```java
package projects.dao;

import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.Objects;
import java.util.Optional;
import java.util.LinkedList;

import projects.entity.entity.Category;
import projects.entity.entity.Material;
import projects.entity.entity.Project;
import projects.entity.entity.Step;
import projects.exception.DbException;
import provided.util.DaoBase;



public class ProjectDao extends DaoBase {
    private static final String CATEGORY_TABLE = "category";
    private static final String MATERIAL_TABLE = "material";
    private static final String PROJECT_TABLE = "project";
    private static final String PROJECT_CATEGORY_TABLE = "project_category";
    private static final String STEP_TABLE = "step";

    public Project insertProject(Project project) {
        //formatter:off
        String sql = ""
                + "INSERT INTO " + PROJECT_TABLE + " "
                + "(project_name, estimated_hours, actual_hours, difficulty, notes) "
                + "VALUES "
                + "(?, ?, ?, ?, ?)";
        //formatter:on

        try(Connection conn = DbConnection.getConnection()) {
            startTransaction(conn);

            try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, project.getProjectName(), String.class);
                setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
                setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
                setParameter(stmt, 4, project.getDifficulty(), Integer.class);
                setParameter(stmt, 5, project.getNotes(), String.class);

                stmt.executeUpdate();

                Integer projectId = getLastInsertId(conn, PROJECT_TABLE);
                commitTransaction(conn);

                project.setProjectId(projectId);
                return project;
            }
            catch(Exception e) {
                rollbackTransaction(conn);
```

```
58                  throw new DbException(e);
59              }
60          }
61              catch(SQLException e) {
62                  throw new DbException(e);
63              }
64      }
65
66      public List<Project> fetchAllProjects() {
67          String sql = "SELECT * FROM " + PROJECT_TABLE + " ORDER BY project_name";
68          System.out.println(sql);
69
70          try(Connection conn = DbConnection.getConnection()) {
71              startTransaction(conn);
72
73
74              try(PreparedStatement stmt = conn.prepareStatement(sql)) {
75                  try(ResultSet rs = stmt.executeQuery()) {
76                      List<Project> projects = new LinkedList<>();
77
78                      while(rs.next()) {
79                          projects.add(extract(rs, Project.class));
80
81                      }
82                      return projects;
83
84                  }
85              }
86              catch(Exception e) {
87                  rollbackTransaction(conn);
88                  throw new DbException(e);
89              }
90      }
91          catch(SQLException e) {
92              throw new DbException(e);
93          }
94      }
95      public Optional<Project> fetchProjectById(Integer projectId) {
96          String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE project_id = ?";
97
98          try(Connection conn = DbConnection.getConnection()) {
99              startTransaction(conn);
100
101             try {
102                 Project project = null;
103
104                 try(PreparedStatement stmt = conn.prepareStatement(sql)) {
105                     setParameter(stmt, 1, projectId, Integer.class);
106
107                     try(ResultSet rs = stmt.executeQuery()) {
108                         if(rs.next()) {
109                             project = extract(rs, Project.class);
110                         }
111                     }
112                 }
113                 if(Objects.isNull(project)) {
114                     project.getMaterials().addAll(fetchMaterialsForProject(conn, projectId));
```

```
115                     project.getSteps().addAll(fetchStepsForProject(conn, projectId));
116                     project.getCategories().addAll(fetchCategoriesForProject(conn,
    projectId));
117                 }
118                 commitTransaction(conn);
119                 return Optional.ofNullable(project);
120             }
121             catch(Exception e) {
122                 rollbackTransaction(conn);
123                 throw new DbException(e);
124         }
125     }
126         catch(SQLException e) {
127             throw new DbException(e);
128         }
129         }
130 
131         private List<Category> fetchCategoriesForProject(Connection conn, Integer projectId)
    throws SQLException {
132             // @formatter:off
133             String sql = ""
134                     + "SELECT c.* FROM " + CATEGORY_TABLE + " C "
135                     + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (category_id) "
136                     + "WHERE project_id = ?";
137             // @formatter:on
138 
139             try(PreparedStatement stmt = conn.prepareStatement(sql)) {
140                 setParameter(stmt, 1, projectId, Integer.class);
141 
142                 try(ResultSet rs = stmt.executeQuery()) {
143                     List<Category> categories = new LinkedList<>();
144 
145                     while(rs.next()) {
146                         categories.add(extract(rs, Category.class));
147                     }
148                     return categories;
149                 }
150             }
151         }
152 
153         private List<Step> fetchStepsForProject(Connection conn, Integer projectId) throws
    SQLException {
154             String sql = "SELECT * FROM " + STEP_TABLE + " WHERE project_id = ?";
155 
156             try(PreparedStatement stmt = conn.prepareStatement(sql)) {
157                 setParameter(stmt, 1, projectId, Integer.class);
158 
159                 try(ResultSet rs = stmt.executeQuery()) {
160                     List<Step> steps = new LinkedList<>();
161 
162                     while(rs.next()) {
163                         steps.add(extract(rs, Step.class));
164                     }
165                     return steps;
166                 }
167             }
168         }
```

```
169     private List<Material> fetchMaterialsForProject(Connection conn, Integer projectId) throws
    SQLException {
170         String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE project_id = ?";
171 
172         try(PreparedStatement stmt = conn.prepareStatement(sql)) {
173             setParameter(stmt, 1, projectId, Integer.class);
174 
175             try(ResultSet rs = stmt.executeQuery()) {
176                 List<Material> materials = new LinkedList<>();
177 
178                 while(rs.next()) {
179                     materials.add(extract(rs, Material.class));
180                 }
181                 return materials;
182     }
183         }
184 }
185 
186 
187 
188 
189 
190 
191 
192 
193 
194 
195 
196 
197 
198 
199 
200 
201 }
202 
```