**#Session: 2**

# 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.

1. Insert at least 5 student records.
2. Create a column ‘GPA’ for all the students.
3. For a student with four courses, delete(deregister) a course name.
4. For the same student you deleted in ‘c’, insert a new course name.
5. Update the name of a course for two different students.
6. Calculate GPA of all students using the GPA formula. Refer the following:

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

1. Upgrade the grade point of a student who has secured ‘7’ in a course.
2. Calculate the updated GPA of the student in ‘g’.
3. Generate a Grade report of a student given the roll no. or name.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_COURSES 4

#define MAX\_STUDENTS 100

typedef struct {

char name[50];

int credits;

int grade;

} Course;

typedef struct {

int rollno;

char name[50];

char dept[10];

Course courses[MAX\_COURSES];

int course\_count;

float gpa;

} Student;

Student students[MAX\_STUDENTS];

int student\_count = 0;

void readStudentsFromFile(const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

printf("Error opening file.\n");

return;

}

student\_count = 0;

while (fscanf(file, "%d,%49[^,],%9[^,]", &students[student\_count].rollno, students[student\_count].name, students[student\_count].dept) == 3) {

for (int i = 0; i < MAX\_COURSES; i++) {

if (fscanf(file, ",%49[^,],%d,%d", students[student\_count].courses[i].name, &students[student\_count].courses[i].credits, &students[student\_count].courses[i].grade) != 3) {

break;

}

students[student\_count].course\_count++;

}

student\_count++;

}

fclose(file);

}

void writeStudentsToFile(const char \*filename) {

FILE \*file = fopen(filename, "w");

if (!file) {

printf("Error opening file.\n");

return;

}

for (int i = 0; i < student\_count; i++) {

fprintf(file, "%d,%s,%s", students[i].rollno, students[i].name, students[i].dept);

for (int j = 0; j < students[i].course\_count; j++) {

fprintf(file, ",%s,%d,%d", students[i].courses[j].name, students[i].courses[j].credits, students[i].courses[j].grade);

}

fprintf(file, "\n");

}

fclose(file);

}

void insertStudent() {

if (student\_count >= MAX\_STUDENTS) {

printf("Maximum student limit reached.\n");

return;

}

Student new\_student;

printf("Enter roll number: ");

scanf("%d", &new\_student.rollno);

printf("Enter name: ");

scanf("%s", new\_student.name);

printf("Enter department: ");

scanf("%s", new\_student.dept);

printf("Enter number of courses (3 or 4): ");

scanf("%d", &new\_student.course\_count);

if (new\_student.course\_count < 3 || new\_student.course\_count > 4) {

printf("Invalid number of courses.\n");

return;

}

for (int i = 0; i < new\_student.course\_count; i++) {

printf("Enter course %d name: ", i + 1);

scanf("%s", new\_student.courses[i].name);

printf("Enter course %d credits: ", i + 1);

scanf("%d", &new\_student.courses[i].credits);

printf("Enter course %d grade: ", i + 1);

scanf("%d", &new\_student.courses[i].grade);

}

students[student\_count++] = new\_student;

writeStudentsToFile("students.txt");

}

void calculateGPA(Student \*student) {

int total\_credits = 0;

int total\_points = 0;

for (int i = 0; i < student->course\_count; i++) {

total\_credits += student->courses[i].credits;

total\_points += student->courses[i].credits \* student->courses[i].grade;

}

student->gpa = (float)total\_points / total\_credits;

}

void calculateAllGPAs() {

for (int i = 0; i < student\_count; i++) {

calculateGPA(&students[i]);

}

writeStudentsToFile("students.txt");

}

void deregisterCourse(int rollno) {

for (int i = 0; i < student\_count; i++) {

if (students[i].rollno == rollno && students[i].course\_count == 4) {

printf("Enter course name to deregister: ");

char course\_name[50];

scanf("%s", course\_name);

int found = 0;

for (int j = 0; j < students[i].course\_count; j++) {

if (strcmp(students[i].courses[j].name, course\_name) == 0) {

found = 1;

for (int k = j; k < students[i].course\_count - 1; k++) {

students[i].courses[k] = students[i].courses[k + 1];

}

students[i].course\_count--;

break;

}

}

if (!found) {

printf("Course not found.\n");

} else {

writeStudentsToFile("students.txt");

printf("Course deregistered successfully.\n");

}

return;

}

}

printf("Student with roll number %d having four courses not found.\n", rollno);

}

void insertCourse(int rollno) {

for (int i = 0; i < student\_count; i++) {

if (students[i].rollno == rollno && students[i].course\_count == 3) {

printf("Enter new course name: ");

scanf("%s", students[i].courses[students[i].course\_count].name);

printf("Enter new course credits: ");

scanf("%d", &students[i].courses[students[i].course\_count].credits);

printf("Enter new course grade: ");

scanf("%d", &students[i].courses[students[i].course\_count].grade);

students[i].course\_count++;

writeStudentsToFile("students.txt");

printf("Course inserted successfully.\n");

return;

}

}

printf("Student with roll number %d having three courses not found.\n", rollno);

}

void updateCourseName() {

for (int i = 0; i < 2; i++) {

printf("Enter roll number for student %d: ", i + 1);

int rollno;

scanf("%d", &rollno);

int found = 0;

for (int j = 0; j < student\_count; j++) {

if (students[j].rollno == rollno) {

printf("Enter old course name to update: ");

char old\_name[50];

scanf("%s", old\_name);

printf("Enter new course name: ");

char new\_name[50];

scanf("%s", new\_name);

for (int k = 0; k < students[j].course\_count; k++) {

if (strcmp(students[j].courses[k].name, old\_name) == 0) {

strcpy(students[j].courses[k].name, new\_name);

found = 1;

break;

}

}

if (!found) {

printf("Course not found for student %d.\n", rollno);

} else {

printf("Course name updated successfully.\n");

}

break;

}

}

if (!found) {

printf("Student with roll number %d not found.\n", rollno);

}

}

writeStudentsToFile("students.txt");

}

void upgradeGrade(int rollno) {

for (int i = 0; i < student\_count; i++) {

if (students[i].rollno == rollno) {

for (int j = 0; j < students[i].course\_count; j++) {

if (students[i].courses[j].grade == 7) {

students[i].courses[j].grade = 8;

}

}

writeStudentsToFile("students.txt");

printf("Grades upgraded successfully.\n");

return;

}

}

printf ("Student with roll number %d not found.\n", rollno);

}

void generateGradeReport (int rollno) {

for (int i = 0; i < student\_count; i++) {

if (students[i].rollno == rollno) {

printf("Grade Report for Roll Number: %d\n", rollno);

printf("Name: %s\n", students[i].name);

printf("Department: %s\n", students[i].dept);

printf("Courses:\n");

for (int j = 0; j < students[i].course\_count; j++) {

printf("%s: Credits = %d, Grade = %d\n", students[i].courses[j].name, students[i].courses[j].credits, students[i].courses[j].grade);

}

printf("GPA: %.2f\n", students[i].gpa);

return;

}

}

printf("Student with roll number %d not found.\n", rollno);

}

void menu() {

int choice;

do {

printf("\n1. Insert Student Records\n");

printf("2. Calculate GPAs\n");

printf("3. Deregister a Course\n");

printf("4. Insert a New Course\n");

printf("5. Update Course Names\n");

printf("6. Upgrade Grade\n");

printf("7. Generate Grade Report\n");

printf("8. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

insertStudent();

break;

case 2:

calculateAllGPAs();

break;

case 3: {

int rollno;

printf("Enter roll number: ");

scanf("%d", &rollno);

deregisterCourse(rollno);

break;

}

case 4: {

int rollno;

printf("Enter roll number: ");

scanf("%d", &rollno);

insertCourse(rollno);

break;

}

case 5:

updateCourseName();

break;

case 6: {

int rollno;

printf("Enter roll number: ");

scanf("%d", &rollno);

upgradeGrade(rollno);

break;

}

case 7: {

int rollno;

printf("Enter roll number: ");

scanf("%d", &rollno);

generateGradeReport(rollno);

break;

}

case 8:

printf("Exiting...\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 8);

}

int main() {

readStudentsFromFile("students.txt");menu();

return 0;

}

# Structured Query Language (SQL)

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

CREATE TABLE student (

rollnum INT PRIMARY KEY,

name VARCHAR(50),

dept VARCHAR(10),

dob DATE NOT NULL,

email VARCHAR(50) CHECK (email LIKE '%@nitt.edu'),

course1 VARCHAR(50),

course2 VARCHAR(50),

course3 VARCHAR(50),

course4 VARCHAR(50)

);

mysql> describe student;

+---------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+-------------+------+-----+---------+-------+

| rollnum | int | NO | PRI | NULL | |

| name | varchar(50) | YES | | NULL | |

| dept | varchar(10) | YES | | NULL | |

| dob | date | NO | | NULL | |

| email | varchar(50) | YES | | NULL | |

| course1 | varchar(50) | YES | | NULL | |

| course2 | varchar(50) | YES | | NULL | |

| course3 | varchar(50) | YES | | NULL | |

| course4 | varchar(50) | YES | | NULL | |

+---------+-------------+------+-----+---------+-------+

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

INSERT INTO student (rollnum, name, dept, dob, email, course1, course2, course3, course4)

VALUES

(106122034, 'deepak', 'cse', '2022-08-22', '106122034@nitt.edu', 'DBMS', 'OS', 'CYK', 'FLAT'),

(106122036, 'dev', 'cse', '2022-08-22', '106122036@nitt.edu', 'DBMS', 'M1', 'M2', 'CHEM'),

(106122122, 'sudhanshu', 'cse', '2022-08-22', '106122122@nitt.edu', 'DBMS', 'PHYSICS', 'CHEM', 'MECH'),

(106122056, 'himanshu', 'cse', '2022-08-22', '106122056@nitt.edu', 'ROL', 'THKI', 'CHIK', 'M3');

mysql> select \* from student;

+-----------+-----------+------+------------+--------------------+---------+---------+---------+---------+

| rollnum | name | dept | dob | email | course1 | course2 | course3 | course4 |

+-----------+-----------+------+------------+--------------------+---------+---------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | DBMS | OS | CYK | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | DBMS | M1 | M2 | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | THKI | CHIK | M3 |

| 106122122 | sudhanshu | cse | 2022-08-22 | 106122122@nitt.edu | DBMS | PHYSICS | CHEM | MECH |

+-----------+-----------+------+------------+--------------------+---------+---------+---------+---------+

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

ALTER TABLE student

DROP COLUMN course2,

DROP COLUMN course3;

mysql> select \* from student;

+-----------+-----------+------+------------+--------------------+---------+---------+

| rollnum | name | dept | dob | email | course1 | course4 |

+-----------+-----------+------+------------+--------------------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | DBMS | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | DBMS | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | M3 |

| 106122122 | sudhanshu | cse | 2022-08-22 | 106122122@nitt.edu | DBMS | MECH |

+-----------+-----------+------+------------+--------------------+---------+---------+

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

Already done while make the table

1. Change Course1 datatype to varchar2.

ALTER TABLE student

MODIFY course1 VARCHAR2(50);

mysql> describe student;

+---------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+-------------+------+-----+---------+-------+

| rollnum | int | NO | PRI | NULL | |

| name | varchar(50) | YES | | NULL | |

| dept | varchar(10) | YES | | NULL | |

| dob | date | NO | | NULL | |

| email | varchar(50) | YES | | NULL | |

| course1 | varchar(50) | YES | | NULL | |

| course4 | varchar(50) | YES | | NULL | |

+---------+-------------+------+-----+---------+-------+

1. rows in set (0.00 sec)
2. Update the column name ‘Std\_rollno’ to ‘Std\_rno’.

ALTER TABLE student

RENAME COLUMN rollnum TO std\_rno;

mysql> select \* from student;

+-----------+-----------+------+------------+--------------------+---------+---------+

| std\_rno | name | dept | dob | email | course1 | course4 |

+-----------+-----------+------+------------+--------------------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | DBMS | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | DBMS | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | M3 |

| 106122122 | sudhanshu | cse | 2022-08-22 | 106122122@nitt.edu | DBMS | MECH |

+-----------+-----------+------+------------+--------------------+---------+---------+

1. rows in set (0.00 sec)
2. Update all student records who pursue a course named “DBMS” to “OS”.

UPDATE student

SET course1 = 'OS'

WHERE course1 = 'DBMS';

mysql> select \* from student;

+-----------+-----------+------+------------+--------------------+---------+---------+

| std\_rno | name | dept | dob | email | course1 | course4 |

+-----------+-----------+------+------------+--------------------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | OS | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | OS | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | M3 |

| 106122122 | sudhanshu | cse | 2022-08-22 | 106122122@nitt.edu | OS | MECH |

+-----------+-----------+------+------------+--------------------+---------+---------+

1. rows in set (0.00 sec)
2. Delete a student record with student name starting with letter ‘S’.

DELETE FROM student

WHERE name LIKE 'S%';

mysql> select \* from student;

+-----------+----------+------+------------+--------------------+---------+---------+

| std\_rno | name | dept | dob | email | course1 | course4 |

+-----------+----------+------+------------+--------------------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | OS | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | OS | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | M3 |

+-----------+----------+------+------------+--------------------+---------+---------+

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

SELECT \* FROM student

WHERE dob > '2005-01-01';

mysql> SELECT \* FROM student WHERE dob > '2005-01-01';

+-----------+----------+------+------------+--------------------+---------+---------+

| std\_rno | name | dept | dob | email | course1 | course4 |

+-----------+----------+------+------------+--------------------+---------+---------+

| 106122034 | deepak | cse | 2022-08-22 | 106122034@nitt.edu | OS | FLAT |

| 106122036 | dev | cse | 2022-08-22 | 106122036@nitt.edu | OS | CHEM |

| 106122056 | himanshu | cse | 2022-08-22 | 106122056@nitt.edu | ROL | M3 |

+-----------+----------+------+------------+--------------------+---------+---------+

1. rows in set (0.00 sec)

I. Simulate DROP and TRUNATE commands with the database you created.

DROP TABLE student;

TRUNCATE TABLE student;