LAB SESSION 2
25/07/2024

File Processing

1. Develop an implementation package using 'C' program to process a FILE containing student details for the given queries.

A student record has the following format:

Std_rollno, Std_name, Dept, C1, C1_c, C1_g, C2, C2_c, C2_g, C3, C3_c, C3_g

Note: C1 refers to Course1, C1_c refers to credit of the course, C1_g refers to the grade in that course and so on.

Every student should have a unique rollno.

A student should have at least 3 courses and maximum four.

A grade point is in integer: S - 10; A - 9; B - 8; C - 7; D - 6; E - 5; F – 0.

Create a file and develop a menu driven system for the following queries.

a. Insert at least 5 student records.

b. Create a column 'GPA' for all the students.

c. For a student with four courses, delete(deregister) a course name.

d. For the same student you deleted in 'c', insert a new course name.

e. Update the name of a course for two different students.

f. Calculate GPA of all students using the GPA formula. Refer the following:

https://www.nitt.edu/home/academics/rules/BTech_Regulations_2019.pdf

g. Upgrade the grade point of a student who has secured '7' in a course.

h. Calculate the updated GPA of the student in 'g'.

i. Generate a Grade report of a student given the roll no. or name.

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_COURSES 4
#define MAX_NAME_LENGTH 50
#define FILE_NAME "students.txt"

typedef struct {
    int rollno;
    char name[MAX_NAME_LENGTH];
    char dept[MAX_NAME_LENGTH];
    char courses[MAX_COURSES][MAX_NAME_LENGTH];
    int credits[MAX_COURSES];
    int grades[MAX_COURSES];
    float GPA;
} Student;
void insertStudentRecords();
void createGPAColumn();
void deleteCourse();
void insertCourse();
void updateCourseName();
void calculateGPA();
void upgradeGradePoint();
void generateGradeReport();
void displayMenu();
void writeToFile(Student *students, int count);
void readFromFile(Student *students, int *count);
/**
 * The main function of the program, which displays a menu and performs
different actions based on the user's choice.
 *
 * @return 0 indicating successful execution
 */
int main() {
    int choice;

    do {
        displayMenu();
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                insertStudentRecords();
                break;
            case 2:
                createGPAColumn();
                break;
            case 3:
                deleteCourse();
                break;
            case 4:
                insertCourse();
                break;
            case 5:
                updateCourseName();
                break;
            case 6:
                calculateGPA();
                break;
            case 7:
                upgradeGradePoint();
                break;
            case 8:
                generateGradeReport();
                break;
            case 9:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while(choice != 9);

    return 0;
}
/**
 * Displays a menu to the user with options for inserting student records,
creating a GPA column, deleting a course,
 * inserting a course, updating the course name, calculating the GPA,
upgrading the grade point, generating a grade report,
 * and exiting the program.
 *
 * @return void
 */
void displayMenu() {
    printf("\nMenu:\n");
    printf("1. Insert student records\n");
    printf("2. Create GPA column\n");
    printf("3. Delete a course\n");
    printf("4. Insert a course\n");
    printf("5. Update course name\n");
    printf("6. Calculate GPA\n");
    printf("7. Upgrade grade point\n");
    printf("8. Generate grade report\n");
    printf("9. Exit\n");
}
/**
 * Writes the student records to a file.
 *
 * @param students A pointer to an array of Student structures.
 * @param count The number of student records to write.
 *
 * @return void
 *
 * @throws None
 */
void writeToFile(Student *students, int count) {
    FILE *file = fopen(FILE_NAME, "w");
    if (file == NULL) {
        printf("Unable to open file.\n");
        return;
    }
    fprintf(file, "%d\n", count);
    for(int i = 0; i < count; i++) {
        fprintf(file, "%d %s %s %f ", students[i].rollno, students[i].name,
students[i].dept, students[i].GPA);
        for(int j = 0; j < MAX_COURSES; j++) {
            fprintf(file, "%s %d %d ", students[i].courses[j],
students[i].credits[j], students[i].grades[j]);
        }
        fprintf(file, "\n");
    }
    fclose(file);
}
/**
 * Reads student records from a file.
 *
 * @param students pointer to an array of Student structures
 * @param count pointer to the number of student records to read
 *
 * @return void
 *
 * @throws None
 */
void readFromFile(Student *students, int *count) {
    FILE *file = fopen(FILE_NAME, "r");
    if (file == NULL) {
        printf("Unable to open file.\n");
        *count = 0;
        return;
    }
    fscanf(file, "%d", count);
    for(int i = 0; i < *count; i++) {
        fscanf(file, "%d %s %s %f", &students[i].rollno, students[i].name,
students[i].dept, &students[i].GPA);
        for(int j = 0; j < MAX_COURSES; j++) {
            fscanf(file, "%s %d %d", students[i].courses[j],
&students[i].credits[j], &students[i].grades[j]);
        }
    }
    fclose(file);
}
/**
 * Inserts student records into the database.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void insertStudentRecords() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    int n, i, j;
    printf("Enter the number of students to insert: ");
    scanf("%d", &n);

    for(i = count; i < count + n; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("Roll Number: ");
        scanf("%d", &students[i].rollno);
        printf("Name: ");
        scanf("%s", students[i].name);
        printf("Department: ");
        scanf("%s", students[i].dept);
        for(j = 0; j < MAX_COURSES; j++) {
            printf("Course %d name (enter 'NA' if no more courses): ", j + 1);
            scanf("%s", students[i].courses[j]);
            if(strcmp(students[i].courses[j], "NA") == 0) break;
            printf("Course %d credits: ", j + 1);
            scanf("%d", &students[i].credits[j]);
            printf("Course %d grade: ", j + 1);
            scanf("%d", &students[i].grades[j]);
        }
    }
    count += n;
    writeToFile(students, count);
}
/**
 * Creates a GPA column for all students by setting the GPA field of each
student to 0.0.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void createGPAColumn() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    for(int i = 0; i < count; i++) {
        students[i].GPA = 0.0;
    }
    writeToFile(students, count);
    printf("GPA column created for all students.\n");
}
/**
 * Deletes a course for a student by setting the course name, credits, and
grade to "NA", 0, and 0 respectively.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void deleteCourse() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    int rollno, found = 0;
    printf("Enter roll number of student to delete a course: ");
    scanf("%d", &rollno);

    for(int i = 0; i < count; i++) {
        if(students[i].rollno == rollno) {
            found = 1;
            int counter = 0;
            for(int j = 0; j < MAX_COURSES; j++) {
                if(strcmp(students[i].courses[j], "NA") != 0) {
                    counter++;
                }
            }
            if(counter == MAX_COURSES) {
                for(int j = 0; j < MAX_COURSES; j++) {
                    if(strcmp(students[i].courses[j], "NA") != 0) {
                        printf("Deleting course %s for student %d.\n",
students[i].courses[j], rollno);
                        strcpy(students[i].courses[j], "NA");
                        students[i].credits[j] = 0;
                        students[i].grades[j] = 0;
                        break;
                    }
                }
            }
        }
    }

    if(!found) {
        printf("Student with roll number %d not found.\n", rollno);
    } else {
        writeToFile(students, count);
    }
}
/**
 * Inserts a course for a student into the database.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void insertCourse() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    int rollno, found = 0;
    printf("Enter roll number of student to insert a course: ");
    scanf("%d", &rollno);

    for(int i = 0; i < count; i++) {
        if(students[i].rollno == rollno) {
            found = 1;
            for(int j = 0; j < MAX_COURSES; j++) {
                if(strcmp(students[i].courses[j], "NA") == 0) {
                    printf("Enter new course name: ");
                    scanf("%s", students[i].courses[j]);
                    printf("Enter new course credits: ");
                    scanf("%d", &students[i].credits[j]);
                    printf("Enter new course grade: ");
                    scanf("%d", &students[i].grades[j]);
                    break;
                }
            }
        }
    }

    if(!found) {
        printf("Student with roll number %d not found.\n", rollno);
    } else {
        writeToFile(students, count);
    }
}
/**
 * Updates the course name for a student in the database.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void updateCourseName() {
    // Declare an array of Student structures to hold student records
    Student students[100];
    // Initialize the count of student records to 0
    int count = 0;
    // Read student records from a file into the students array
    readFromFile(students, &count);

    // Declare variables to hold the old and new course names
    char oldCourse[MAX_NAME_LENGTH], newCourse[MAX_NAME_LENGTH];
    // Prompt the user to enter the old course name to update
    printf("Enter the old course name to update: ");
    // Read the old course name from the user input
    scanf("%s", oldCourse);
    // Prompt the user to enter the new course name
    printf("Enter the new course name: ");
    // Read the new course name from the user input
    scanf("%s", newCourse);
    int counter = 0;
    // Loop through each student record
    for(int i = 0; i < count; i++) {
        // Loop through each course for the current student record
        for(int j = 0; j < MAX_COURSES; j++) {
            // Check if the current course name matches the old course name
            if(strcmp(students[i].courses[j], oldCourse) == 0) {
                // Update the course name to the new course name
                strcpy(students[i].courses[j], newCourse);
                // Print a message indicating the course name was updated
                printf("Updated course name from %s to %s for student %d.\n",
oldCourse, newCourse, students[i].rollno);
                counter++;
            }
        }
        if(counter == 2){
            break;
        }
```

```c
    }
    // Write the updated student records back to the file
    writeToFile(students, count);
}
/**
 * Calculates the GPA for all students in the database.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void calculateGPA() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    for(int i = 0; i < count; i++) {
        int totalCredits = 0;
        float totalPoints = 0.0;
        for(int j = 0; j < MAX_COURSES; j++) {
            if(strcmp(students[i].courses[j], "NA") != 0) {
                totalCredits += students[i].credits[j];
                totalPoints += students[i].grades[j] * students[i].credits[j];
            }
        }
        if(totalCredits != 0) {
            students[i].GPA = totalPoints / totalCredits;
        } else {
            students[i].GPA = 0.0;
        }
    }
    writeToFile(students, count);
    printf("GPA calculated for all students.\n");
}
/**
 * Upgrades the grade points for a student in the database.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void upgradeGradePoint() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    int found = 0;
    for(int i = 0; i < count; i++) {
        for(int j = 0; j < MAX_COURSES; j++) {
            if(strcmp(students[i].courses[j], "NA") != 0) {
                if(students[i].grades[j] == 7) {
                    found = 1;
                    students[i].grades[j]++;
                    printf("Upgraded grade for course %s to %d for student
%d.\n", students[i].courses[j], students[i].grades[j], students[i].rollno);
                }
            }
        }
    }

    if(!found) {
        printf("Student with grade 7 not found.\n");
    } else {
        writeToFile(students, count);
    }
}
/**
 * Generates a grade report for a student based on their roll number.
 *
 * @param None
 *
 * @return None
 *
 * @throws None
 */
void generateGradeReport() {
    Student students[100];
    int count = 0;
    readFromFile(students, &count);

    int rollno, found = 0;
    printf("Enter roll number of student to generate grade report: ");
    scanf("%d", &rollno);

    for(int i = 0; i < count; i++) {
        if(students[i].rollno == rollno) {
            found = 1;
            printf("Grade Report for %s (Roll No: %d):\n", students[i].name,
students[i].rollno);
            for(int j = 0; j < MAX_COURSES; j++) {
                if(strcmp(students[i].courses[j], "NA") != 0) {
                    printf("Course %d: %s, Credits: %d, Grade: %d\n", j + 1,
students[i].courses[j], students[i].credits[j], students[i].grades[j]);
                }
            }
            break;
        }
    }

    if(!found) {
        printf("Student with roll number %d not found.\n", rollno);
    }
}
```

Output:

```
Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 1

Enter the number of students to insert: 5

Enter details for student 3:

Roll Number: 1

Name: pranav

Department: cse

Course 1 name (enter 'NA' if no more courses): os

Course 1 credits: 3

Course 1 grade: 10

Course 2 name (enter 'NA' if no more courses): dsa
Course 2 credits: 3

Course 2 grade: 9

Course 3 name (enter 'NA' if no more courses): eco

Course 3 credits: 2

Course 3 grade: 8

Course 4 name (enter 'NA' if no more courses): NA

Enter details for student 4:

Roll Number: 2

Name: rahul

Department: ece

Course 1 name (enter 'NA' if no more courses): dsd

Course 1 credits: 3

Course 1 grade: 9

Course 2 name (enter 'NA' if no more courses): ep

Course 2 credits: 2

Course 2 grade: 7

Course 3 name (enter 'NA' if no more courses): eg

Course 3 credits: 2

Course 3 grade: 8

Course 4 name (enter 'NA' if no more courses): eng

Course 4 credits: 4

Course 4 grade: 10

Enter details for student 5:

Roll Number: 4

Name: tina

Department: ice

Course 1 name (enter 'NA' if no more courses): os

Course 1 credits: 4

Course 1 grade: 7

Course 2 name (enter 'NA' if no more courses): adsa

Course 2 credits: 3
Course 2 grade: 8

Course 3 name (enter 'NA' if no more courses): cc

Course 3 credits: 2

Course 3 grade: 10

Course 4 name (enter 'NA' if no more courses): NA

Enter details for student 6:

Roll Number: 6

Name: gauri

Department: mech

Course 1 name (enter 'NA' if no more courses): math

Course 1 credits: 4

Course 1 grade: 9

Course 2 name (enter 'NA' if no more courses): chem

Course 2 credits: 3

Course 2 grade: 8

Course 3 name (enter 'NA' if no more courses): NA

Enter details for student 7:

Roll Number: 9

Name: riya

Department: prod

Course 1 name (enter 'NA' if no more courses): eg

Course 1 credits: 2

Course 1 grade:

10

Course 2 name (enter 'NA' if no more courses): eng

Course 2 credits: 4

Course 2 grade: 9

Course 3 name (enter 'NA' if no more courses): dsa

Course 3 credits: 3

Course 3 grade: 9

Course 4 name (enter 'NA' if no more courses): NA
```

LAB SESSION 2
25/07/2024

```
Menu:
1. Insert student records
2. Create GPA column
3. Delete a course
4. Insert a course
5. Update course name
6. Calculate GPA
7. Upgrade grade point
8. Generate grade report
9. Exit
Enter your choice: 2
GPA column created for all students.

Menu:
1. Insert student records
2. Create GPA column
3. Delete a course
4. Insert a course
5. Update course name
6. Calculate GPA
7. Upgrade grade point
8. Generate grade report
9. Exit
Enter your choice: 2
GPA column created for all students.
2. Create GPA column
3. Delete a course
4. Insert a course
5. Update course name
6. Calculate GPA
```

```
7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 2

GPA column created for all students.

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 2

GPA column created for all students.

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 2

GPA column created for all students.

8. Generate grade report

9. Exit

Enter your choice: 2

GPA column created for all students.

9. Exit

Enter your choice: 2

GPA column created for all students.

Enter your choice: 2

GPA column created for all students.


Menu:

1. Insert student records
```

```
2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 3

Enter roll number of student to delete a course: 2

Deleting course co for student 2.

Deleting course dsd for student 2.


Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 4

Enter roll number of student to insert a course: 2

Enter new course name: DBMS

Enter new course credits: 3

Enter new course grade: 9

Enter new course name: DBMS

Enter new course credits: 3

Enter new course grade: 9
```

LAB SESSION 2
25/07/2024

```
Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 5

Enter the old course name to update: os

Enter the new course name: co

Updated course name from os to co for student 1.

Updated course name from os to co for student 4.


Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 6

GPA calculated for all students.


Menu:
```

```
1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 7

Enter roll number of student to upgrade grade points: 4

Upgraded grade for course co to 8 for student 4.

Upgraded grade for course adsa to 9 for student 4.

Upgraded grade for course cc to 11 for student 4.


Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 6

GPA calculated for all students.


Menu:

1. Insert student records

2. Create GPA column
```

```
3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 8

Enter roll number of student to generate grade report: 4

Grade Report for tina (Roll No: 4):

Course 1: co, Credits: 4, Grade: 8

Course 2: adsa, Credits: 3, Grade: 9

Course 3: cc, Credits: 2, Grade: 11


Menu:

1. Insert student records

2. Create GPA column

3. Delete a course

4. Insert a course

5. Update course name

6. Calculate GPA

7. Upgrade grade point

8. Generate grade report

9. Exit

Enter your choice: 9

Exiting...
```

LAB SESSION 2
25/07/2024

Structured Query Language (SQL) DDL Commands

1. Create a Student schema using the student details given in Q.No.1 and execute the following basic queries.

Note: When defining the schema, exclude the following columns: Course_credit and Course_grade for all the courses.

Make sure you have the following constraints: Course is declared in char datatype.

DoB should be in date (dd/mm/yyyy) format. Provide a not-null constraint for dob.

Email should have the following format: xxx@nitt.edu

a. Insert at least 5 student records into the Student table.

b. Delete Course2 and Course3 attributes from the Student table.

c. Insert two new columns DoB and email into the Student table.

d. Change Course1 datatype to varchar2.

e. Update the column name ‘Std_rollno’ to ‘Std_rno’.

f. Update all student records who pursue a course named “DBMS” to “OS”.

g. Delete a student record with student name starting with letter ‘S’.

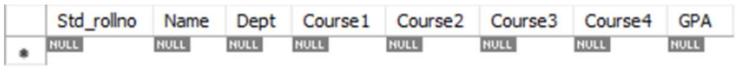
h. Display all records in which a student has born after the year 2005.

i. Simulate RENAME, COMMENT, TRUNATE and DROP.

Commands:

```
CREATE TABLE Student (
   Std_rollno INT PRIMARY KEY,
   Name VARCHAR(50),
   Dept VARCHAR(50),
   Course1 CHAR(50),
   Course2 CHAR(50),
   Course3 CHAR(50),
   Course4 CHAR(50),
   GPA FLOAT,
   DoB DATE NOT NULL,
   Email VARCHAR(100) CHECK (Email LIKE '%@nitt.edu')
);
```

LAB SESSION 2
25/07/2024



	Std_rollno	Name	Dept	Course1	Course2	Course3	Course4	GPA
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

a. INSERT INTO Student (Std_rollno, Name, Dept, Course1, Course2, Course3, Course4, GPA, DoB, Email) VALUES

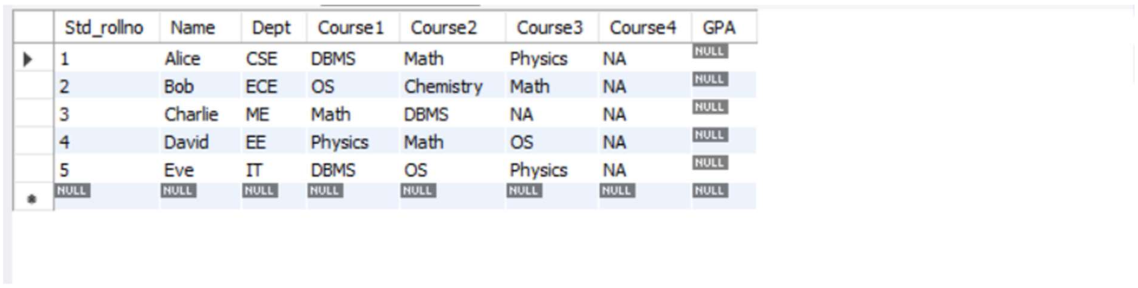
   (1, 'Alice', 'CSE', 'DBMS', 'Math', 'Physics', 'NA', NULL, TO_DATE('01/01/2000', 'DD/MM/YYYY'), 'alice@nitt.edu'),

   (2, 'Bob', 'ECE', 'OS', 'Chemistry', 'Math', 'NA', NULL, TO_DATE('05/02/2001', 'DD/MM/YYYY'), 'bob@nitt.edu'),

   (3, 'Charlie', 'ME', 'Math', 'DBMS', 'NA', 'NA', NULL, TO_DATE('15/03/2002', 'DD/MM/YYYY'), 'charlie@nitt.edu'),

   (4, 'David', 'EE', 'Physics', 'Math', 'OS', 'NA', NULL, TO_DATE('25/04/2003', 'DD/MM/YYYY'), 'david@nitt.edu'),

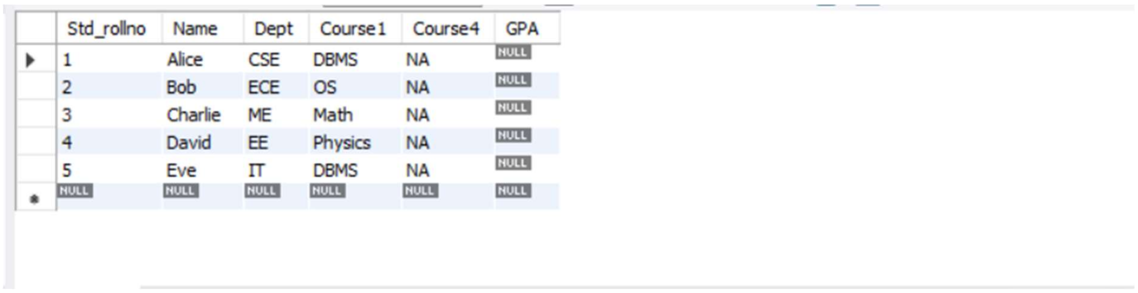
   (5, 'Eve', 'IT', 'DBMS', 'OS', 'Physics', 'NA', NULL, TO_DATE('05/05/2004', 'DD/MM/YYYY'), 'eve@nitt.edu');



	Std_rollno	Name	Dept	Course1	Course2	Course3	Course4	GPA
▶	1	Alice	CSE	DBMS	Math	Physics	NA	NULL
	2	Bob	ECE	OS	Chemistry	Math	NA	NULL
	3	Charlie	ME	Math	DBMS	NA	NA	NULL
	4	David	EE	Physics	Math	OS	NA	NULL
	5	Eve	IT	DBMS	OS	Physics	NA	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

b. ALTER TABLE Student DROP COLUMN Course2;

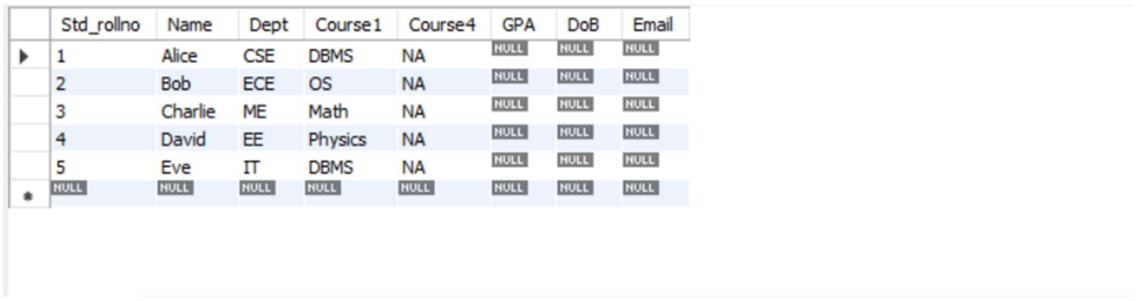
   ALTER TABLE Student DROP COLUMN Course3;



	Std_rollno	Name	Dept	Course1	Course4	GPA
▶	1	Alice	CSE	DBMS	NA	NULL
	2	Bob	ECE	OS	NA	NULL
	3	Charlie	ME	Math	NA	NULL
	4	David	EE	Physics	NA	NULL
	5	Eve	IT	DBMS	NA	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

c. ALTER TABLE Student ADD (DoB DATE NOT NULL, Email VARCHAR(100) CHECK (Email LIKE '%@nitt.edu'));

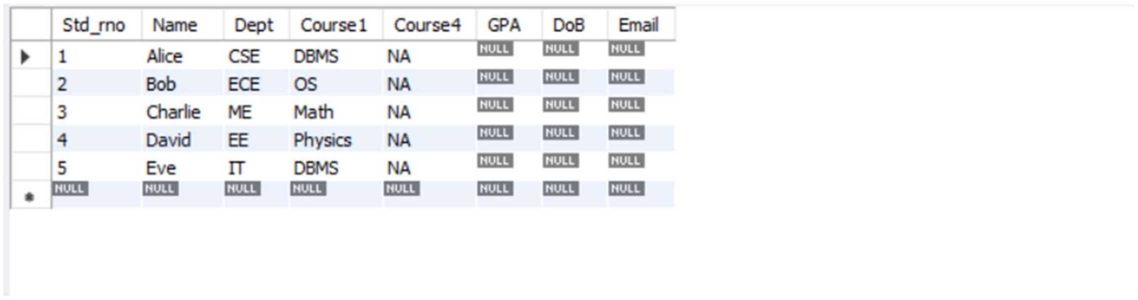
LAB SESSION 2
25/07/2024



Std_rollno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	DBMS	NA	NULL	NULL	NULL
2	Bob	ECE	OS	NA	NULL	NULL	NULL
3	Charlie	ME	Math	NA	NULL	NULL	NULL
4	David	EE	Physics	NA	NULL	NULL	NULL
5	Eve	IT	DBMS	NA	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

d. ALTER TABLE Student MODIFY Course1 VARCHAR2(50);

e. ALTER TABLE Student RENAME COLUMN Std_rollno TO Std_rno;



Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	DBMS	NA	NULL	NULL	NULL
2	Bob	ECE	OS	NA	NULL	NULL	NULL
3	Charlie	ME	Math	NA	NULL	NULL	NULL
4	David	EE	Physics	NA	NULL	NULL	NULL
5	Eve	IT	DBMS	NA	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

f. UPDATE Student

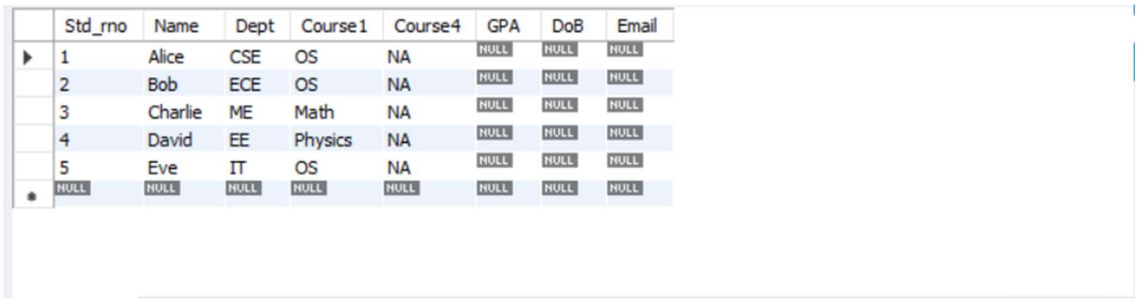
SET Course1 = 'OS'

WHERE Course1 = 'DBMS';

UPDATE Student

SET Course4 = 'OS'

WHERE Course4 = 'DBMS';



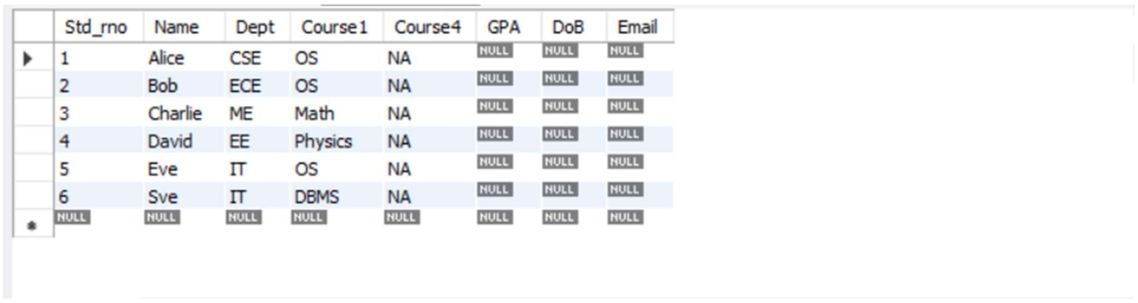
Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	OS	NA	NULL	NULL	NULL
2	Bob	ECE	OS	NA	NULL	NULL	NULL
3	Charlie	ME	Math	NA	NULL	NULL	NULL
4	David	EE	Physics	NA	NULL	NULL	NULL
5	Eve	IT	OS	NA	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

g. DELETE FROM Student

WHERE Name LIKE 'S%';

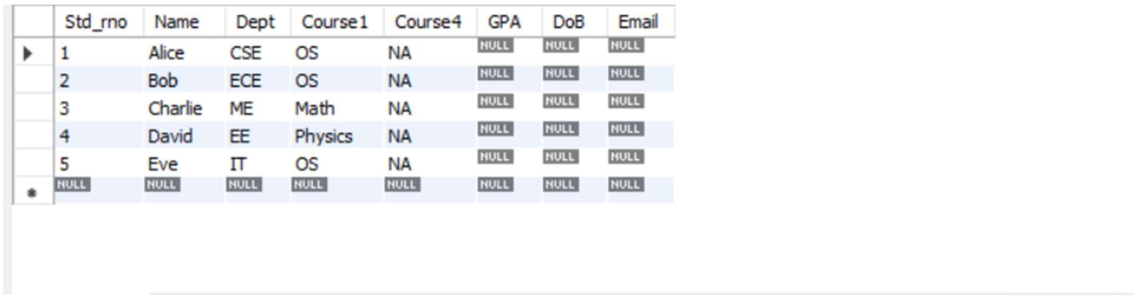
Before deletion:

LAB SESSION 2
25/07/2024



Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	OS	NA	NULL	NULL	NULL
2	Bob	ECE	OS	NA	NULL	NULL	NULL
3	Charlie	ME	Math	NA	NULL	NULL	NULL
4	David	EE	Physics	NA	NULL	NULL	NULL
5	Eve	IT	OS	NA	NULL	NULL	NULL
6	Sve	IT	DBMS	NA	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

After deletion:

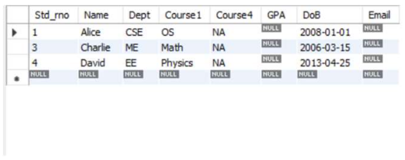


Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	OS	NA	NULL	NULL	NULL
2	Bob	ECE	OS	NA	NULL	NULL	NULL
3	Charlie	ME	Math	NA	NULL	NULL	NULL
4	David	EE	Physics	NA	NULL	NULL	NULL
5	Eve	IT	OS	NA	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

h. SELECT *

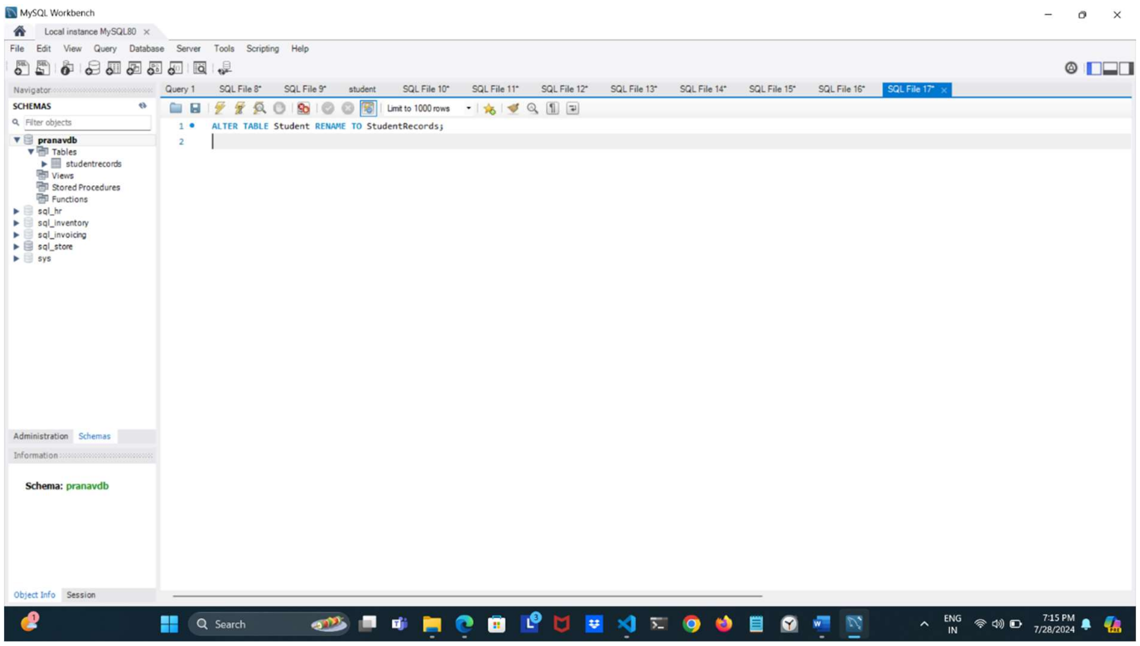
FROM Student

WHERE DoB > TO_DATE('31/12/2005', 'DD/MM/YYYY');



Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
1	Alice	CSE	OS	NA	NULL	2008-01-01	NULL
3	Charlie	ME	Math	NA	NULL	2006-03-15	NULL
4	David	EE	Physics	NA	NULL	2013-04-25	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. ALTER TABLE Student RENAME TO StudentRecords;


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Query 1 SQL File 8* SQL File 9* student SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15* SQL File 16* SQL File 17*
Limit to 1000 rows
Administration Schemas
Information
Object Info Session
Search
ENG IN
7:15 PM 7/28/2024

```
ALTER TABLE Student RENAME TO StudentRecords;
```

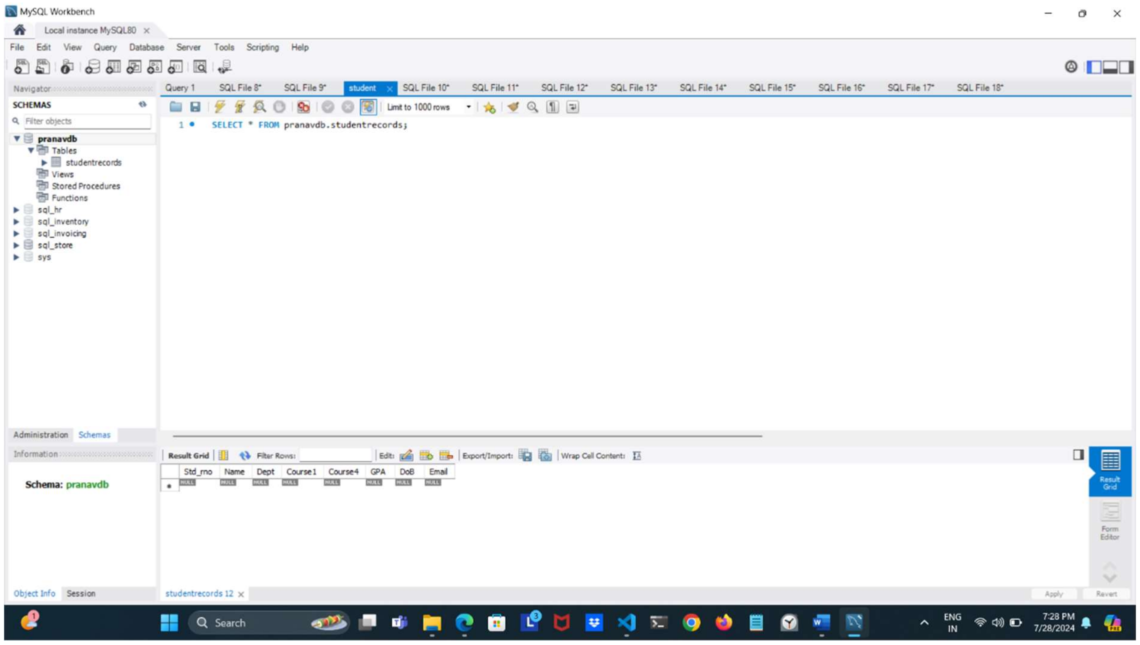
```
pranavdb
  Tables
    studentrecords
  Views
  Stored Procedures
  Functions
sql_hr
sql_inventory
sql_invoicing
sql_store
sys
```

Schema: pranavdb

j. COMMENT ON TABLE StudentRecords IS 'This table contains records of students including their courses and GPA';

   COMMENT ON COLUMN StudentRecords.Std_rno IS 'Student Roll Number';

k. TRUNCATE TABLE StudentRecords;



```
SELECT * FROM pranavdb.studentrecords;
```

Std_rno	Name	Dept	Course1	Course4	GPA	DoB	Email
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

l. DROP TABLE StudentRecords;