A.S.MOHAMED MUBEEN

HIBERNATE CODING CHALLENGE 1.0

1. A CRUD operation deals with creating, retrieving, updating and deleting from the table We have already described previously how to persist "Employee" Class to database. Here we are adding more operation on that Employee Class. private long empno; private String ename; private int sal; private String job; private int deptno ; a. Persisting the class to database b. Retrieving records from database c. Updating record d. Deleting record

```
/* starts from here hibernate coding challenge */

-- hibernate challenge 1.0—

CREATE TABLE employee
(
	empno int primary key,
	employeename varchar(40),
	salary int,
	job varchar(30),
	deptno varchar(30)

);
```

Empolyee.java

```
package com.model;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity(name = "Employee")
public class Employee implements Serializable {

    @Id
    private long empno;
    private String employeename;
    private int salary;
    private String job;
    private int deptno;

    public Employee() {
        // Default constructor
    }

      public long getEmpno() {
            return empno;
      }

      public void setEmpno(long empno) {
            this.empno = empno;
```

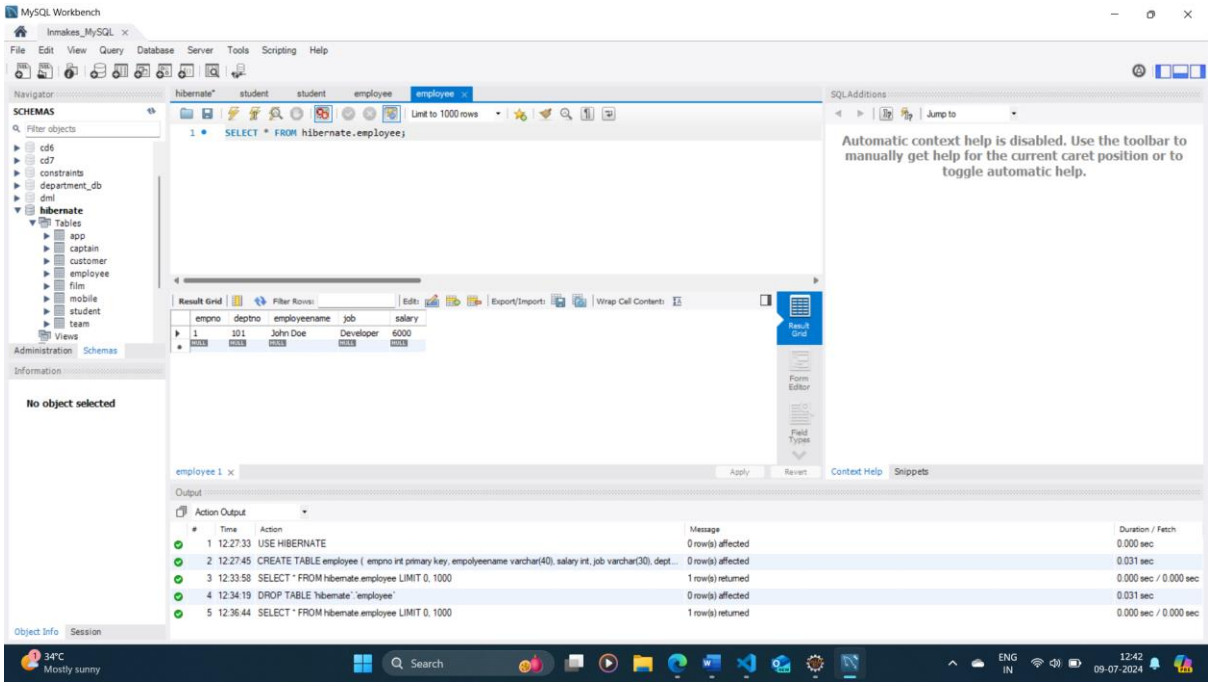
```
	}

	public String getEmployeename() {
		return employeename;
	}

	public void setEmployeename(String employeename) {
		this.employeename = employeename;
	}

	public int getSalary() {
		return salary;
	}

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

	public String getJob() {
		return job;
	}

	public void setJob(String job) {
		this.job = job;
	}

	public int getDeptno() {
		return deptno;
	}

	public void setDeptno(int deptno) {
		this.deptno = deptno;
	}

}
```

Persistingclass.java

```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.model.Employee;

public class Persistingclass {

    public static void main(String[] args) {

        Configuration cfg = new Configuration();
        cfg.configure("hibernate.cfg.xml");
        cfg.addAnnotatedClass(Employee.class);
        SessionFactory factory = cfg.buildSessionFactory();
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();


        Employee emp1 = new Employee();
        emp1.setEmpno(1);
```

```
        emp1.setEmployeename("John Doe");
        emp1.setSalary(6000);
        emp1.setJob("Developer");
        emp1.setDeptno(101);


        session.save(emp1);
        tx.commit();
        System.out.println("Record successfully inserted");
        session.close();

    }
}
```

Output: Employee_table



Retrievingrecords.java

```
package com.hibernatechallenge;

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

import com.model.Employee;

public class RetrievingRecords {

    public static void main(String[] args) {
        Configuration cfg = new
Configuration().configure("hibernate.cfg.xml");
        SessionFactory factory = cfg.buildSessionFactory();
        Session session = factory.openSession();

        // Retrieve an Employee instance
```

```
        Employee emp1 = session.get(Employee.class, 1L);
            System.out.println("Employee found:");
            System.out.println("Employee Number: " + emp1.getEmpno());
            System.out.println("Employee Name: " + emp1.getEmployeename());
            System.out.println("Salary: " + emp1.getSalary());
            System.out.println("Job: " + emp1.getJob());
            System.out.println("Department Number: " + emp1.getDeptno());
        session.close();
        factory.close();
    }
}
```

Output:

```
Hibernate: select employee0_.empno as empno1_0_0_, employee0_.deptno as
deptno2_0_0_, employee0_.employeename as employee3_0_0_, employee0_.job as
job4_0_0_, employee0_.salary as salary5_0_0_ from Employee employee0_ where
employee0_.empno=?
----------------
Employee found:
Employee Number: 1
Employee Name: John Doe
Salary: 6000
Job: Developer
Department Number: 101
```

Updaterecord.java

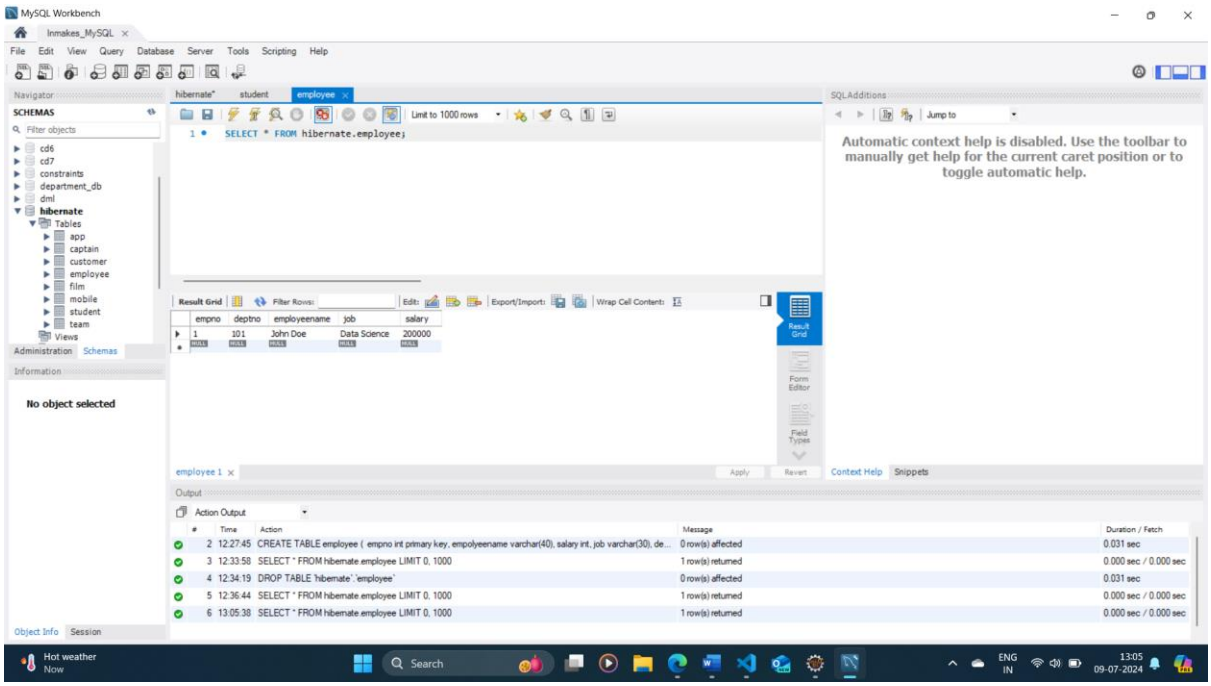
```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import com.model.Customer;
import com.model.Employee;

public class Updaterecord {

      public static void main(String[] args) {

            Configuration cfg=new Configuration();
            cfg.configure("hibernate.cfg.xml");

            SessionFactory factory=cfg.buildSessionFactory();
            Session session=factory.openSession();
            Transaction tx=session.beginTransaction();

             Employee emp1 = session.get(Employee.class, 1L);
```

```
            emp1.setJob("Data Science");
            emp1.setSalary(200000);

            session.update(emp1);
            tx.commit();
            System.out.println("record update sucessfully");
            session.close();



      }

}
```

Output:

```
Hibernate: select employee0_.empno as empno1_0_0_, employee0_.deptno as
deptno2_0_0_, employee0_.employeename as employee3_0_0_, employee0_.job as
job4_0_0_, employee0_.salary as salary5_0_0_ from Employee employee0_ where
employee0_.empno=?
Hibernate: update Employee set deptno=?, employeename=?, job=?, salary=?
where empno=?
record update sucessfully
```

Employee_table:



Deletingrecord.java

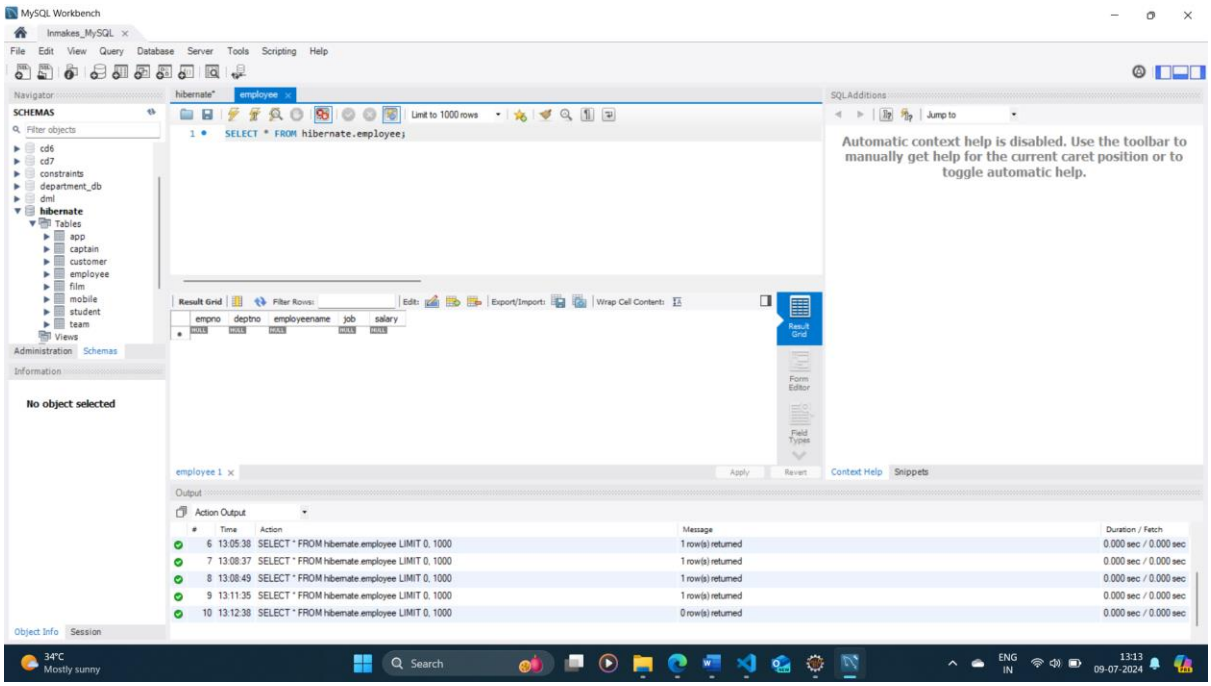
```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
```

```java
import com.model.Employee;

public class Deletingrecord {

	public static void main(String[] args) {

		Configuration cfg=new Configuration();
		cfg.configure("hibernate.cfg.xml");

		SessionFactory factory=cfg.buildSessionFactory();
		Session session=factory.openSession();
		Transaction tx=session.beginTransaction();

		 Employee emp1 = session.get(Employee.class, 1L);

		 session.delete(emp1);
		 session.getTransaction().commit();
		 System.out.println("record deleted sucessfully");
		 session.close();

	}

}
```

Output:

```
Hibernate: select employee0_.empno as empno1_0_0_, employee0_.deptno as
deptno2_0_0_, employee0_.employeename as employee3_0_0_, employee0_.job as
job4_0_0_, employee0_.salary as salary5_0_0_ from Employee employee0_ where
employee0_.empno=?
Hibernate: delete from Employee where empno=?
record deleted sucessfully
```

Employee_table:



HIBERNATE CODING CHALLENGE 2.0

2. HQL (Hibernate Query Language) a. Retrieving the records based on Employee name Starts with letter 'A'. b. Retrieving the records whose salary in between 5000 to 7000. c. Retrieving 2nd minimum and 2nd maximum salaries. d. Write an SQL query to fetch the list of employees with the same salary and update the list of employee salary to 5000.

```
-- hibernate challenge 2.0--
INSERT INTO employee (empno, employeename, salary, job, deptno)
VALUES
(1, 'John Doe', 5000, 'Manager', '101'),
(2, 'Jane Smith', 45000, 'Developer', '101'),
(3, 'Michael Johnson', 6000, 'Analyst', '104'),
(4, 'Emily Davis', 55000, 'Designer', '103');
```

StartsA.java

```
package com.hibernatechallenge;

import java.util.ArrayList;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;

import com.model.Customer;
import com.model.Employee;

public class StartsA {

	public static void main(String[] args) {

		try {

			Configuration cfg=new Configuration();
			cfg.configure("hibernate.cfg.xml");

			SessionFactory factory=cfg.buildSessionFactory();
			Session session=factory.openSession();
			Transaction tx=session.beginTransaction();

			Query query = session.createQuery("from Employee where
employeename like 'A%'");
			ArrayList<Employee> empdet = (ArrayList<Employee>)
query.list();
			System.out.println("Employee details starting with A");
			System.out.println("-------------------------");
			for(Employee emp : empdet) {
			    System.out.println(emp.getEmpno() + "\t" +
emp.getEmployeename() + "\t" + emp.getSalary() + "\t" + emp.getJob() + "\t"
+ emp.getDeptno());
			}
```

```
            }
            catch(Exception e){

                  e.printStackTrace();
            }

      }

}
```

Output:

```
Hibernate: select employee0_.empno as empno1_0_, employee0_.deptno as
deptno2_0_, employee0_.employeename as employee3_0_, employee0_.job as
job4_0_, employee0_.salary as salary5_0_ from Employee employee0_ where
employee0_.employeename like 'A%'
Employee details starting with A
-------------------------
```

Salbetween.java

```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.criterion.Restrictions;

import com.model.Employee;

import java.util.List; // Correct import for List

public class Salbetween {

    public static void main(String[] args) {

        try {

            Configuration cfg = new Configuration();
            cfg.configure("hibernate.cfg.xml");

            SessionFactory factory = cfg.buildSessionFactory();
            Session session = factory.openSession();
            Transaction tx = session.beginTransaction();

            List<Employee> employees =
session.createCriteria(Employee.class)
                    .add(Restrictions.between("salary", 5000, 7000))
                    .list();


            for (Employee employee : employees) {
```

```
                System.out.println("Employee ID: " + employee.getEmpno()
                        + ", Name: " + employee.getEmployeename()
                        + ", Salary: " + employee.getSalary());
            }

            tx.commit();
            session.close();
            factory.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Hibernate: select this_.empno as empno1_0_0_, this_.deptno as deptno2_0_0_,
this_.employeename as employee3_0_0_, this_.job as job4_0_0_, this_.salary
as salary5_0_0_ from Employee this_ where this_.salary between ? and ?
Employee ID: 1, Name: John Doe, Salary: 5000
Employee ID: 3, Name: Michael Johnson, Salary: 6000
```

Maximumsalaries.java

```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;


import com.model.Employee;

import java.util.List;

public class Maximumsalaries {

    public static void main(String[] args) {

        try {

            Configuration cfg = new Configuration();
            cfg.configure("hibernate.cfg.xml");

            SessionFactory factory = cfg.buildSessionFactory();
            Session session = factory.openSession();
            Transaction tx = session.beginTransaction();

            String hqlMin = "from Employee e where 2 = (select
count(distinct salary) from Employee where salary < e.salary)";
```

```
            Employee secondMinSalary = session.createQuery(hqlMin,
Employee.class).uniqueResult();
            System.out.println("2nd Minimum Salary: " +
secondMinSalary.getSalary());

            String hqlMax = "from Employee e where 2 = (select
count(distinct salary) from Employee where salary > e.salary)";
            Employee secondMaxSalary = session.createQuery(hqlMax,
Employee.class).uniqueResult();
            System.out.println("2nd Maximum Salary: " +
secondMaxSalary.getSalary());


            tx.commit();
            session.close();
            factory.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Hibernate: select employee0_.empno as empno1_0_, employee0_.deptno as
deptno2_0_, employee0_.employeename as employee3_0_, employee0_.job as
job4_0_, employee0_.salary as salary5_0_ from Employee employee0_ where
2=(select count(distinct employee1_.salary) from Employee employee1_ where
employee1_.salary<employee0_.salary)
2nd Minimum Salary: 45000
Hibernate: select employee0_.empno as empno1_0_, employee0_.deptno as
deptno2_0_, employee0_.employeename as employee3_0_, employee0_.job as
job4_0_, employee0_.salary as salary5_0_ from Employee employee0_ where
2=(select count(distinct employee1_.salary) from Employee employee1_ where
employee1_.salary>employee0_.salary)
2nd Maximum Salary: 6000
```

Updatesalary.java

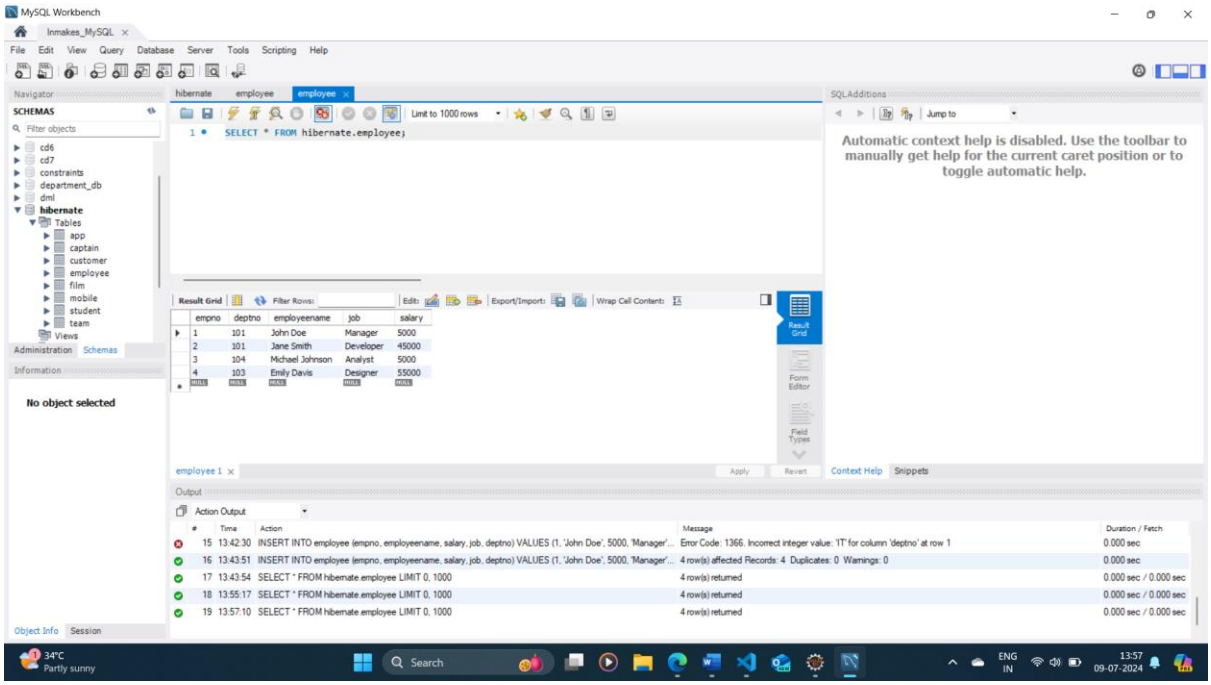
```
package com.hibernatechallenge;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
```

```
import org.hibernate.criterion.Restrictions;

import com.model.Employee;

public class Updatesalary {

	public static void main(String[] args) {

		try {

			Configuration cfg = new Configuration();
			cfg.configure("hibernate.cfg.xml");

			SessionFactory factory = cfg.buildSessionFactory();
			Session session = factory.openSession();
			Transaction tx = session.beginTransaction();

			String hqlUpdate = "update Employee set salary = 5000
where salary = :oldSalary";
			int updatedEntities = session.createQuery(hqlUpdate)
					.setParameter("oldSalary",
6000)
					.executeUpdate();

			tx.commit();
			session.close();
			factory.close();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	}
```

Output:

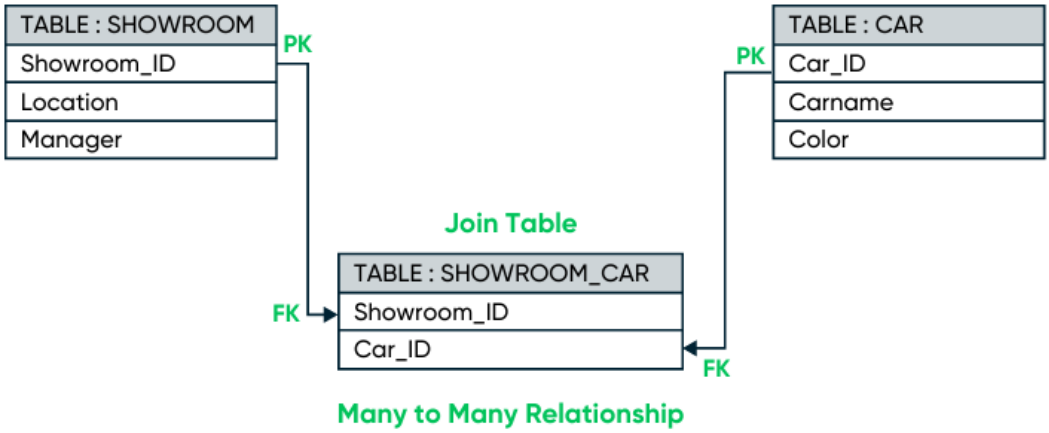
```
Hibernate: update Employee set salary=5000 where salary=?
```

Employee_table:



HIBERNATE CODING CHALLENGE 3.0

1. Many to Many Mapping for the following Tables



Showrrom.java

```java
package com.model;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name ="showroom")
public class Showroom implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int showroom_id;
    private String location;
    private String manager;

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(
        name = "SHOWROOM_CAR",
        joinColumns = @JoinColumn(name = "SHOWROOM_ID"),
        inverseJoinColumns = @JoinColumn(name = "CAR_ID")
    )
    private Set<Car> cars = new HashSet<>();

    public Showroom() {
    }

    public int getShowroom_id() {
        return showroom_id;
    }

    public void setShowroom_id(int showroom_id) {
        this.showroom_id = showroom_id;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getManager() {
        return manager;
    }

    public void setManager(String manager) {
```

```
        this.manager = manager;
    }

    public Set<Car> getCars() {
        return cars;
    }

    public void setCars(Set<Car> cars) {
        this.cars = cars;
    }
}
```

Car.java

```
package com.model;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "Car")
public class Car implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int car_id;
    private String carname;
    private String color;

    @ManyToMany(mappedBy = "cars", cascade = CascadeType.ALL)
    private Set<Showroom> showrooms = new HashSet<>();

    public Car() {
    }

    public int getCar_id() {
        return car_id;
    }

    public void setCar_id(int car_id) {
        this.car_id = car_id;
    }

    public String getCarname() {
        return carname;
    }

    public void setCarname(String carname) {
        this.carname = carname;
    }

    public String getColor() {
```

```
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public Set<Showroom> getShowrooms() {
        return showrooms;
    }

    public void setShowrooms(Set<Showroom> showrooms) {
        this.showrooms = showrooms;
    }
}
```

CarDao.java

```
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import com.model.Car;


public class CarDao {

public  void save(Car c ) {

      try {
            Configuration cfg=new Configuration();
            cfg.configure();

            SessionFactory factory=cfg.buildSessionFactory();
            Session session=factory.openSession();
            Transaction tx=session.beginTransaction();
            session.save(c);
            tx.commit();
            session.close();
      }
      catch(Exception e) {
            e.printStackTrace();
      }


      }
}
```

ShowroomDao.java

```java
package com.hibernatechallenge;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.model.Showroom;

public class ShowroomDao {

    public void save(Showroom showroom) {
        try {
            Configuration cfg = new Configuration();
            cfg.configure();

            SessionFactory factory = cfg.buildSessionFactory();
            Session session = factory.openSession();
            Transaction tx = session.beginTransaction();
            session.save(showroom);
            tx.commit();
            session.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Cartest.java

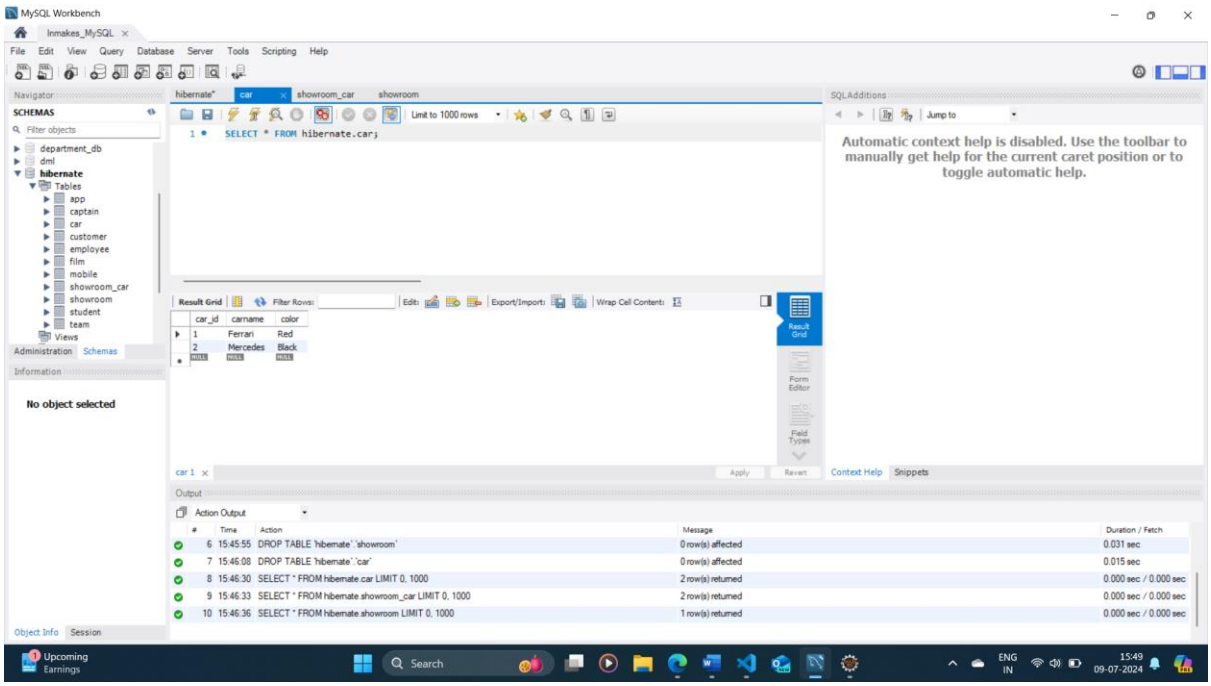
```java
package com.hibernatechallenge;

import java.util.HashSet;
import java.util.Set;

import com.model.Car;
import com.model.Showroom;

public class Cartest {

    public static void main(String[] args) {
        Car c1 = new Car();
        c1.setCarname("Ferrari");
        c1.setColor("Red");

        Car c2 = new Car();
        c2.setCarname("Mercedes");
        c2.setColor("Black");

        Showroom showroom = new Showroom();
        showroom.setLocation("Chennai");
        showroom.setManager("Ramesh");

        Set<Car> cars = new HashSet<>();
        cars.add(c1);
        cars.add(c2);
```

```
        showroom.setCars(cars);
        c1.getShowrooms().add(showroom);
        c2.getShowrooms().add(showroom);

        ShowroomDao showroomDao = new ShowroomDao();
        showroomDao.save(showroom);
    }
}
```

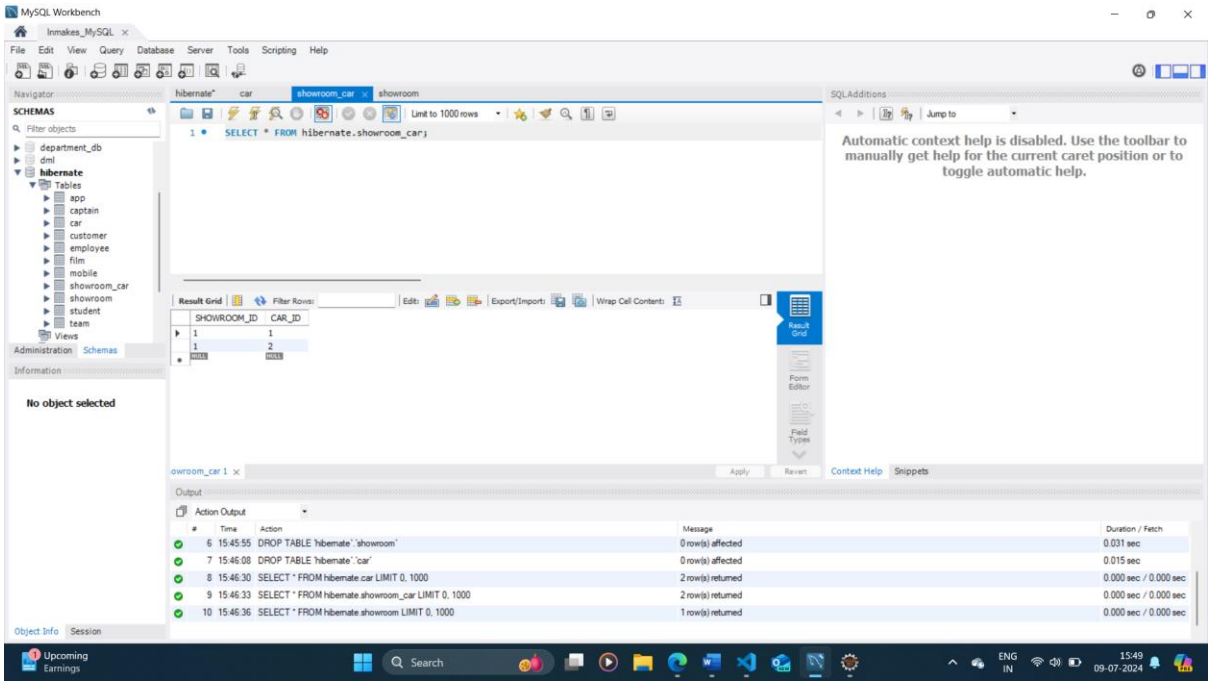
Output:

```
Hibernate: create table Car (car_id integer not null auto_increment,
carname varchar(255), color varchar(255), primary key (car_id))
engine=InnoDB
Hibernate: create table showroom (showroom_id integer not null
auto_increment, location varchar(255), manager varchar(255), primary key
(showroom_id)) engine=InnoDB
Hibernate: create table SHOWROOM_CAR (SHOWROOM_ID integer not null, CAR_ID
integer not null, primary key (SHOWROOM_ID, CAR_ID)) engine=InnoDB
Hibernate: alter table SHOWROOM_CAR add constraint
FKftd7tina4q60lcrvrvuq2nv9o foreign key (CAR_ID) references Car (car_id)
Hibernate: alter table SHOWROOM_CAR add constraint
FKgv4j7a5cpndvkn4gbcqs2p3oe foreign key (SHOWROOM_ID) references showroom
(showroom_id)
Hibernate: insert into showroom (location, manager) values (?, ?)
Hibernate: insert into Car (carname, color) values (?, ?)
Hibernate: insert into Car (carname, color) values (?, ?)
Hibernate: insert into SHOWROOM_CAR (SHOWROOM_ID, CAR_ID) values (?, ?)
Hibernate: insert into SHOWROOM_CAR (SHOWROOM_ID, CAR_ID) values (?, ?)
```

Car_table:



Showroom_car_table:



showroom_table: