ENSF 607 – Principles of Software Development

Fall 2023

Lab Assignment #3

Contributors:

Eric Yoon

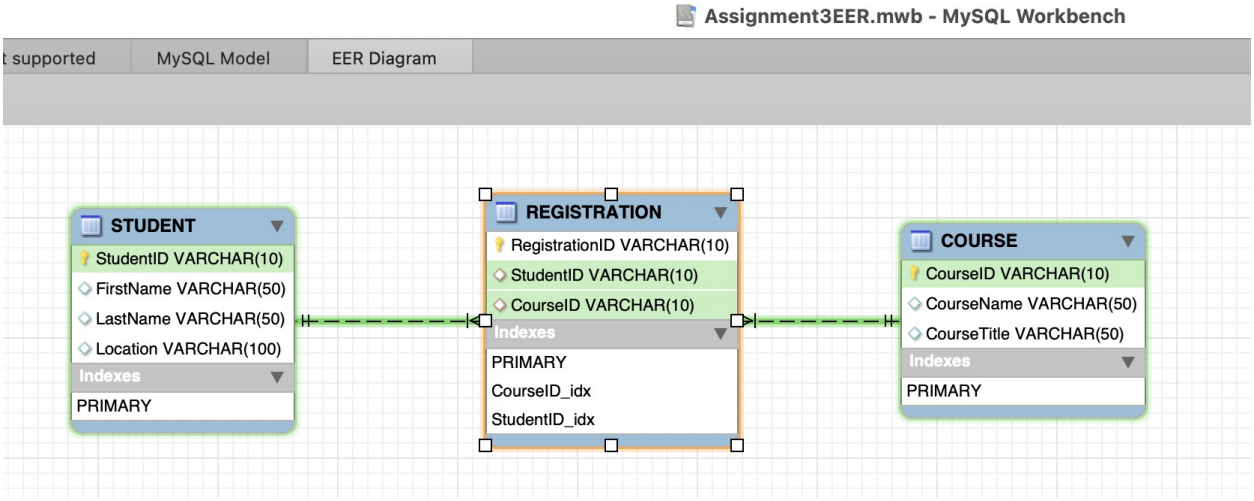
Dhananjay Roy

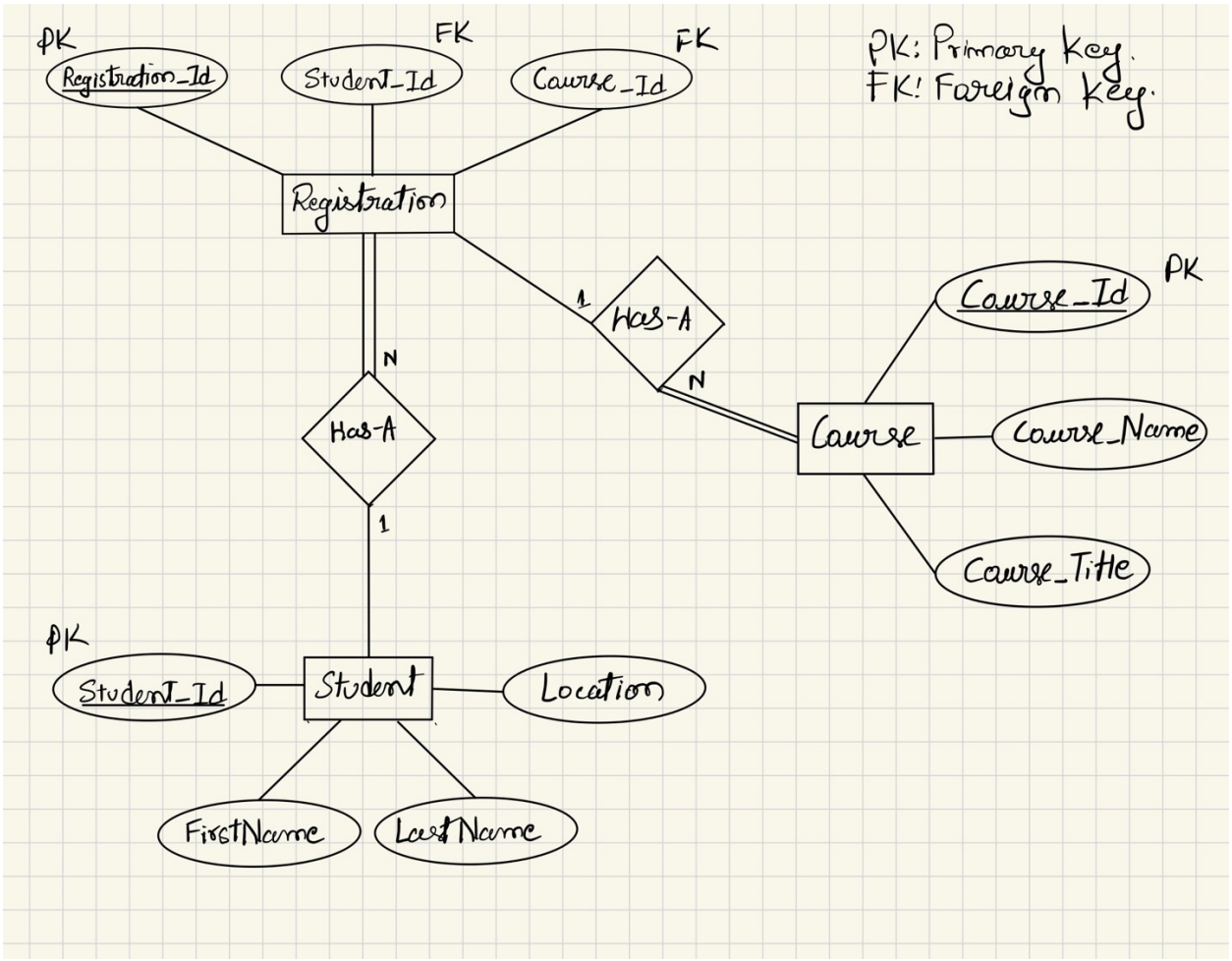
John Chernoff

Sean Temple

Exercise 1:

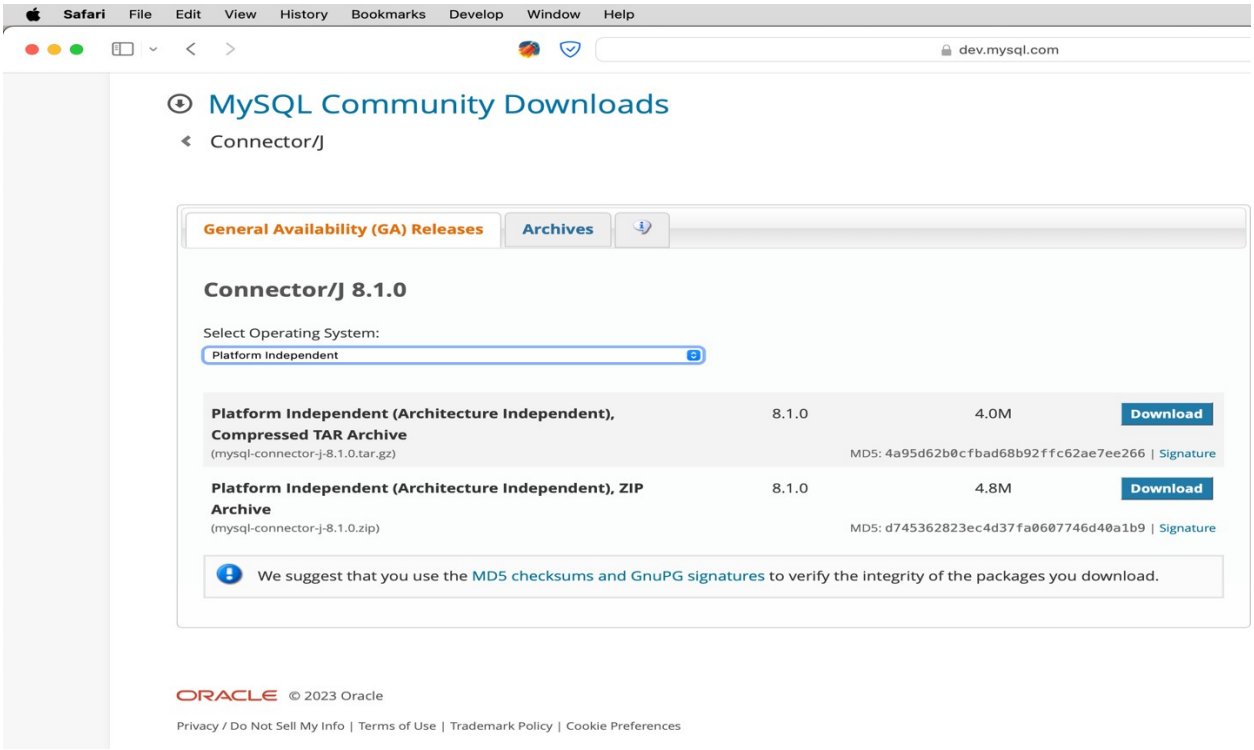
Task 1:



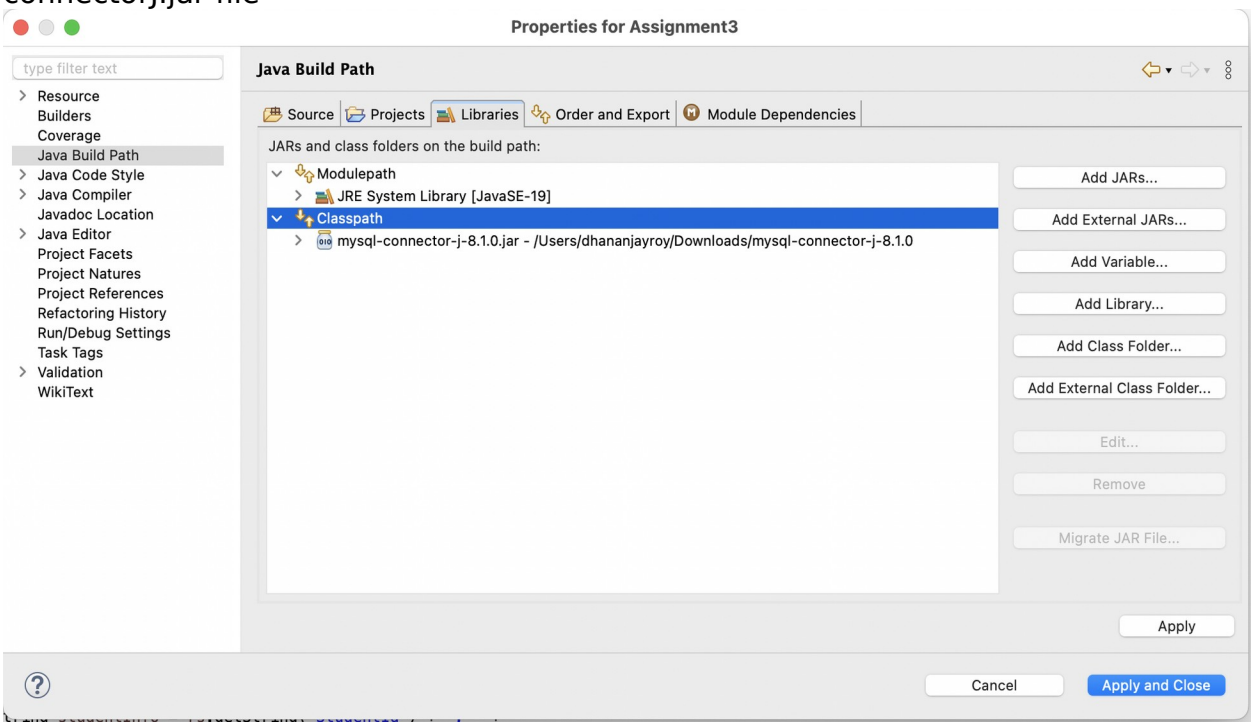


Task 2:

Steps to install JDBC Driver:
Go to https://dev.mysql.com/downloads/connector/j/
Select “Platform independent” for Mac OS platform then download “ZIP” file.



Java Build Path >> Libraries >> Classpath >> Add External Jars >> select the connectorJ.jar file



Task 3:

Populating the Database (set of demo values 10)

```
DROP DATABASE IF EXISTS SCHOOL;
CREATE DATABASE SCHOOL;
USE SCHOOL;

-- Create the 'STUDENT' table
DROP TABLE IF EXISTS STUDENT;
CREATE TABLE STUDENT (
    StudentId VARCHAR(10) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Location VARCHAR(100)
);

-- Inserting data into STUDENT
INSERT INTO STUDENT (StudentId, FirstName, LastName, Location)
VALUES
('S001', 'John', 'Doe', '123 Main St'),
('S002', 'Jane', 'Smith', '456 Elm St'),
('S003', 'Alice', 'Johnson', '789 Pine St'),
('S004', 'Bob', 'Williams', '101 Maple St'),
('S005', 'Charlie', 'Brown', '202 Oak St'),
('S006', 'David', 'Davis', '303 Birch St'),
('S007', 'Eve', 'White', '404 Cedar St'),
('S008', 'Frank', 'Jones', '505 Redwood St'),
('S009', 'Grace', 'Black', '606 Walnut St'),
('S010', 'Harry', 'Green', '707 Spruce St');

-- Create the 'COURSE' table
DROP TABLE IF EXISTS COURSE;
CREATE TABLE COURSE (
    CourseId VARCHAR(10) PRIMARY KEY,
    CourseName VARCHAR(50),
    CourseTitle VARCHAR(50)
);

-- Inserting data into COURSE
INSERT INTO COURSE (CourseId, CourseName, CourseTitle)
VALUES
('C001', 'Math 101', 'Introduction to Mathematics'),
('C002', 'Hist 101', 'World History: Ancient Civilizations'),
('C003', 'Bio 101', 'Basic Biology'),
('C004', 'Chem 101', 'Introduction to Chemistry'),
('C005', 'Phys 101', 'Basic Physics'),
('C006', 'Engl 101', 'Introduction to Literature'),
('C007', 'Comp 101', 'Introduction to Computers'),
('C008', 'Econ 101', 'Microeconomics'),
('C009', 'Psych 101', 'General Psychology'),
('C010', 'Soc 101', 'Introduction to Sociology');

-- Create the 'REGISTRATION' table with foreign key references
DROP TABLE IF EXISTS REGISTRATION;
CREATE TABLE REGISTRATION (
```

```
    RegistrationId VARCHAR(10) PRIMARY KEY,
    CourseId VARCHAR(10),
    StudentId VARCHAR(10),
    FOREIGN KEY (CourseId) REFERENCES COURSE(CourseId),
    FOREIGN KEY (StudentId) REFERENCES STUDENT(StudentId)
);

-- Inserting data into REGISTRATION
INSERT INTO REGISTRATION (RegistrationId, CourseId, StudentId)
VALUES
('R001', 'C001', 'S001'),
('R002', 'C001', 'S002'),
('R003', 'C002', 'S003'),
('R004', 'C002', 'S004'),
('R005', 'C003', 'S005'),
('R006', 'C003', 'S006'),
('R007', 'C004', 'S007'),
('R008', 'C004', 'S008'),
('R009', 'C005', 'S009'),
('R010', 'C005', 'S010');
```

Task 4:

Correct connection string

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/SCHOOL";
private static final String USER = "root";
private static final String PASS = "hariom123";
```

Task 5: Queries via Java Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class DatabaseConnection {
   private static final String DB_URL = "jdbc:mysql://localhost:3306/SCHOOL";
         private static final String USER = "root";
         private static final String PASS = "hariom123";
         private Connection conn = null;

   static {
      try {
         // Load MySQL JDBC driver
         Class.forName("com.mysql.cj.jdbc.Driver");
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }

   public DatabaseConnection() {
      try {
         conn = DriverManager.getConnection(DB_URL, USER, PASS);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   public Connection getConnection() {
      return conn;
   }

   public List<String> getAllStudents() {
      List<String> students = new ArrayList<>();
      try (Statement stmt = conn.createStatement();
           ResultSet rs = stmt.executeQuery("SELECT * FROM STUDENT")) {

         while(rs.next()) {
            String studentInfo = rs.getString("StudentId") + ", " +
                                 rs.getString("FirstName") + ", " +
                                 rs.getString("LastName") + ", " +
                                 rs.getString("Location");
            students.add(studentInfo);
         }

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

   public List<String> getAllCourses() {
      List<String> courses = new ArrayList<>();
      try (Statement stmt = conn.createStatement();
           ResultSet rs = stmt.executeQuery("SELECT * FROM COURSE")) {

         while(rs.next()) {
            String courseInfo = rs.getString("CourseId") + ", " +
                                 rs.getString("CourseName") + ", " +
```

```
                        rs.getString("CourseTitle");
                courses.add(courseInfo);
            }

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

    public List<String> getAllRegistrations() {
        List<String> registrations = new ArrayList<>();
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM REGISTRATION")) {

            while(rs.next()) {
                String registrationInfo = rs.getString("RegistrationId") + ", " +
                                          rs.getString("StudentId") + ", " +
                                          rs.getString("CourseId");
                registrations.add(registrationInfo);
            }

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

    public static void main(String[] args) {
        DatabaseConnection dbConn = new DatabaseConnection();

        System.out.println("All Students:");
        for (String student : dbConn.getAllStudents()) {
            System.out.println(student);
        }

        System.out.println("\nAll Courses:");
        for (String course : dbConn.getAllCourses()) {
            System.out.println(course);
        }

        System.out.println("\nAll Registrations:");
        for (String registration : dbConn.getAllRegistrations()) {
            System.out.println(registration);
        }
    }

}
```

Task 6:

A. **ER Diagram Explanation:**
Entities:

1. **STUDENT**:
    - Represents individual students in the system.
    - **Attributes**:
        - **StudentId**: A unique identifier for each student.

        - **FirstName**: The first name of the student.
        - **LastName**: The last name of the student.
        - **Location**: The location details of the student.
2. **COURSE**:
    - Represents individual courses available for students to register.
    - **Attributes**:
        - **CourseId**: A unique identifier for each course.
        - **CourseName**: The name of the course.
        - **CourseTitle**: The title or brief description of the course.
3. **REGISTRATION**:
    - Represents the registration records indicating which student registered for which course.
    - **Attributes**:
        - **RegistrationId**: A unique identifier for each registration record.
        - **StudentId**: Refers to the student who has registered.
        - **CourseId**: Refers to the course for which the student has registered.

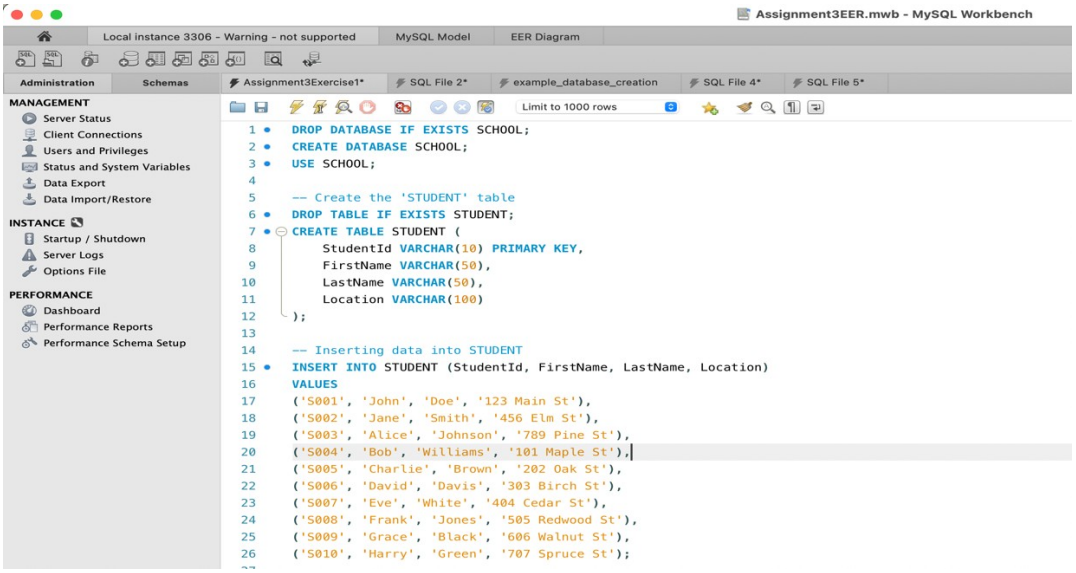
Relationships:

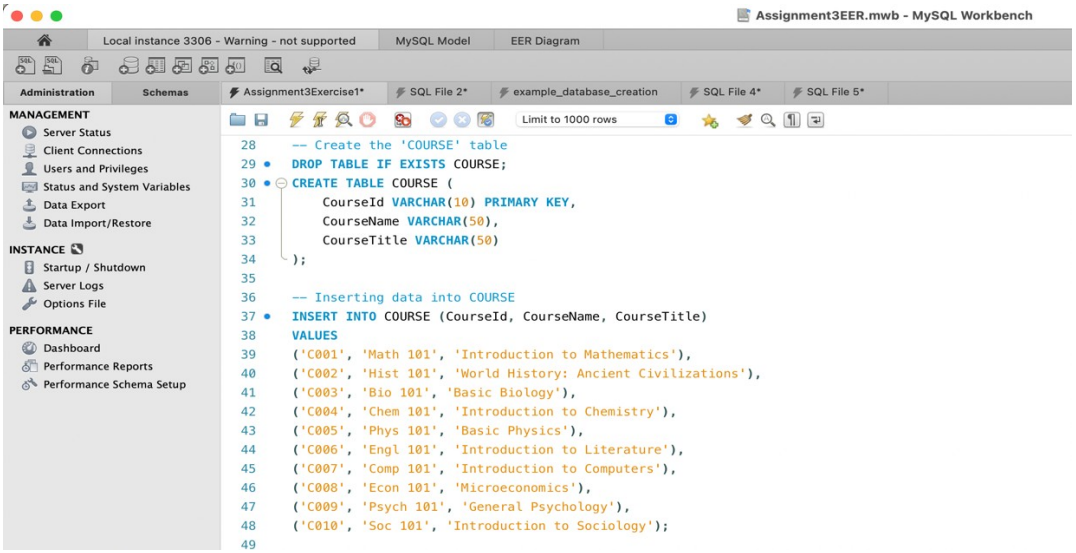
1. **STUDENT and REGISTRATION**:
    - **Type**: One-to-Many
    - **Description**: One student can have many registration records, but each registration record refers to a single student. This relationship can be described with the phrase "A STUDENT **Has** many REGISTRATIONS".
    - **Cardinality**:
        - A student can register for zero to many courses.
        - Each registration is associated with one and only one student.
2. **COURSE and REGISTRATION**:
    - **Type**: One-to-Many
    - **Description**: One course can have many registration records, but each registration record refers to a single course. This can be described with the phrase "A COURSE **Has A** REGISTRATION".
    - **Cardinality**:
        - A course can have zero to many students registered.
        - Each registration is associated with one and only one course.

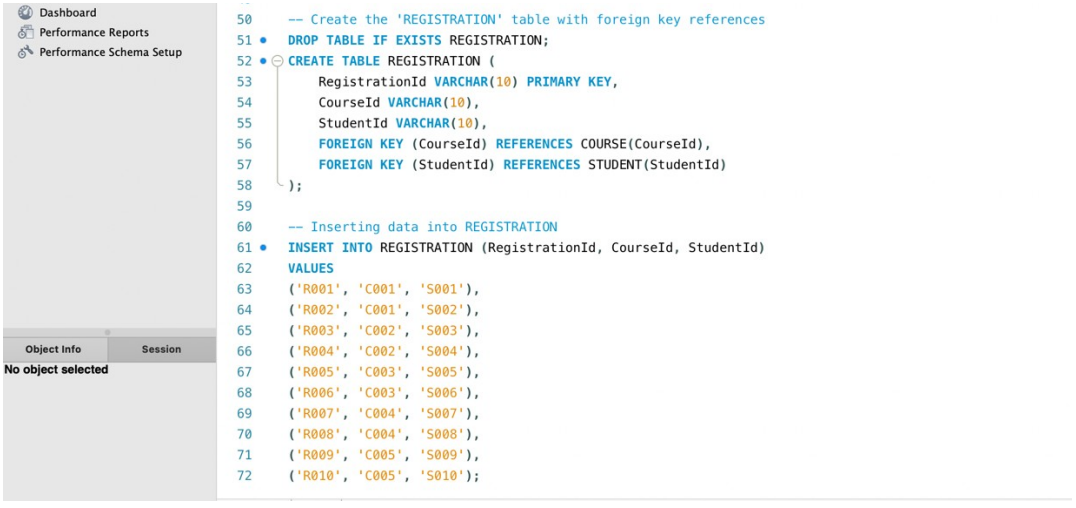
The design and relationships ensure the database supports the scenario of a student registration system. The ER diagram clearly visualizes the relationships and cardinalities between entities. Each entity and its attributes are well-defined, ensuring data integrity and minimizing redundancy.

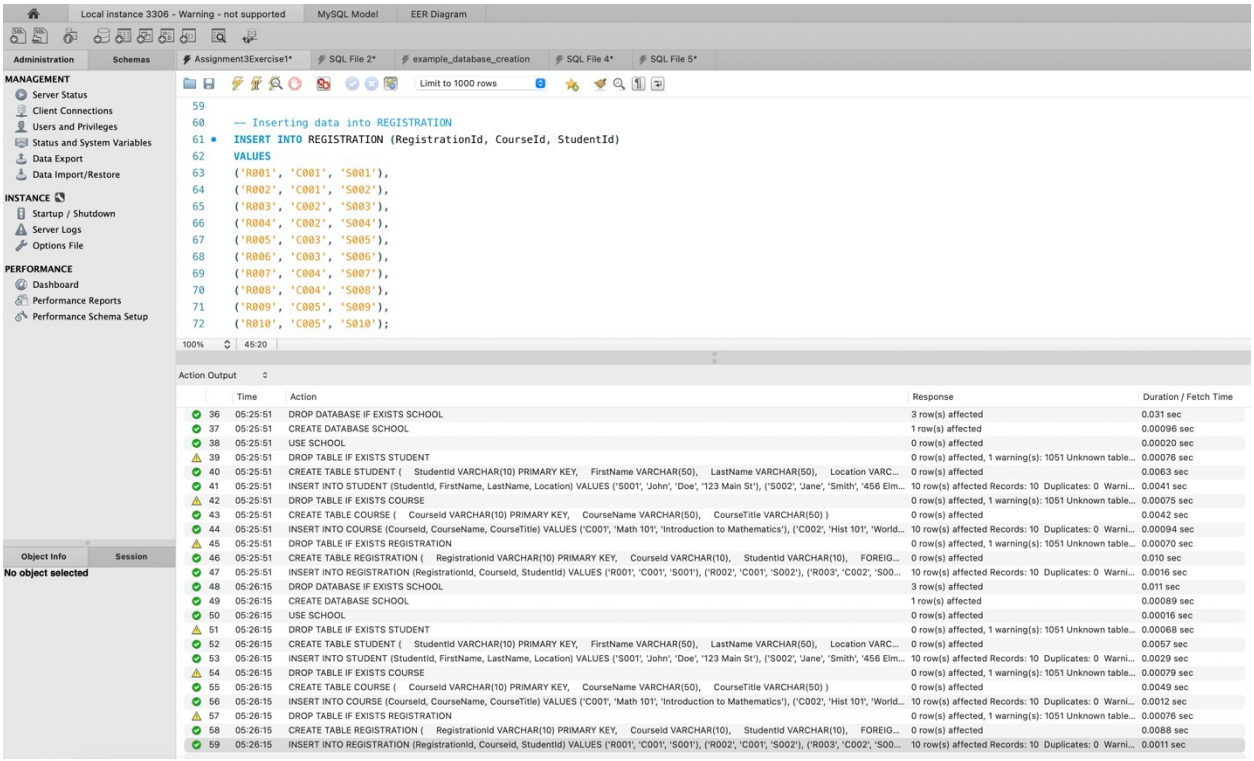
B. Screen print of the database and JDBC install:

Database:


Assignment3EER.mwb - MySQL Workbench
Local instance 3306 - Warning - not supported
MySQL Model
EER Diagram
Administration
Schemas
Assignment3Exercise1*
SQL File 2*
example_database_creation
SQL File 4*
SQL File 5*
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Limit to 1000 rows
1 DROP DATABASE IF EXISTS SCHOOL;
2 CREATE DATABASE SCHOOL;
3 USE SCHOOL;
4
5 -- Create the 'STUDENT' table
6 DROP TABLE IF EXISTS STUDENT;
7 CREATE TABLE STUDENT (
8 StudentId VARCHAR(10) PRIMARY KEY,
9 FirstName VARCHAR(50),
10 LastName VARCHAR(50),
11 Location VARCHAR(100)
12);
13
14 -- Inserting data into STUDENT
15 INSERT INTO STUDENT (StudentId, FirstName, LastName, Location)
16 VALUES
17 ('S001', 'John', 'Doe', '123 Main St'),
18 ('S002', 'Jane', 'Smith', '456 Elm St'),
19 ('S003', 'Alice', 'Johnson', '789 Pine St'),
20 ('S004', 'Bob', 'Williams', '101 Maple St'),
21 ('S005', 'Charlie', 'Brown', '202 Oak St'),
22 ('S006', 'David', 'Davis', '303 Birch St'),
23 ('S007', 'Eve', 'White', '404 Cedar St'),
24 ('S008', 'Frank', 'Jones', '505 Redwood St'),
25 ('S009', 'Grace', 'Black', '606 Walnut St'),
26 ('S010', 'Harry', 'Green', '707 Spruce St');

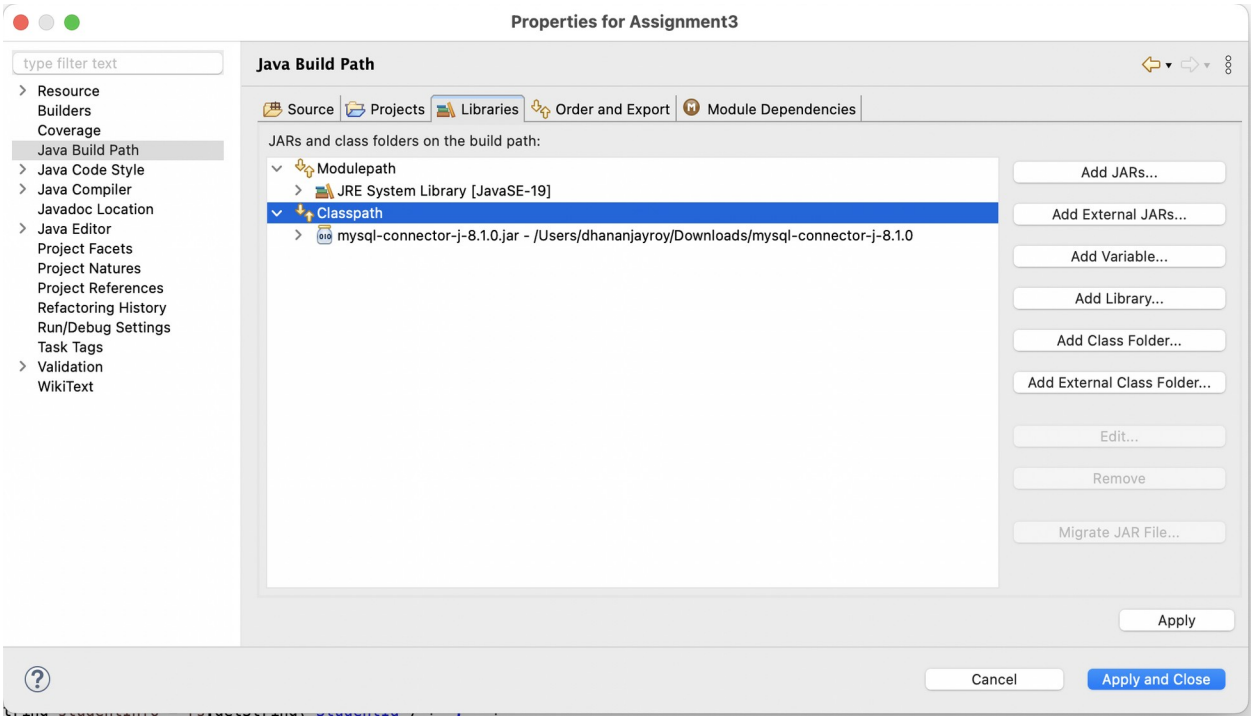

Assignment3EER.mwb - MySQL Workbench
Local instance 3306 - Warning - not supported
MySQL Model
EER Diagram
Administration
Schemas
Assignment3Exercise1*
SQL File 2*
example_database_creation
SQL File 4*
SQL File 5*
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Limit to 1000 rows
28 -- Create the 'COURSE' table
29 DROP TABLE IF EXISTS COURSE;
30 CREATE TABLE COURSE (
31 CourseId VARCHAR(10) PRIMARY KEY,
32 CourseName VARCHAR(50),
33 CourseTitle VARCHAR(50)
34);
35
36 -- Inserting data into COURSE
37 INSERT INTO COURSE (CourseId, CourseName, CourseTitle)
38 VALUES
39 ('C001', 'Math 101', 'Introduction to Mathematics'),
40 ('C002', 'Hist 101', 'World History: Ancient Civilizations'),
41 ('C003', 'Bio 101', 'Basic Biology'),
42 ('C004', 'Chem 101', 'Introduction to Chemistry'),
43 ('C005', 'Phys 101', 'Basic Physics'),
44 ('C006', 'Engl 101', 'Introduction to Literature'),
45 ('C007', 'Comp 101', 'Introduction to Computers'),
46 ('C008', 'Econ 101', 'Microeconomics'),
47 ('C009', 'Psych 101', 'General Psychology'),
48 ('C010', 'Soc 101', 'Introduction to Sociology');
49


Dashboard
Performance Reports
Performance Schema Setup
Object Info
Session
No object selected
50 -- Create the 'REGISTRATION' table with foreign key references
51 DROP TABLE IF EXISTS REGISTRATION;
52 CREATE TABLE REGISTRATION (
53 RegistrationId VARCHAR(10) PRIMARY KEY,
54 CourseId VARCHAR(10),
55 StudentId VARCHAR(10),
56 FOREIGN KEY (CourseId) REFERENCES COURSE(CourseId),
57 FOREIGN KEY (StudentId) REFERENCES STUDENT(StudentId)
58);
59
60 -- Inserting data into REGISTRATION
61 INSERT INTO REGISTRATION (RegistrationId, CourseId, StudentId)
62 VALUES
63 ('R001', 'C001', 'S001'),
64 ('R002', 'C001', 'S002'),
65 ('R003', 'C002', 'S003'),
66 ('R004', 'C002', 'S004'),
67 ('R005', 'C003', 'S005'),
68 ('R006', 'C003', 'S006'),
69 ('R007', 'C004', 'S007'),
70 ('R008', 'C004', 'S008'),
71 ('R009', 'C005', 'S009'),
72 ('R010', 'C005', 'S010');



JDBC install:

- The JDBC driver was added to the library in Eclipse, take a screenshot showing the library jar file in the project structure.
- Java Build Path >> Libraries >> Classpath >> Add External Jars >> select the connectorJ.jar file.



3. Commented Source code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 * A class for managing a database connection and performing various database operations.
 */
public class DatabaseConnection {
    // Define database connection details
    private static final String DB_URL = "jdbc:mysql://localhost:3306/SCHOOL"; // JDBC URL for
MySQL database
    private static final String USER = "root"; // Database username
    private static final String PASS = "hariom123"; // Database password

    private Connection conn = null;

    /**
     * Static block to load the MySQL JDBC driver class.
     */
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver"); // Load the MySQL JDBC driver
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    /**
     * Constructor to establish a database connection when an instance of
DatabaseConnection is created.
     */
    public DatabaseConnection() {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS); // Establish a connection
to the database
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Get the database connection.
     *
     * @return The database connection.
     */
    public Connection getConnection() {
        return conn;
    }

    /**
     * Retrieve a list of all students from the STUDENT table.
     *
```

```
     * @return A list of strings containing student information.
     */
    public List<String> getAllStudents() {
        List<String> students = new ArrayList<>();
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM STUDENT")) {

            while(rs.next()) {
                // Extract student information from the result set and add it to the list
                String studentInfo = rs.getString("StudentId") + ", " +
                                     rs.getString("FirstName") + ", " +
                                     rs.getString("LastName") + ", " +
                                     rs.getString("Location");
                students.add(studentInfo);
            }

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

    /**
     * Retrieve a list of all courses from the COURSE table.
     *
     * @return A list of strings containing course information.
     */
    public List<String> getAllCourses() {
        List<String> courses = new ArrayList<>();
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM COURSE")) {

            while(rs.next()) {
                // Extract course information from the result set and add it to the list
                String courseInfo = rs.getString("CourseId") + ", " +
                                    rs.getString("CourseName") + ", " +
                                    rs.getString("CourseTitle");
                courses.add(courseInfo);
            }

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

    /**
     * Retrieve a list of all registrations from the REGISTRATION table.
     *
     * @return A list of strings containing registration information.
     */
    public List<String> getAllRegistrations() {
        List<String> registrations = new ArrayList<>();
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM REGISTRATION")) {

            while(rs.next()) {
                // Extract registration information from the result set and add it to the list
```

```
            String registrationInfo = rs.getString("RegistrationId") + ", " +
                                      rs.getString("StudentId") + ", " +
                                      rs.getString("CourseId");
            registrations.add(registrationInfo);
        }

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

/**
 * Main method to demonstrate database operations.
 *
 * @param args The command-line arguments (not used in this example).
 */
public static void main(String[] args) {
    DatabaseConnection dbConn = new DatabaseConnection();

    // Retrieve and display all students
    System.out.println("All Students:");
    for (String student : dbConn.getAllStudents()) {
        System.out.println(student);
    }

    // Retrieve and display all courses
    System.out.println("\nAll Courses:");
    for (String course : dbConn.getAllCourses()) {
        System.out.println(course);
    }

    // Retrieve and display all registrations
    System.out.println("\nAll Registrations:");
    for (String registration : dbConn.getAllRegistrations()) {
        System.out.println(registration);
    }

    // Assuming you add a close method to close the database connection
    // dbConn.close();
}
}
```

4. Output:


Fall Semester - Assignment3/src/DatabaseConnection.java - Eclipse IDE
Package Explorer
Assignment3
src
(default package)
DatabaseConnection.java
JRE System Library [JavaSE-19]
Referenced Libraries
ER
Exercise1SQL
DatabaseConnection.java
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.ResultSet;
4 import java.sql.Statement;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 /**
9 * A class for managing a database connection and performing various database operations.
10 */
11 public class DatabaseConnection {
12 // Define database connection details
13 private static final String DB_URL = "jdbc:mysql://localhost:3306/SCHOOL"; // JDBC URL for MySQL database
14 private static final String USER = "root"; // Database username
15 private static final String PASS = "hariom123"; // Database password
16
17 private Connection conn = null;
18
19 /**
20 * Static block to load the MySQL JDBC driver class.
21 */
22 static {
23 try {
24 Class.forName("com.mysql.cj.jdbc.Driver"); // Load the MySQL JDBC driver
Javadoc
Console
Problems
Declaration
PlantUML
PlantUML Project Class Diagram
PlantUML Source
PlantUML Svg
<terminated> DatabaseConnection [Java Application] /Library/Java/JavaVirtualMachines/jdk-19.0.2.jdk/Contents/Home/bin/java (Oct 16, 2023, 3:37:10 a.m. – 3:37:13 a.m.) [pid: 84689]
All Students:
S001, John, Doe, 123 Main St
S002, Jane, Smith, 456 Elm St
S003, Alice, Johnson, 789 Pine St
S004, Bob, Williams, 101 Maple St
S005, Charlie, Brown, 202 Oak St
S006, David, Davis, 303 Birch St
S007, Eve, White, 404 Cedar St
S008, Frank, Jones, 505 Redwood St
S009, Grace, Black, 606 Walnut St
S010, Harry, Green, 707 Spruce St
All Courses:
C001, Math 101, Introduction to Mathematics
C002, Hist 101, World History: Ancient Civilizations
C003, Bio 101, Basic Biology
C004, Chem 101, Introduction to Chemistry
C005, Phys 101, Basic Physics
C006, Engl 101, Introduction to Literature
C007, Comp 101, Introduction to Computers
C008, Econ 101, Microeconomics
C009, Psych 101, General Psychology
C010, Soc 101, Introduction to Sociology
All Registrations:
R001, S001, C001
R002, S002, C001
R003, S003, C002
R004, S004, C002
R005, S005, C003
R006, S006, C003
R007, S007, C004
R008, S008, C004
R009, S009, C005
R010, S010, C005

Exercise 2:

1. The SQL scripts for your tables.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class ServiceTicketDatabase {

    public static void main(String[] args) {
        try {
            // Database connection properties
            String jdbcUrl = "jdbc:mysql://localhost:3306/Assignment3";
            String username = "root";
            String password = "password";

            // Establish a database connection
            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

            // Create a Statement object
            Statement statement = connection.createStatement();

            // Create the EventActivity table
            String createEventActivityTable = "CREATE TABLE EventActivity ("
                    + "ID INT AUTO_INCREMENT PRIMARY KEY, "
                    + "Activityname VARCHAR(20)"
                    + ")";
            statement.execute(createEventActivityTable);

            // Create the EventOrigin table
            String createEventOriginTable = "CREATE TABLE EventOrigin ("
                    + "ID INT AUTO_INCREMENT PRIMARY KEY, "
                    + "Activityname VARCHAR(20)"
                    + ")";
            statement.execute(createEventOriginTable);

           // Create the EventClass table
            String createEventClassTable = "CREATE TABLE EventClass ("
                + "ID INT AUTO_INCREMENT PRIMARY KEY, "
                + "Class VARCHAR(20)"
                + ")";
            statement.execute(createEventClassTable);

            // Create the EventLog table
            String createEventLogTable = "CREATE TABLE EventLog ("
                + "ID INT AUTO_INCREMENT PRIMARY KEY, "
                + "Caseid VARCHAR(20) UNIQUE, "
                + "Activity VARCHAR(20), "
                + "Urgency VARCHAR(1), "
                + "Impact VARCHAR(1), "
                + "Priority VARCHAR(1), "
                + "StartDate DATE, "
                + "EndDate DATE, "
                + "TicketStatus VARCHAR(20), "
                + "UpdateDateTime DATETIME, "
                + "Duration INT, "
                + "Origin VARCHAR(20), "
                + "Class VARCHAR(20)"
                + ")";
            statement.execute(createEventLogTable);

            // Close the statement and connection
            statement.close();
            connection.close();

            System.out.println("Service ticket database setup completed.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

SQL Visualization:

ID	Caseid	Activity	Urgency	Impact	Priority	StartDate	EndDate	TicketStatus	UpdateDateTime	Duration	Origin	Class
582	CS_582	Password Reset	1	2	2	2023-02-06	2023-03-30	Deployed	2023-10-18 21:42:42	51	Joe S.	Incident
583	CS_583	Design	1	2	2	2023-04-01	2023-04-07	Open	2023-10-18 21:42:42	6	George E.	SRforServiceRequest
584	CS_584	Password Reset	3	3	9	2023-01-12	2023-04-06	Deployed	2023-10-18 21:42:42	83	George E.	SRforServiceRequest
585	CS_585	Construction	3	1	3	2023-04-29	2023-06-04	On Hold	2023-10-18 21:42:42	35	George E.	Problem
586	CS_586	Test	3	2	6	2023-05-15	2023-05-27	Deployed	2023-10-18 21:42:42	12	Achmed M.	Problem
587	CS_587	Password Reset	1	3	3	2023-01-28	2023-04-19	Deployed Failed	2023-10-18 21:42:42	80	Bill B.	Problem
588	CS_588	Test	2	2	4	2023-02-28	2023-03-23	On Hold	2023-10-18 21:42:42	23	Joe S.	SRforServiceRequest
589	CS_589	Construction	1	3	3	2023-02-03	2023-05-25	On Hold	2023-10-18 21:42:42	110	Joe S.	Change
590	CS_590	Design	2	1	2	2023-02-07	2023-02-18	In Process	2023-10-18 21:42:42	11	Rona E.	Incident
591	CS_591	Construction	3	3	9	2023-01-13	2023-05-16	In Process	2023-10-18 21:42:42	123	Rona E.	Problem
592	CS_592	Construction	2	2	4	2023-03-25	2023-05-07	Deployed Failed	2023-10-18 21:42:42	42	George E.	Problem
593	CS_593	Construction	3	2	6	2023-02-19	2023-04-05	Open	2023-10-18 21:42:42	45	George E.	Change
594	CS_594	Test	3	1	3	2023-06-28	2023-06-29	Open	2023-10-18 21:42:42	1	Bill B.	Change
595	CS_595	Construction	3	2	6	2023-05-31	2023-06-16	Deployed	2023-10-18 21:42:42	16	Rona E.	Change
596	CS_596	Test	1	1	1	2023-03-25	2023-04-21	Deployed Failed	2023-10-18 21:42:42	27	Achmed M.	Change
597	CS_597	Password Reset	1	2	2	2023-05-08	2023-06-18	On Hold	2023-10-18 21:42:42	41	George E.	Change
598	CS_598	Design	2	3	6	2023-03-27	2023-05-01	In Process	2023-10-18 21:42:42	34	Achmed M.	Incident
599	CS_599	Test	1	3	3	2023-03-02	2023-04-02	Deployed Failed	2023-10-18 21:42:42	31	Bill B.	Incident
600	CS_600	Test	2	2	4	2023-02-28	2023-04-27	Deployed Failed	2023-10-18 21:42:42	57	George E.	Problem
601	CS_601	Design	1	1	1	2023-04-16	2023-05-31	In Process	2023-10-18 21:42:42	45	Achmed M.	SRforServiceRequest
602	CS_602	Test	2	3	6	2023-06-17	2023-06-27	Deployed	2023-10-18 21:42:42	10	Rona E.	Change
603	CS_603	Design	2	3	6	2023-05-21	2023-06-21	Deployed Failed	2023-10-18 21:42:42	30	Joe S.	SRforServiceRequest
604	CS_604	Test	1	1	1	2023-03-06	2023-04-06	Deployed Failed	2023-10-18 21:42:42	30	Achmed M.	Problem
605	CS_605	Password Reset	2	2	4	2023-05-07	2023-05-16	Open	2023-10-18 21:42:42	8	Rona E.	Incident
606	CS_606	Design	2	3	6	2023-03-15	2023-05-25	Open	2023-10-18 21:42:42	70	Achmed M.	Incident
607	CS_607	Password Reset	2	3	6	2023-05-04	2023-05-29	In Process	2023-10-18 21:42:42	24	George E.	Incident
608	CS_608	Password Reset	3	2	6	2023-04-22	2023-05-22	On Hold	2023-10-18 21:42:42	29	Joe S.	Incident
609	CS_609	Construction	3	1	3	2023-03-22	2023-06-28	On Hold	2023-10-18 21:42:42	98	Achmed M.	Problem

Dashboard visualization found in submitted powerpoint