C Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_COURSES 4
#define MAX_STUDENTS 100
typedef struct {
char course_name[20];
int credits;
int grade;
} Course;
typedef struct {
char roll_no[10];
char name[50];
char department[20];
Course courses[MAX_COURSES];
int course_count;
float gpa;
} Student;
Student students [MAX_STUDENTS];
int student_count = 0;
void read_file() {
FILE *file = fopen("students.txt", "r");
if (file == NULL) {
printf("Could not open file for reading.\n");
return;
}
student_count = 0;
while(fscanf(file,"%[^,],%[^,],%[^,],",students[student_count].roll_no,
students[student_count].name, students[student_count]. department) != EOF) {
int i;
for (i = 0; i < MAX_COURSES; i++) {
if (fscanf(file, "%[^,],%d,%d,", students[student_count].courses[i].course_name,
  &students[student_count].courses[i].credits,
&students[student_count].courses[i].grade) == EOF) {
break;
```

```
}
}
students[student_count].course_count = i;
student_count++;
}
fclose(file);
}
void write_file() {
FILE *file = fopen("students.txt", "w");
if (file == NULL) {
printf("Could not open file for writing.\n");
return;
}
for (int i = 0; i < student_count; i++) {
fprintf(file, "%s,%s,%s,", students[i].roll_no, students[i].name, students[i].department);
for (int j = 0; j < students[i].course_count; j++) {
fprintf(file, "%s,%d,%d,", students[i].courses[j].course_name,
students[i].courses[j].credits, students[i].courses[j].grade);
}
fprintf(file, "\n");
}
fclose(file);
}
void insert_student() {
if (student_count >= MAX_STUDENTS) {
printf("Student limit reached.\n");
return;
}
printf("Enter roll number: ");
scanf("%s", students[student_count].roll_no);
printf("Enter name: ");
scanf("%s", students[student_count].name);
printf("Enter department: ");
scanf("%s", students[student_count].department);
for (int i = 0; i < MAX_COURSES; i++) {
printf("Enter course %d name: ", i + 1);
```

```
scanf("%s", students[student_count].courses[i].course_name);
printf("Enter course %d credits: ", i + 1);
scanf("%d", &students[student_count].courses[i].credits);
printf("Enter course %d grade: ", i + 1);
scanf("%d", &students[student_count].courses[i].grade);
students[student_count].course_count++;
char more;
if (i < MAX_COURSES - 1) {
printf("Do you want to enter more courses? (y/n): ");
scanf(" %c", &more);
if (more == 'n') break;
}
}
student_count++;
write_file();
printf("Student record inserted.\n");
}
void create_gpa_column() {
for (int i = 0; i < student_count; i++) {
float total_points = 0;
int total_credits = 0;
for (int j = 0; j < students[i].course_count; j++) {
total_points += students[i].courses[j].credits * students[i].courses[j].grade;
total_credits += students[i].courses[j].credits;
}
students[i].gpa = total_points / total_credits;
}
write_file();
printf("GPA column created.\n");
}
void deregister_course() {
char roll_no[10];
printf("Enter roll number: ");
scanf("%s", roll_no);
for (int i = 0; i < student_count; i++) {
if (strcmp(students[i].roll_no, roll_no) == 0) {
```

```
if (students[i].course_count == 4) {
printf("Enter course name to deregister: ");
char course_name[20];
scanf("%s", course_name);
for (int j = 0; j < students[i].course_count; j++) {
if (strcmp(students[i].courses[j].course_name, course_name) == 0) {
for (int k = j; k < students[i].course_count - 1; k++) {
students[i].courses[k] = students[i].courses[k + 1];
}
students[i].course_count--;
write_file();
printf("Course deregistered.\n");
return;
}
printf("Course not found.\n");
return;
} else {
printf("Student does not have 4 courses.\n");
return;
}
}
}
printf("Student not found.\n");
}
void insert_course() {
char roll_no[10];
printf("Enter roll number: ");
scanf("%s", roll_no);
for (int i = 0; i < student_count; i++) {
if (strcmp(students[i].roll_no, roll_no) == 0) {
if (students[i].course_count < 4) {
printf("Enter new course name: ");
scanf("%s", students[i].courses[students[i].course_count].course_name);
printf("Enter course credits: ");
scanf("%d", &students[i].courses[students[i].course_count].credits);
printf("Enter course grade: ");
```

```
scanf("%d", &students[i].courses[students[i].course_count].grade);
students[i].course_count++;
write_file();
printf("Course added.\n");
return;
} else {
printf("Student already has 4 courses.\n");
return;
}
}
}
printf("Student not found.\n");
}
void update_course_name() {
for (int i = 0; i < 2; i++) {
char roll_no[10];
printf("Enter roll number for student %d: ", i + 1);
scanf("%s", roll_no);
for (int j = 0; j < student_count; j++) {
if (strcmp(students[j].roll_no, roll_no) == 0) {
printf("Enter old course name: ");
char old_course_name[20];
scanf("%s", old_course_name);
for (int k = 0; k < students[j].course_count; k++) {
if (strcmp(students[j].courses[k].course_name, old_course_name) == 0) {
printf("Enter new course name: ");
scanf("%s", students[j].courses[k].course_name);
write_file();
printf("Course name updated.\n");
return;
}
}
printf("Course not found.\n");
return;
}
}
```

```
printf("Student not found.\n");
}
}
void upgrade_grade() {
char roll_no[10];
printf("Enter roll number: ");
scanf("%s", roll_no);
for (int i = 0; i < student_count; i++) {
if (strcmp(students[i].roll_no, roll_no) == 0) {
for (int j = 0; j < students[i].course_count; j++) {
if (students[i].courses[j].grade == 7) {
students[i].courses[j].grade = 8;
write_file();
printf("Grade upgraded.\n");
return;
}
}
printf("No course with grade 7 found.\n");
return;
}
}
printf("Student not found.\n");
}
void calculate_updated_gpa() {
char roll_no[10];
printf("Enter roll number: ");
scanf("%s", roll_no);
for (int i = 0; i < student_count; i++) {
if (strcmp(students[i].roll_no, roll_no) == 0) {
float total_points = 0;
int total_credits = 0;
for (int j = 0; j < students[i].course_count; j++) {
total_points += students[i].courses[j].credits * students[i].courses[j].grade;
total_credits += students[i].courses[j].credits;
}
students[i].gpa = total_points / total_credits;
```

```
printf("Updated GPA: %.2f\n", students[i].gpa);
return;
}
}
printf("Student not found.\n");
}
void generate_grade_report() {
char identifier[50];
printf("Enter roll number or name: ");
scanf("%s", identifier);
for (int i = 0; i < student_count; i++) {
if (strcmp(students[i].roll_no, identifier) == 0 || strcmp(students[i].name, identifier) == 0) {
printf("Roll No: %s\n", students[i].roll_no);
printf("Name: %s\n", students[i].name);
printf("Department: %s\n", students[i].department);
for (int j = 0; j < students[i].course_count; j++) {
printf("Course: %s, Credits: %d, Grade: %d\n", students[i].courses[j].course_name,
students[i].courses[j].credits, students[i].courses[j].grade);
}
printf("GPA: %.2f\n", students[i].gpa);
return;
}
}
printf("Student not found.\n");
}
int main() {
int choice;
read_file();
while (1) {
printf("\nMenu:\n");
printf("1. Insert student record\n");
printf("2. Create GPA column\n");
printf("3. Deregister course\n");
printf("4. Insert course\n");
printf("5. Update course name\n");
printf("6. Upgrade grade\n");
```

```
printf("7. Calculate updated GPA\n");
printf("8. Generate grade report\n");
printf("9. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
insert_student();
break;
case 2:
create_gpa_column();
break;
case 3:
deregister_course();
break;
case 4:
insert_course();
break;
case 5:
update_course_name();
break;
case 6:
upgrade_grade();
break;
case 7:
calculate_updated_gpa();
break;
case 8:
generate_grade_report();
break;
case 9:
exit(0);
default:
printf("Invalid choice.\n");
}
}
```

return 0;

}

Output:

Menu:

1. Insert student record
2. Create GPA column
3. Deregister course
4. Insert course
5. Update course name
6. Upgrade grade
7. Calculate updated GPA
8. Generate grade report
9. Exit

Enter your choice: 1

Enter roll number: 18

Enter name: rocky

Enter department: cse

Enter course 1 name: analog

Enter course 1 credits: 3

Enter course 1 grade: 10

Do you want to enter more courses? (y/n): y

Enter course 2 name: digital

Enter course 2 credits: 3

Enter course 2 grade: 9

Do you want to enter more courses? (y/n): y

Enter course 3 name: verilog

Enter course 3 credits: 2

Enter course 3 grade: 8

Do you want to enter more courses? (y/n): n

Student record inserted.

Enter your choice: 8

Enter roll number or name: 18

Roll No: 18

Name: rocky

Department: cse

Course: analog, Credits: 3, Grade: 10

Course: digital, Credits: 3, Grade: 9

Course: verilog, Credits: 2, Grade: 8

GPA: 0.00

Enter your choice: 2

GPA column created.

Enter your choice: 8

Enter roll number or name: 18

Roll No: 18

Name: rocky

Department: cse

Course: analog, Credits: 3, Grade: 10

Course: digital, Credits: 3, Grade: 9

Course: verilog, Credits: 2, Grade: 8

GPA: 9.13

Enter your choice: 5

Enter roll number for student 1: 18

Enter old course name: analog

Enter new course name: analog_device

Course name updated.

Records that are created in file student .txt

},cd,"c:\Users\dinesh\OneDDSA\",DSA\",0,0,if,0,0,$?),0,0,gcc,0,0,

d2.c,-o,d2,},0,0,if,0,0,$?),0,0,.\d2,0,0,

,,,

1,rocky,cse,sql,3,8,python,3,8,physics,2,9,

2,rocky,cse,python,3,8,sql,3,8,physics,2,10,

18,rocky,cse,analog_device,3,10,digital,3,9,verilog,2,8,

2.

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.

SQL Query

```
CREATE TABLE Student (
 Std_rollno CHAR(10) PRIMARY KEY,
 Std_name VARCHAR(50),
 Dept VARCHAR(20),
 Course1 CHAR(20),
 Course2 CHAR(20),
 Course3 CHAR(20),
 Course4 CHAR(20) -- Include a maximum of four courses
);
-- Step 2: Insert at least 5 student records into the Student table
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4)
VALUES ('001', 'John Doe', 'CSE', 'DBMS', 'OS', 'Networks', 'Compiler');
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4)
VALUES ('002', 'Jane Smith', 'ECE', 'DSP', 'VLSI', 'Embedded', 'Control');
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4)
VALUES ('003', 'Alice Brown', 'EEE', 'Power', 'Machines', 'Circuits', 'Signals');
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4)
VALUES ('004', 'Bob White', 'MECH', 'Thermo', 'Design', 'Dynamics', 'Kinematics');
INSERT INTO Student (Std_rollno, Std_name, Dept, Course1, Course2, Course3, Course4)
VALUES ('005', 'Sandy ', 'CIVIL', 'Structures', 'Hydraulics', 'Materials', 'Geotech');
SELECT * FROM Student;
```

Output:

```
+------------+--------------+-------+------------+------------+-----------+------------+
| Std_rollno | Std_name | Dept | Course1 | Course2 | Course3 | Course4 |
+------------+--------------+-------+------------+------------+-----------+------------+
| 001 | John Doe | CSE | DBMS | OS | Networks | Compiler |
| 002 | Jane Smith | ECE | DSP | VLSI | Embedded | Control |
| 003 | Alice Brown | EEE | Power | Machines | Circuits | Signals |
| 004 | Bob White | MECH | Thermo | Design | Dynamics | Kinematics |
| 005 | Sandy | CIVIL | Structures | Hydraulics | Materials | Geotech |
+------------+--------------+-------+------------+------------+-----------+------------+
```

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

```
ALTER TABLE Student
DROP COLUMN Course2;
ALTER TABLE Student
DROP COLUMN Course3;
SELECT * FROM Student;
```

Output:

```
+------------+--------------+-------+------------+------------+
| Std_rollno | Std_name | Dept | Course1 | Course4 |
+------------+--------------+-------+------------+------------+
| 001 | John Doe | CSE | DBMS | Compiler |
| 002 | Jane Smith | ECE | DSP | Control |
| 003 | Alice Brown | EEE | Power | Signals |
| 004 | Bob White | MECH | Thermo | Kinematics |
| 005 | Sandy | CIVIL | Structures | Geotech |
+------------+--------------+-------+------------+------------+
```

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

SQL QUERY

```
ALTER TABLE Student
ADD DoB DATE,
ADD email VARCHAR(50);
```

```
UPDATE Student
SET DoB = '2004-12-23', email = 'abc1@nitt.edu'
WHERE Std_rollno = '001';
UPDATE Student
SET DoB = '2004-11-23', email = 'abc2@nitt.edu'
WHERE Std_rollno = '002';
UPDATE Student
SET DoB = '2004-10-23', email = 'abc3@nitt.edu'
WHERE Std_rollno = '003';
UPDATE Student
SET DoB = '2004-09-23', email = 'abc4@nitt.edu'
WHERE Std_rollno = '004';
UPDATE Student
SET DoB = '2004-08-23', email = 'abc5@nitt.edu'
WHERE Std_rollno = '005';
SELECT *FROM Student
OUTPUT
+------------+---------------+-------+------------+------------+------------+---------------+
| Std_rollno | Std_name | Dept | Course1 | Course4 | DoB | email |
+------------+---------------+-------+------------+------------+------------+---------------+
| 001 | John Doe | CSE | DBMS | Compiler | 2006-12-23 | abc1@nitt.edu |
| 002 | Jane Smith | ECE | DSP | Control | 2004-11-23 | abc2@nitt.edu |
| 003 | Alice Brown | EEE | Power | Signals | 2006-10-23 | abc3@nitt.edu |
| 004 | Bob White | MECH | Thermo | Kinematics | 2004-09-23 | abc4@nitt.edu |
| 005 | Sandy | CIVIL | Structures | Geotech | 2004-08-23 | abc5@nitt.edu |
+------------+---------------+-------+------------+------------+------------+---------------+
```

d. Change Course1 datatype to varchar2.

In MySQL, the VARCHAR type is used instead of VARCHAR2

SQL Query

```
ALTER TABLE Student
MODIFY Course1 VARCHAR(20);
```

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

SQL Query

```
ALTER TABLE Student
CHANGE COLUMN Std_rollno Std_rno CHAR(10);
SELECT *FROM Student
```

Output

| Std_rno | Std_name | Dept | Course1 | Course4 | DoB | email |
|---|---|---|---|---|---|---|
| 001 | John Doe | CSE | DBMS | Compiler | 2006-12-23 | abc1@nitt.edu |
| 002 | Jane Smith | ECE | DSP | Control | 2004-11-23 | abc2@nitt.edu |
| 003 | Alice Brown | EEE | Power | Signals | 2006-10-23 | abc3@nitt.edu |
| 004 | Bob White | MECH | Thermo | Kinematics | 2004-09-23 | abc4@nitt.edu |
| 005 | Sandy | CIVIL | Structures | Geotech | 2004-08-23 | abc5@nitt.edu |

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

SQL Query

```
UPDATE Student
SET Course1 = 'OS'
WHERE Course1 = 'DBMS';
SELECT *FROM Student
```

| Std_rno | Std_name | Dept | Course1 | Course4 | DoB | email |
|---|---|---|---|---|---|---|
| 001 | John Doe | CSE \|OS | Compiler | 2006-12-23 | abc1@nitt.edu |
| 002 | Jane Smith | ECE | DSP | Control | 2004-11-23 | abc2@nitt.edu |
| 003 | Alice Brown | EEE | Power | Signals | 2006-10-23 | abc3@nitt.edu |
| 004 | Bob White | MECH | Thermo | Kinematics | 2004-09-23 | abc4@nitt.edu |
| 005 | Sandy | CIVIL | Structures | Geotech | 2004-08-23 | abc5@nitt.edu |

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

SQL Query

```
DELETE FROM Student
WHERE Std_name LIKE 'S%';
SELECT *FROM Student
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
| Std_rno | Std_name | Dept | Course1 | Course4 | DoB | email |
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
| 001 | John Doe | CSE |OS | Compiler | 2006-12-23 | abc1@nitt.edu |
| 002 | Jane Smith | ECE | DSP | Control | 2004-11-23 | abc2@nitt.edu |
| 003 | Alice Brown | EEE | Power | Signals | 2006-10-23 | abc3@nitt.edu |
| 004 | Bob White | MECH | Thermo | Kinematics | 2004-09-23 | abc4@nitt.edu |
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
```

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

SQL Query

```
SELECT * FROM Student
WHERE YEAR(DoB) > 2005;
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
| Std_rno | Std_name | Dept | Course1 | Course4 | DoB | email |
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
| 001 | John Doe | CSE |OS | Compiler | 2006-12-23 | abc1@nitt.edu |
| 003 | Alice Brown | EEE | Power | Signals | 2006-10-23 | abc3@nitt.edu |
+-----------+--------------+-------+-----------+-----------+-----------+--------------+
```

i. Simulate RENAME, COMMENT, TRUNCATE and DROP

RENAME

• Rename a Table

```
RENAME TABLE Student TO StudentInfo;
```

• Rename a Column

```
ALTER TABLE StudentInfo
CHANGE COLUMN Std_rno Std_rollno CHAR(10);
```

COMMENT

• Add a Comment to a Table

```
ALTER TABLE StudentInfo
```

COMMENT = 'Table storing student records including their courses and contact information.';

• Add a Comment to a Column

```
ALTER TABLE StudentInfo
MODIFY COLUMN Std_rollno CHAR(10) COMMENT 'Unique roll number for each student';
```

TRUNCATE

The TRUNCATE TABLE statement removes all rows from a table but does not remove the table itself.

• Truncate a Table

```
TRUNCATE TABLE StudentInfo;
```

DROP

The DROP statement completely removes a table or column, including all its data and structure.

• Drop a Table

```
DROP TABLE StudentInfo;
```

• Drop a Column

```
ALTER TABLE StudentInfo
DROP COLUMN DoB
```