Course Performance Information System

DBSL mini project

Team members:

Name	Reg. no.	Roll no.	Section
Anika Jha	210905366	54	CSE, section C
Dhruthi K	210905388	58	

1 ABSTRACT:

This project involves design and implementation of a system to record course performance information. The system will enable teachers to input and update marks for each subject for a student. It will also enable them to modify, insert and delete student records, update the cut-off for subjects as well as view information about a given subject or student. The project has been implemented using Python, Tkinter (GUI) and Oracle SQL.

Problem Statement: Design and implementation of course performance management system for updation of cut-offs, grade display and modification of student records.

Data Requirements:

- Student Information: Name, ID, Contact details, Address, Enrolment details
- Course Information: Course name, Course code, Branch
- Instructor Information: Name, ID, Subjects taught
- Grade Information: Subject-wise cut-offs

Functional Requirements:

- Computation of sum of marks to get the total course marks
- Insertion, updation and deletion of student records
- Provision for the number of assignments/exams to not be predefined
- Appropriate grading for subjects
- Provision for cut-offs to be specified for various grades

2 ER DIAGRAM AND RELATIONAL SCHEMA

Relational Schema:

student = (<u>studentid</u>, firstname, lastname, address, dob, mobile, email)

instructor = (<u>instructorid</u>, fullname, course, branch)

subject = (<u>subjectid</u>, subjectname, course, branch, cutoff, semester, numofassignments)

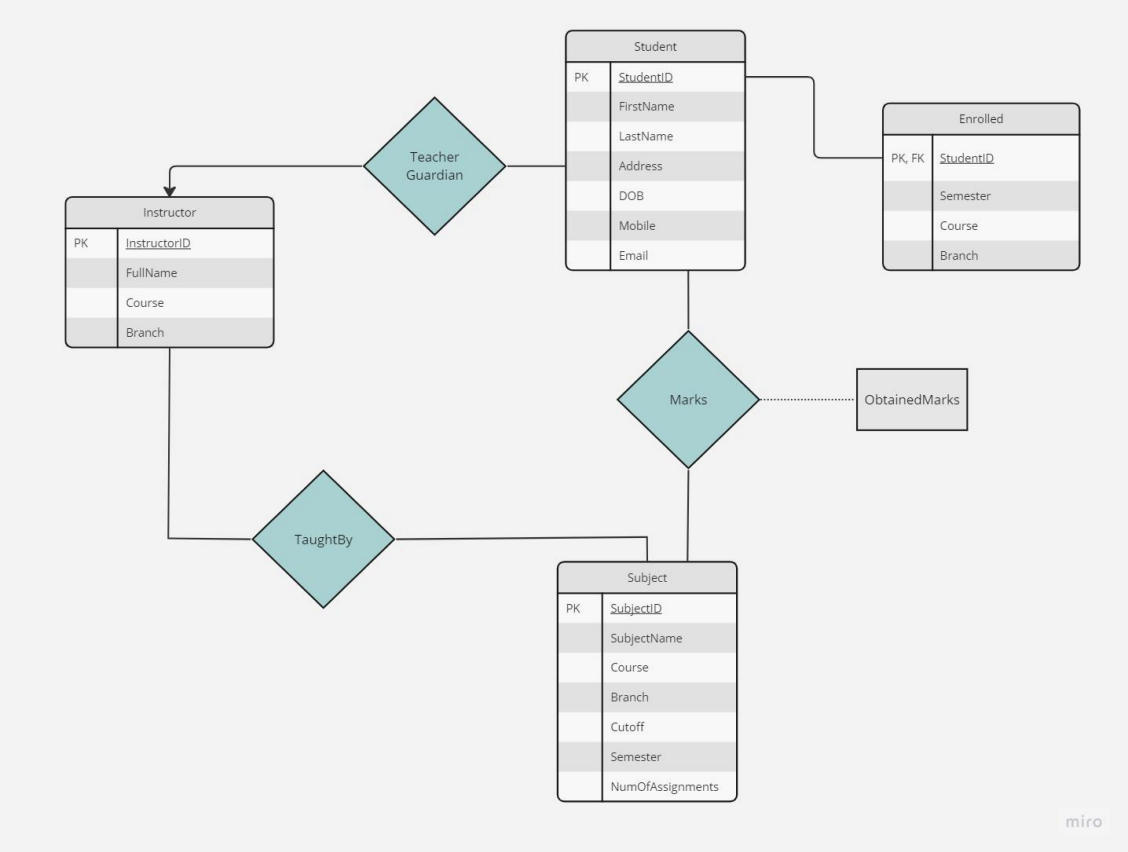
enrolled = (<u>studentid</u>, semester, course, branch)

taughtby = (<u>subjectid</u>, <u>instructorid</u>)

marks = (<u>studentid</u>, <u>subjectid</u>, obtainedmarks)

teacherguardian = (<u>studentid</u>, instructorid)

ER Diagram:



3 DDL COMMANDS TO CREATE TABLES

```
CREATE TABLE Student (
   StudentID VARCHAR2(10) PRIMARY KEY,
   FirstName VARCHAR2(30),
   LastName VARCHAR2(30),
   Address VARCHAR2(100),
   DOB DATE,
   Mobile VARCHAR2(20),
   Email VARCHAR2(50)
);
CREATE TABLE Enrolled (
   StudentID VARCHAR2(10) PRIMARY KEY,
   Semester VARCHAR2(10),
   Course VARCHAR2(20),
   Branch VARCHAR2(30),
   CONSTRAINT fk_enrolled
        FOREIGN KEY (StudentID)
        REFERENCES Student(StudentID)
        ON DELETE CASCADE );
```

```
CREATE TABLE Instructor (
    InstructorID VARCHAR2(10) PRIMARY KEY,
    FullName VARCHAR2(50),
    Course VARCHAR2(20),
    Branch VARCHAR2(30)
);
CREATE TABLE TaughtBy (
    SubjectID VARCHAR2(30),
    InstructorID VARCHAR2(10),
    CONSTRAINT fks_taughtby
        FOREIGN KEY (SubjectID)
        REFERENCES Subject(SubjectID)
        ON DELETE CASCADE,
    CONSTRAINT fkt_taughtby
        FOREIGN KEY (InstructorID)
        REFERENCES Instructor(InstructorID)
        ON DELETE CASCADE
);
CREATE TABLE TeacherGuardian (
    StudentID VARCHAR2(10),
    InstructorID VARCHAR2(10),
    CONSTRAINT fks_teachergrd
        FOREIGN KEY (StudentID)
        REFERENCES Student(StudentID)
        ON DELETE CASCADE,
    CONSTRAINT fkt_ teachergrd
        FOREIGN KEY (InstructorID)
        REFERENCES Instructor(InstructorID)
        ON DELETE CASCADE
);
CREATE TABLE Subject (
    SubjectID VARCHAR2(10) PRIMARY KEY,
    SubjectName VARCHAR2(30),
    Course VARCHAR2(20),
    Branch VARCHAR2(30),
    Cutoff NUMBER,
    Semester NUMBER,
    NumOfAssignments NUMBER );
```

```
CREATE TABLE Marks (
    StudentID VARCHAR2(10),
    SubjectID VARCHAR2(10),
    ObtainedMarks NUMBER(10),
    CONSTRAINT fk_marks
        FOREIGN KEY (StudentID)
        REFERENCES Student(StudentID)
        ON DELETE CASCADE,
    CONSTRAINT fks_marks
        FOREIGN KEY (SubjectID)
        REFERENCES Subject(SubjectID)
        ON DELETE CASCADE
);
```

4 SAMPLE DATA

Student:

StudentID	FirstName	LastName	Address	DOB	Mobile	Email
ST0001	Emma	Davis	123 Main St, Anytown USA	10-05-99	555-1234	emma.davis@example.com
ST0002	Michael	Garcia	456 High St, Anytown USA	18-02-98	555-5678	michael.garcia@example.com

Instructor:

InstructorID	FullName	Course	Branch
IN0026	Taylor Patel	B.Tech	Electronics
IN0011	Justin Peterson	B.Tech	Computer Science

TeacherGuardian:

StudentID	InstructorID
ST0021	IN0026
ST0064	IN0011

Subject:

SubjectID	SubjectName	Course	Branch	CutOff	Semester	NumAssignments
EE204	Circuit Design	B.Tech	Electronics	35	2	4
ME606	Refrigeration	B.Tech	Mechanical	30	6	6

Enrolled:

StudentID	Semester	Course	Branch
ST0098	2	B.Sc	Physics
ST0020	4	B.Tech	Electronics

TaughtBy:

SubjectID	InstructorID
EE404	IN0026
CS610	IN0011

Marks:

StudentID	SubjectID	ObtainedMarks
ST0001	ME210	58
ST0062	CH404	91

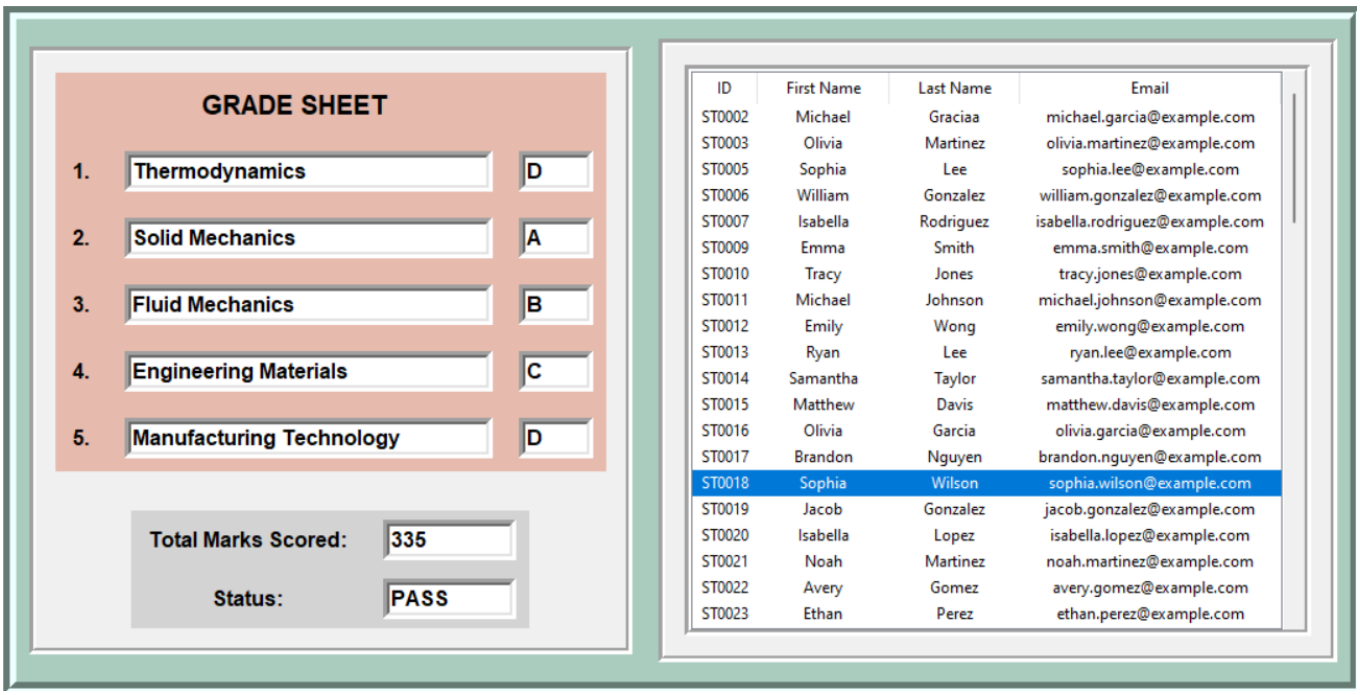
5 SQL Queries

A number of select, insert, update and delete queries have been utilised in the program to modify, insert, delete and display student and subject entries along with other information. Some of them are listed below:

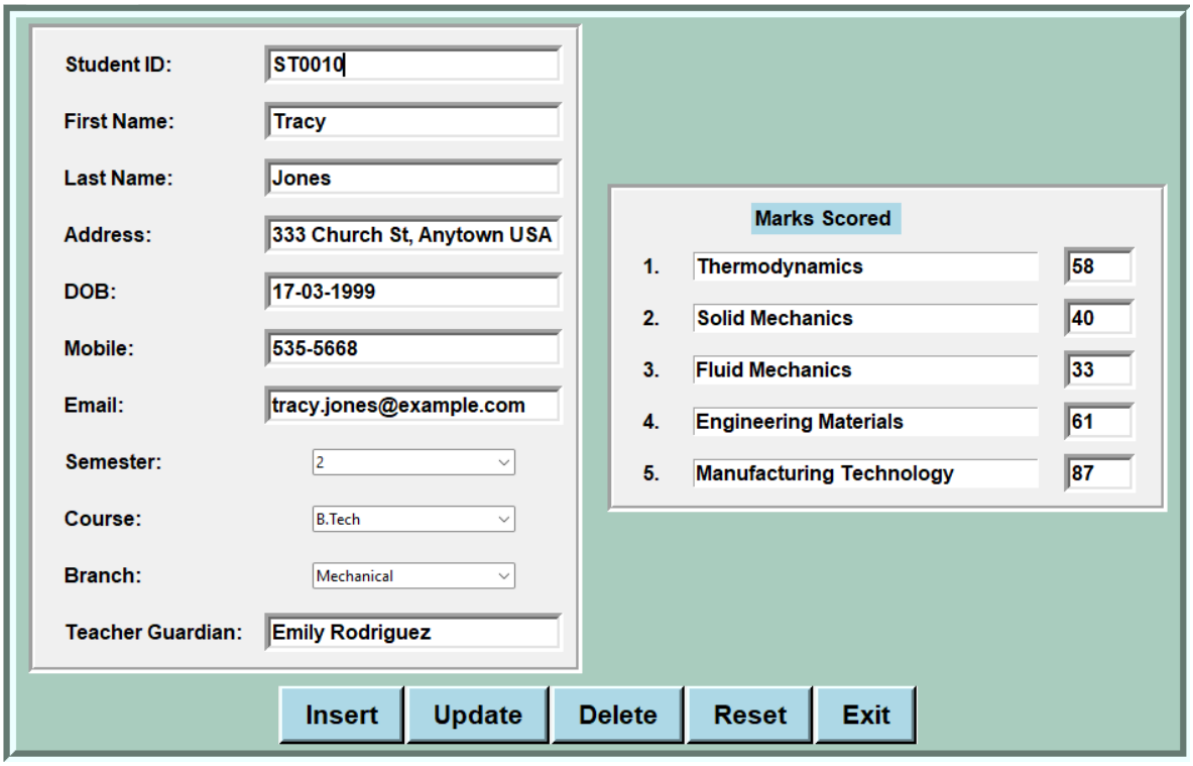
- `SELECT * FROM Subject WHERE SubjectID = '{sub_id}' ORDER BY SubjectID`
- `SELECT * FROM Subject WHERE Branch='{branch}' AND Semester={sem} ORDER BY SubjectID`
- `SELECT FullName FROM Instructor WHERE InstructorID = (SELECT InstructorID FROM`
- `SELECT COUNT(*) FROM Enrolled WHERE Branch = '{branch}' AND Semester = '{sem}'`
- `DELETE FROM Student WHERE StudentID='{stu_id}'`
- `UPDATE Student SET {attr[i]}=TO_DATE('{val}', 'DD-MM-YYYY') WHERE StudentID='{stu_id}'`
- `UPDATE Subject SET NumOfAssignments={int(numA)} WHERE SubjectID='{sub_id}'`
- `INSERT INTO  Student VALUES ('{stu_id}', '{f_name}', '{l_name}', '{addr}', TO_DATE('{dob}', 'DD-MM-YYYY'), '{mob}', '{email}')`

6 UI Design

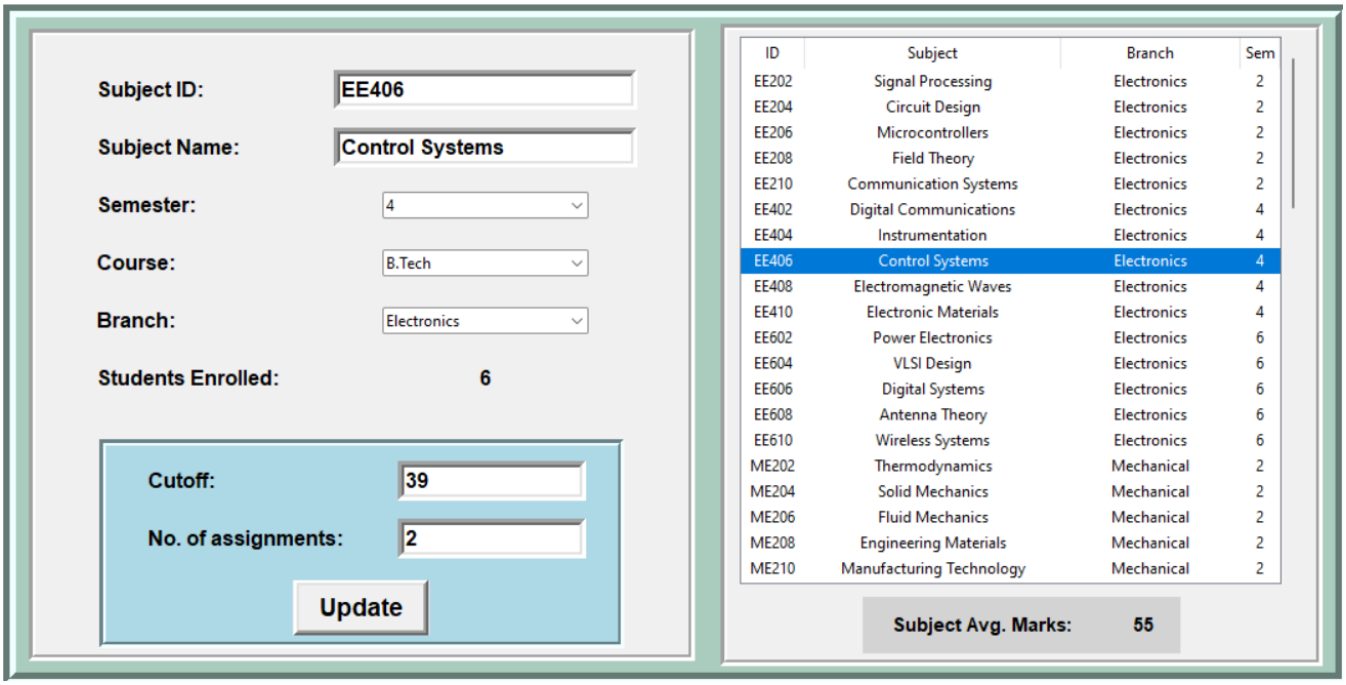
Screen 1: Grade Sheet display and Student selection



Screen 2: Student details viewing, updation, insertion and deletion



Screen 3: Subject details viewing, updation of cut-off



7 PL/SQL

7.1 TRIGGERS

Trigger on Student Table:

```
CREATE OR REPLACE TRIGGER logStudent
BEFORE INSERT OR UPDATE OR DELETE ON Student
FOR EACH ROW
BEGIN
CASE
WHEN INSERTING THEN
    INSERT INTO LogStudentChange
    VALUES(SYSDATE,:NEW.StudentID, :NEW.FirstName, :NEW.LastName, :NEW.Address, :NEW.DOB,
:NEW.Mobile, :NEW.Email);
WHEN UPDATING OR DELETING THEN
    INSERT INTO LogStudentChange
    VALUES(SYSDATE,:OLD.StudentID, :OLD.FirstName, :OLD.LastName, :OLD.Address, :OLD.DOB,
:OLD.Mobile, :OLD.Email);
END CASE;
END;
/
```

Trigger on Subject Table:

```
CREATE OR REPLACE TRIGGER logSubject
BEFORE UPDATE OF Cutoff, NumOfAssignments ON Subject
FOR EACH ROW
BEGIN
CASE
```

```
WHEN UPDATING THEN
    INSERT INTO LogSubjectChange
    VALUES(SYSDATE,:OLD.SubjectID, :OLD.Cutoff, :OLD.NumOfAssignments);
END CASE;
END;
/
```

7.2 FUNCTIONS

Function to calculate total marks:

```
CREATE OR REPLACE FUNCTION calcTotMarks(p_student_id IN VARCHAR2) RETURN NUMBER IS
  v_marks NUMBER(10);
  v_tot_marks NUMBER(10) := 0;

  CURSOR c_marks IS
    SELECT ObtainedMarks FROM Marks WHERE StudentID = p_student_id;
BEGIN
  FOR c IN c_marks LOOP
    v_marks := c.ObtainedMarks;
    v_tot_marks := v_tot_marks + v_marks;
  END LOOP;
  RETURN v_tot_marks;
END;
/
```

Function to calculate pass or fail:

```
CREATE OR REPLACE FUNCTION calcStatus(p_student_id IN VARCHAR2) RETURN NUMBER IS
    v_marks NUMBER(10);
    v_cutoff NUMBER(10) := 0;
    v_sub_id VARCHAR2(10);
    res NUMBER := 1;
    CURSOR c_marks IS
    SELECT * FROM Marks WHERE StudentID = p_student_id;
BEGIN
    FOR c IN c_marks LOOP
        v_marks := c.ObtainedMarks;
        v_sub_id := c.SubjectID;
        SELECT Cutoff INTO v_cutoff FROM Subject WHERE SubjectID = v_sub_id;
        IF v_marks < v_cutoff THEN
            res := 0;
        END IF;
    END LOOP;
    RETURN res;
END;
/
```

8 DB CONNECTIVITY

Database connection through cx_Oracle:

```
import cx_Oracle
cx_Oracle.init_oracle_client(lib_dir=r"path\to\oracle\21c\binaries")
dsn_tns = cx_Oracle.makedsn('localhost', '1521', service_name='xe')
conn = cx_Oracle.connect(user=r'system', password='<redacted>', dsn=dsn_tns)
cursor = conn.cursor()

cursor.execute(<query statement>)
```

PL/SQL calls through cx_Oracle:

```
var = cursor.callfunc("function_name", return-type, [“param1”, “param2”])
```

Data access through cx_Oracle:

```
cursor.execute(f"SELECT col1, col2 FROM table WHERE col3='{value}'")
rows=cursor.fetchall()
```

9 REFERENCES

https://cx-oracle.readthedocs.io/en/latest/

https://docs.python.org/3/library/tk.html