```
------------------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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ameya</groupId>
  <artifactId>019-jdbcproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.26</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.zaxxer/HikariCP -->
    <dependency>
        <groupId>com.zaxxer</groupId>
        <artifactId>HikariCP</artifactId>
        <version>5.0.0</version>
    </dependency>
  </dependencies>
</project>
----------------hikaripool.properties---------------
CONURL=jdbc:mysql://localhost:3306/sapientdb
DRIVERCLASSNAME=com.mysql.cj.jdbc.Driver
DBUSERNAME=root
DBPASSWORD=root
CACHEPREPSTMTS=true
PREPSTMTCACHESIZE=250
PREPSTMTCACHESQLLIMIT=2048
MAXPOOLSIZE=50
----------------HikariPoolHelper.java--------------
package com.ameya.helpers;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

public class HikariPoolHelper {
```

```
46
47      private HikariDataSource ds;
48      private HikariConfig configure() {
49          HikariConfig config=null;
50          try {
51              Properties properties=new Properties();
52              InputStream inputStream=getClass()
53                      .getClassLoader()
54                      .getResourceAsStream("hikaripool.properties");
55              if(inputStream==null) {
56                  throw new IOException("FILE NOT FOUND");
57              }
58              properties.load(inputStream);
59              config=new HikariConfig();
60              config.setDriverClassName(properties.getProperty("DRIVERCLASSNAME"));
61              config.setJdbcUrl(properties.getProperty("CONURL"));
62              config.setUsername(properties.getProperty("DBUSERNAME"));
63              config.setPassword(properties.getProperty("DBPASSWORD"));
64
                config.setMaximumPoolSize(Integer.parseInt(properties.getProperty("MAXPO
                OLSIZE")));
65              config.addDataSourceProperty("cachePrepStmts",
                properties.getProperty("CACHEPREPSTMTS"));
66              config.addDataSourceProperty("prepStmtCacheSize",
                properties.getProperty("PREPSTMTCACHESIZE"));
67              config.addDataSourceProperty("prepStmtCacheSqlLimit",
                properties.getProperty("PREPSTMTCACHESQLLIMIT"));
68
69          }catch(Exception e) {
70              e.printStackTrace();
71          }
72
73          return config;
74      }
75      public HikariDataSource getDataSource() {
76          ds=new HikariDataSource(configure());
77          return ds;
78      }
79  }
80  ------------------HikariPoolEmployeeDAOImpl.java--------------
81  package com.ameya.daos.impl;
82
83  import java.sql.Connection;
84  import java.sql.PreparedStatement;
85  import java.sql.ResultSet;
86  import java.sql.SQLException;
87  import java.sql.Savepoint;
88  import java.util.ArrayList;
```

```
89  import java.util.List;
90
91  import com.ameya.daos.EmployeeDAO;
92  import com.ameya.domain.Employee;
93  import com.ameya.helpers.HikariPoolHelper;
94  import com.zaxxer.hikari.HikariDataSource;
95
96  public class HikariPoolEmployeeDAOImpl implements EmployeeDAO {
97      private HikariPoolHelper helper=null;
98      private HikariDataSource ds=null;
99      private Connection con;
100     private PreparedStatement ps;
101     public HikariPoolEmployeeDAOImpl() {
102         helper=new HikariPoolHelper();
103         ds=helper.getDataSource();
104     }
105     @Override
106     public void createConnection() {
107         try {
108             con=ds.getConnection();
109             System.out.println("## Connected To DB ##");
110         } catch (SQLException e) {
111             e.printStackTrace();
112         }
113
114     }
115
116     @Override
117     public void addEmployee(Employee employee) {
118         final String SQL = "insert into employee values(?,?,?,?,?)";
119         createConnection();
120         try {
121             ps = con.prepareStatement(SQL);
122             ps.setInt(1, employee.getEmpId());
123             ps.setString(2, employee.getFirstName());
124             ps.setString(3, employee.getLastName());
125             ps.setDouble(4, employee.getSalary());
126             ps.setInt(5, employee.getAge());
127             int cnt = ps.executeUpdate();
128             if (cnt != 0) {
129                 System.out.println("## Row Inserted Into Employee Table ##");
130             }
131         } catch (SQLException e) {
132             e.printStackTrace();
133         } finally {
134             closeConnection();
135         }
136
```

```
137     }
138
139     @Override
140     public Employee getEmployee(int empId) {
141         Employee employee = null;
142         final String SQL = "select * from employee where empid = ?";
143         createConnection();
144         try {
145             ps = con.prepareStatement(SQL);
146             ps.setInt(1, empId);
147             ResultSet rs = ps.executeQuery();
148             if (rs.next()) {
149                 employee = new Employee();
150                 employee.setEmpId(rs.getInt("empid"));
151                 employee.setFirstName(rs.getString("firstname"));
152                 employee.setLastName(rs.getString("lastname"));
153                 employee.setSalary(rs.getDouble("salary"));
154                 employee.setAge(rs.getInt("age"));
155             }
156         } catch (SQLException e) {
157             e.printStackTrace();
158         } finally {
159             closeConnection();
160         }
161
162         return employee;
163     }
164
165     @Override
166     public List<Employee> getAllEmployees() {
167         final String SQL = "select * from employee";
168         ArrayList<Employee> employees = new ArrayList<Employee>();
169         createConnection();
170         try {
171             ps = con.prepareStatement(SQL);
172             ResultSet rs = ps.executeQuery();
173             while (rs.next()) {
174                 employees.add(new Employee(rs.getInt("empId"),
175                         rs.getString("firstname"), rs.getString("lastname"),
176                         rs.getDouble("salary"), rs.getInt("age")));
177             }
178         } catch (SQLException e) {
179             e.printStackTrace();
180         } finally {
181             closeConnection();
182         }
183         return employees;
184     }
```

```
185
186     @Override
187     public void updateEmployee(Employee employee) {
188         final String SQL = "update employee set firstname = ? , lastname = ? , "
189                 + "salary = ? , age = ? where empid = ?";
190         createConnection();
191         try {
192             ps = con.prepareStatement(SQL);
193             ps.setString(1, employee.getFirstName());
194             ps.setString(2, employee.getLastName());
195             ps.setDouble(3, employee.getSalary());
196             ps.setInt(4, employee.getAge());
197             ps.setInt(5, employee.getEmpId());
198             int cnt = ps.executeUpdate();
199             if (cnt != 0) {
200                 System.out.println("## Employee record Updated ##");
201             }
202         } catch (SQLException e) {
203             e.printStackTrace();
204         } finally {
205             closeConnection();
206         }
207     }
208
209     @Override
210     public void deleteEmployee(int empId) {
211         final String SQL = "delete from employee where empid = ?";
212         createConnection();
213         try {
214             ps = con.prepareStatement(SQL);
215             ps.setInt(1, empId);
216             int cnt = ps.executeUpdate();
217             if (cnt != 0) {
218                 System.out.println("## Employee Record Deleted ##");
219             }
220         } catch (SQLException e) {
221             e.printStackTrace();
222         } finally {
223             closeConnection();
224         }
225
226     }
227
228     @Override
229     public void closeConnection() {
230         if (con != null) {
231             try {
232                 con.close();
```

```
233                 System.out.println("## DB Connection Closed ##");
234             } catch (SQLException e) {
235                 e.printStackTrace();
236             }
237         }
238 
239     }
240 
241     @Override
242     public void transactions() {
243         final String SQL = "insert into employee values(?,?,?,?,?)";
244         Employee e1=new Employee(101, "AAAA", "AAAA", 12345, 44);
245         Employee e2=new Employee(102, "BBBB", "BBBB", 23456, 42);
246         Savepoint s1=null; //Add after the rollback demo;
247         createConnection();
248         try {
249             con.setAutoCommit(false);
250             java.sql.DatabaseMetaData dmd=con.getMetaData();
251 
                if(dmd.supportsTransactionIsolationLevel(Connection.TRANSACTION_SERIAL
                IZABLE))
252             {
253                 System.out.println("Current Isolation Level :
                    "+con.getTransactionIsolation());
254                 con.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
255                 System.out.println("Isolation Level Set To :
                    "+con.getTransactionIsolation());
256             }
257             ps = con.prepareStatement(SQL);
258             ps.setInt(1, e1.getEmpId());
259             ps.setString(2, e1.getFirstName());
260             ps.setString(3, e1.getLastName());
261             ps.setDouble(4, e1.getSalary());
262             ps.setInt(5, e1.getAge());
263             ps.executeUpdate();
264             s1=con.setSavepoint("s1"); //Add after the rollback demo
265             ps.clearParameters();
266             //Erroneous insert
267             ps.setInt(1, e2.getEmpId());
268             ps.setString(2, e2.getFirstName());
269             ps.setString(3, e2.getLastName());
270             ps.setDouble(4, e2.getSalary());
271             //ps.setInt(5, e2.getAge());
272             ps.executeUpdate();
273             con.commit();
274             System.out.println("Transaction Committed Successfully");
275         } catch (SQLException e) {
276             System.out.println("Rolled Back The Transaction");
```

```
            try {
                //con.rollback();
                con.rollback(s1); //Add after the rollback demo
            } catch (SQLException e3) {
                e3.printStackTrace();
            }
        } finally {
            try {
                con.setAutoCommit(true);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            closeConnection();
        }

    }

}
-------------------HikariPoolEmployeeServiceImpl.java---------------
package com.ameya.services.impl;

import java.util.List;

import com.ameya.daos.EmployeeDAO;
import com.ameya.domain.Employee;
import com.ameya.services.EmployeeService;

public class HikariPoolEmployeeServiceImpl implements EmployeeService{

    private EmployeeDAO employeeDao;

    public HikariPoolEmployeeServiceImpl(EmployeeDAO employeeDao) {
        this.employeeDao=employeeDao;
    }
    @Override
    public void createEmployee(Employee employee) {
        employeeDao.addEmployee(employee);
    }
    @Override
    public void modifyEmployee(Employee employee) {
        employeeDao.updateEmployee(employee);
    }
    @Override
    public void removeEmployee(int empId) {
        employeeDao.deleteEmployee(empId);
    }
    @Override
    public Employee findEmployeeById(int empId) {
```

```
        return employeeDao.getEmployee(empId);
    }
    @Override
    public List<Employee> findAll() {
        return employeeDao.getAllEmployees();
    }
    @Override
    public void transaction() {
        employeeDao.transactions();
    }
}
--------------------TestConnectionPool.java---------------
package com.ameya.test;

import java.util.List;

import com.ameya.daos.impl.HikariPoolEmployeeDAOImpl;
import com.ameya.domain.Employee;
import com.ameya.services.EmployeeService;
import com.ameya.services.impl.HikariPoolEmployeeServiceImpl;

public class TestConnectionPool {

    public static void main(String[] args) {
        EmployeeService empService=new HikariPoolEmployeeServiceImpl(new
        HikariPoolEmployeeDAOImpl());
        empService.createEmployee(new Employee(1,"Ameya","Joshi",45000,42));
        empService.createEmployee(new Employee(2,"Amol","Patil",47000,41));
        empService.createEmployee(new Employee(3,"Amit","Shah",55000,43));
        empService.createEmployee(new Employee(4,"Sanjay","Kadam",65000,41));
        empService.createEmployee(new Employee(5,"Rahul","Pawar",55000,42));
        System.out.println("+++++++++++++++++++++++++++++++++++++++++++");
        Employee emp=empService.findEmployeeById(3);
        System.out.println(emp);
        System.out.println("+++++++++++++++++++++++++++++++++++++++++++");
        empService.modifyEmployee(new Employee(3,"Pratap","Shah",66000,47));
        System.out.println("+++++++++++++++++++++++++++++++++++++++++++");
        emp=empService.findEmployeeById(3);
        System.out.println(emp);
        System.out.println("+++++++++++++++++++++++++++++++++++++++++++");
        List<Employee> emps=empService.findAll();
        for(Employee e : emps) {
            System.out.println(e);
        }
        System.out.println("+++++++++++++++++++++++++++++++++++++++++++");
        empService.removeEmployee(3);
        emps=empService.findAll();
        for(Employee e : emps) {
```

```
            System.out.println(e);
        }

    }

}
-------------------TestTransactions.java---------------
package com.ameya.test;

import com.ameya.daos.impl.HikariPoolEmployeeDAOImpl;
import com.ameya.services.EmployeeService;
import com.ameya.services.impl.HikariPoolEmployeeServiceImpl;

public class TestTransactions {

    public static void main(String[] args) {
        //EmployeeService service=new EmployeeServiceImpl(new EmployeeDAOImpl());
        EmployeeService service=new HikariPoolEmployeeServiceImpl(new
        HikariPoolEmployeeDAOImpl());
        service.transaction();

    }

}

```