get(Employees.class, id);
        if (employee != null) {
            employee.setDepartment(newDepartment);
```

```
            session.update(employee);
            transaction.commit();
            System.out.println("Employee updated successfully!");
        } else {
            System.out.println("Employee with ID " + id + " not found!");
        }
    } catch (Exception e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
}

// Delete Employee
public void deleteEmployee(int id) {
    SessionFactory sessionFactory = NewHibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    Transaction transaction = session.beginTransaction();
    try {
        Employees employee = (Employees) session.get(Employees.class, id);
        if (employee != null) {
            session.delete(employee);
            transaction.commit();
            System.out.println("Employee deleted successfully!");
        } else {
            System.out.println("Employee with ID " + id + " not found!");
        }
    } catch (Exception e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
}

// Main Method for User Interaction
public static void main(String[] args) {
    CS operations = new CS();
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("\n=== Employee Management System ===");
        System.out.println("1. Add Employee");
```

```
System.out.println("2. View Employee");
System.out.println("3. Update Employee");
System.out.println("4. Delete Employee");
System.out.println("5. Exit");
System.out.print("Enter your choice: ");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
    case 1:
        // Add Employee
         System.out.print("Enter id: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter name: ");
        String name = scanner.nextLine();
        System.out.print("Enter email: ");
        String email = scanner.nextLine();
        System.out.print("Enter department: ");
        String department = scanner.nextLine();
        System.out.print("Enter salary: ");
        double salary = scanner.nextDouble();
        operations.addEmployee(id,name, email, department, salary);
        break;

    case 2:
        // View Employee
        System.out.print("Enter Employee ID to view: ");
        int viewId = scanner.nextInt();
        operations.viewEmployee(viewId);
        break;

    case 3:
        // Update Employee
        System.out.print("Enter Employee ID to update: ");
        int updateId = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter new department: ");
        String newDepartment = scanner.nextLine();
        operations.updateEmployee(updateId, newDepartment);
        break;

    case 4:
        // Delete Employee
```

```
                System.out.print("Enter Employee ID to delete: ");
                int deleteId = scanner.nextInt();
                operations.deleteEmployee(deleteId);
                break;

            case 5:
                // Exit
                System.out.println("Exiting... Goodbye!");
                scanner.close();
                NewHibernateUtil.getSessionFactory().close();
                System.exit(0);
                break;

            default:
                System.out.println("Invalid choice. Please try again!");
        }
    }
  }
}
```

Employees.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 3 Dec, 2024 6:52:41 PM by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
    <class name="pojos.Employees" table="employees" catalog="db" optimistic-
lock="version">
        <id name="id" type="int">
            <column name="id" />
            <generator class="identity" />
        </id>
        <property name="name" type="string">
            <column name="name" length="100" not-null="true" />
        </property>
        <property name="email" type="string">
            <column name="email" length="100" not-null="true" unique="true" />
        </property>
        <property name="department" type="string">
            <column name="department" length="100" />
        </property>
```

```
        <property name="salary" type="java.lang.Double">
            <column name="salary" precision="10" scale="2" />
        </property>
    </class>
</hibernate-mapping>
```

Employees.java

```
package pojos;

public class Employees implements java.io.Serializable {

    private int id;
    private String name;
    private String email;
    private String department;
    private Double salary;

    public Employees() {
    }

    // Updated constructor
    public Employees(int id,String name, String email, String department, Double
salary) {
        this.id=id;
        this.name = name;
        this.email = email;
        this.department = department;
        this.salary = salary;
    }

    public Employees(int id, String name, String email, String department, double
salary) {
        throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
    }

    public int getId() {
        return this.id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getName() {
        return this.name;
    }

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

    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getDepartment() {
        return this.department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    public Double getSalary() {
        return this.salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee[ID=" + id + ", Name=" + name + ", Email=" + email + ",
Department=" + department + ", Salary=" + salary + "]";
    }
}
```

NewHibernateUtil.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package connection;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;



/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author Sneha Sameera
 */
public class NewHibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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

Hibernate.reveng.xml

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

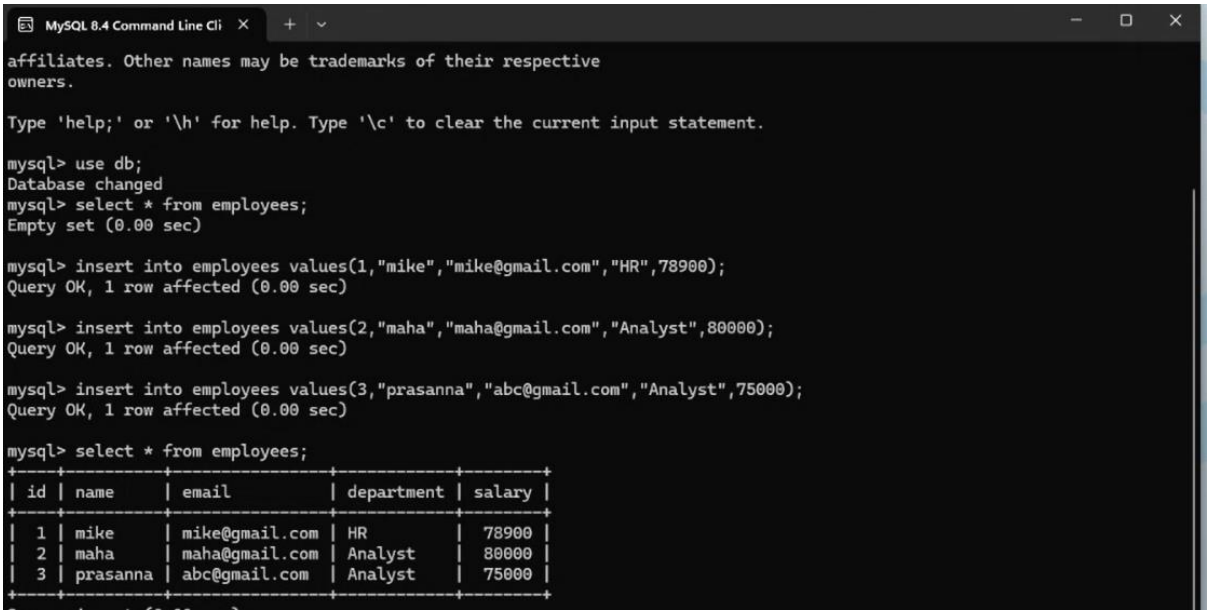
```
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate
Reverse Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
  <schema-selection match-catalog="db"/>
  <table-filter match-name="employees"/>
</hibernate-reverse-engineering>
```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/db?zeroDateTimeBeh
avior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">password</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <mapping class="pojos.Employees" />

    <mapping resource="pojos/Employees.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

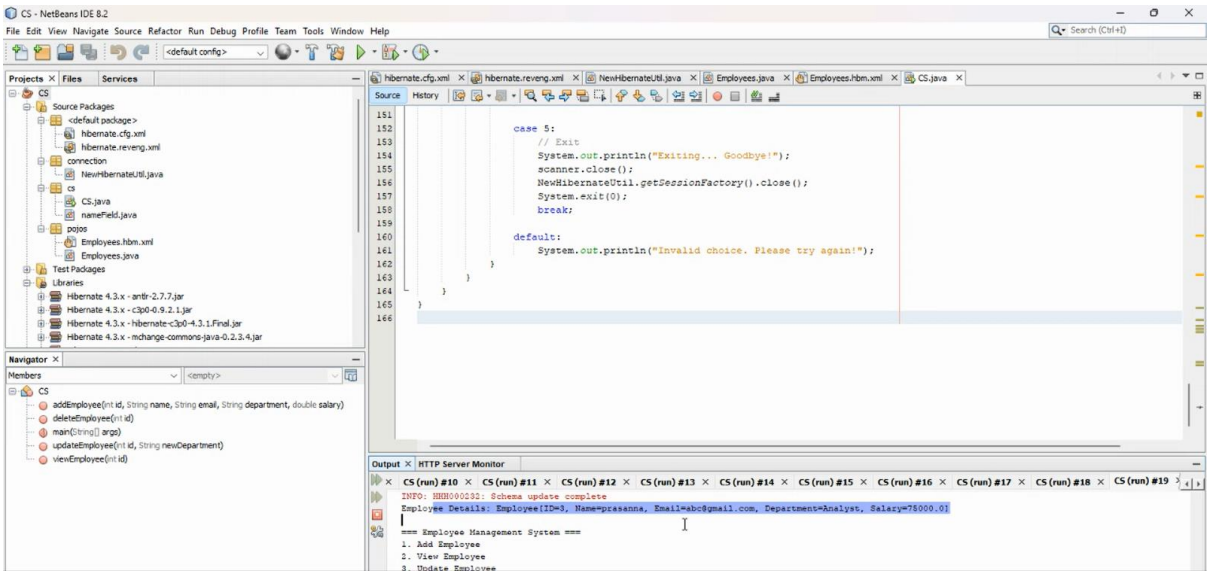
OUTPUTS:

view:



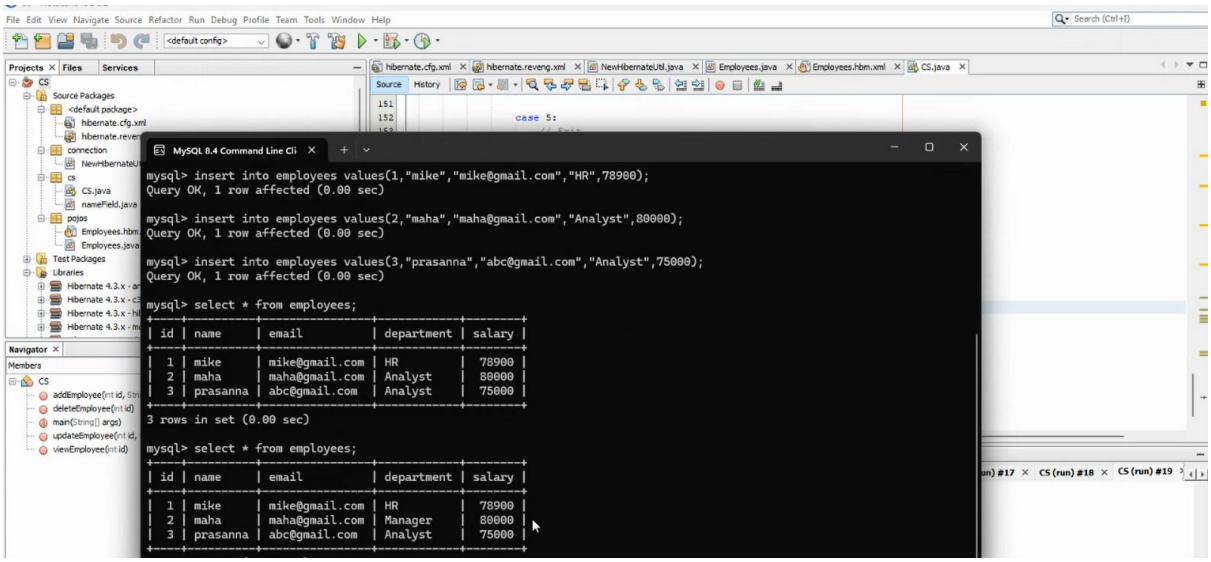
```
MySQL 8.4 Command Line Cli
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use db;
Database changed
mysql> select * from employees;
Empty set (0.00 sec)

mysql> insert into employees values(1,"mike","mike@gmail.com","HR",78900);
Query OK, 1 row affected (0.00 sec)

mysql> insert into employees values(2,"maha","maha@gmail.com","Analyst",80000);
Query OK, 1 row affected (0.00 sec)

mysql> insert into employees values(3,"prasanna","abc@gmail.com","Analyst",75000);
Query OK, 1 row affected (0.00 sec)

mysql> select * from employees;
+----+----------+----------------+------------+--------+
| id | name     | email          | department | salary |
+----+----------+----------------+------------+--------+
|  1 | mike     | mike@gmail.com | HR         |  78900 |
|  2 | maha     | maha@gmail.com | Analyst    |  80000 |
|  3 | prasanna | abc@gmail.com  | Analyst    |  75000 |
+----+----------+----------------+------------+--------+
```

add:



```
CS - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects | Files | Services
CS
  Source Packages
    <default package>
      hibernate.cfg.xml
      hibernate.reveng.xml
    connection
      NewHibernateUtil.java
    cs
      CS.java
      nameField.java
    pojos
      Employees.hbm.xml
      Employees.java
  Test Packages
  Libraries
    Hibernate 4.3.x - antlr-2.7.7.jar
    Hibernate 4.3.x - c3p0-0.9.2.1.jar
    Hibernate 4.3.x - hibernate-c3p0-4.3.1.Final.jar
    Hibernate 4.3.x - mchange-commons-java-0.2.3.4.jar

Navigator
Members
CS
  addEmployee(int id, String name, String email, String department, double salary)
  deleteEmployee(int id)
  main(String[] args)
  updateEmployee(int id, String newDepartment)
  viewEmployee(int id)

hibernate.cfg.xml | hibernate.reveng.xml | NewHibernateUtil.java | Employees.java | Employees.hbm.xml | CS.java
Source | History

151                 case 5:
152                     // Exit
153                     System.out.println("Exiting... Goodbye!");
154                     scanner.close();
155                     NewHibernateUtil.getSessionFactory().close();
156                     System.exit(0);
157                     break;
158
159                 default:
160                     System.out.println("Invalid choice. Please try again!");
161             }
162         }
163     }
164 }
165
166

Output | HTTP Server Monitor
CS (run) #10 | CS (run) #11 | CS (run) #12 | CS (run) #13 | CS (run) #14 | CS (run) #15 | CS (run) #16 | CS (run) #17 | CS (run) #18 | CS (run) #19
INFO: HHH000232: Schema update complete
Employee Details: Employee[ID=3, Name=prasanna, Email=abc@gmail.com, Department=Analyst, Salary=75000.0]

=== Employee Management System ===
1. Add Employee
2. View Employee
3. Update Employee
```

Update:



delete: