**Q1** Here is the SQL script for creating a database to represent the students, departments, and classes as described:

CREATE DATABASE student\_records;

USE student\_records;

CREATE TABLE Departments (

name VARCHAR(255) NOT NULL,

campus VARCHAR(255) NOT NULL,

PRIMARY KEY (name)

);

CREATE TABLE Students (

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

id INT(9) NOT NULL,

credits INT,

PRIMARY KEY (id)

);

CREATE TABLE Classes (

name VARCHAR(255) NOT NULL,

credits INT,

PRIMARY KEY (name)

);

CREATE TABLE Majors (

sid INT(9),

dname VARCHAR(255),

FOREIGN KEY (sid) REFERENCES Students(id),

FOREIGN KEY (dname) REFERENCES Departments(name),

PRIMARY KEY (sid, dname)

);

CREATE TABLE Minors (

sid INT(9),

dname VARCHAR(255),

FOREIGN KEY (sid) REFERENCES Students(id),

FOREIGN KEY (dname) REFERENCES Departments(name),

PRIMARY KEY (sid, dname)

);

CREATE TABLE IsTaking (

sid INT(9),

name VARCHAR(255),

FOREIGN KEY (sid) REFERENCES Students(id),

FOREIGN KEY (name) REFERENCES Classes(name),

PRIMARY KEY (sid, name)

);

CREATE TABLE HasTaken (

sid INT(9),

name VARCHAR(255),

grade CHAR(1),

FOREIGN KEY (sid) REFERENCES Students(id),

FOREIGN KEY (name) REFERENCES Classes(name),

PRIMARY KEY (sid, name)

);

This creates a database called "student\_records" with tables for departments, students, classes, majors, minors, courses a student is taking, and courses a student has taken. The tables are connected through foreign keys to ensure data integrity.

\\

**Q3** .  
  
    

                     

Here's an outline of the program:

Import necessary packages, including java.sql.\* for JDBC.

Create a database connection using JDBC.

Define methods for each of the required queries and operations, including:

a. A method to search students by name. This method takes a search string as input, and queries the database to find any students whose first or last name contains the search string (case insensitive). It returns a list of matching students, including their ID, name, major, minor, credits, and GPA.

b. A method to search students by year. This method takes a year (Fr, So, Ju, Sr) as input, and queries the database to find any students whose number of credits completed falls within the range for the given year. It returns a list of matching students, including their ID, name, major, minor, credits, and GPA.

c. A method to search for students with a GPA equal to or above a given threshold. This method takes a threshold as input, and calculates the GPA for each student based on their grades in the classes they've taken. It then queries the database to find any students whose GPA is greater than or equal to the threshold. It returns a list of matching students, including their ID, name, major, minor, credits, and GPA.

d. A method to search for students with a GPA equal to or below a given threshold. This method is similar to the previous method, but queries the database to find any students whose GPA is less than or equal to the threshold.

e. A method to report the number of students and the average GPA for a given department. This method takes a department name as input, and queries the database to find the number of students in the department and their average GPA.

f. A method to report the number of students currently taking a given class, and the number of students who've gotten each letter grade for the class. This method takes a class name as input, and queries the database to find the number of students currently enrolled in the class, as well as the number of students who've received each letter grade in the past.

import java.sql.\*;

import java.util.\*;

public class UniDB {

private static final String DB\_URL = "jdbc:sqlite:unidb.sqlite";

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Connection connection = null;

try {

// Connect to database

connection = DriverManager.getConnection(DB\_URL);

System.out.println("Connected to database successfully.");

// Main program loop

while (true) {

System.out.println("\nQueries available:");

System.out.println("1. Search students by name.");

System.out.println("2. Search students by year.");

System.out.println("3. Search for students with a GPA >= threshold.");

System.out.println("4. Search for students with a GPA <= threshold.");

System.out.println("5. Get department statistics.");

System.out.println("6. Get class statistics.");

System.out.println("7. Execute an arbitrary SQL query.");

System.out.println("8. Exit the application.");

System.out.println("Which query would you like to run (1-8)?");

int queryNum = scanner.nextInt();

scanner.nextLine();

if (queryNum == 1) {

System.out.println("Please enter the name:");

String name = scanner.nextLine();

searchStudentsByName(connection, name);

} else if (queryNum == 2) {

System.out.println("Please enter the year (Fr, So, Ju, Sr):");

String year = scanner.nextLine();

searchStudentsByYear(connection, year);

} else if (queryNum == 3) {

System.out.println("Please enter the threshold GPA:");

double threshold = scanner.nextDouble();

scanner.nextLine();

searchStudentsByGpa(connection, threshold, ">=");

} else if (queryNum == 4) {

System.out.println("Please enter the threshold GPA:");

double threshold = scanner.nextDouble();

scanner.nextLine();

searchStudentsByGpa(connection, threshold, "<=");

} else if (queryNum == 5) {

System.out.println("Please enter the department:");

String department = scanner.nextLine();

getDepartmentStatistics(connection, department);

} else if (queryNum == 6) {

System.out.println("Please enter the class name:");

String className = scanner.nextLine();

getClassStatistics(connection, className);

} else if (queryNum == 7) {

System.out.println("Please enter the query:");

String query = scanner.nextLine();

executeQuery(connection, query);

} else if (queryNum == 8) {

break;

} else {

System.out.println("Invalid query number.");

}

}

} catch (SQLException e) {

System.err.println("Error connecting to database: " + e.getMessage());

} finally {

// Close database connection

try {

if (connection != null) {

connection.close();

System.out.println("Database connection closed.");

}

} catch (SQLException e) {

System.err.println("Error closing database connection: " + e.getMessage());

}

}

}

/\*\*

\* Search for students by name.

\*

\* @param connection the database connection

\* @param name the name to search for

\*/

private static void searchStudentsByName(Connection connection, String name) throws SQLException {

String query = "SELECT \* FROM students WHERE LOWER(first\_name) LIKE ? OR LOWER(last\_name) LIKE ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, "%" + name.toLowerCase() + "%");

statement.setString(2, "%" + name.toLowerCase() + "%");

ResultSet resultSet = statement.executeQuery();

}