PL/SQL Project

Student Name: Aman **UID: 24MCC20015**
Branch: MCA (CCD) **Section/Group: 24MCD-1A**
Semester: 1st **Date of Performance: 21/10/2024**
Subject Name: PL/SQL **Subject Code: 24CAP-602**

Student Management System

Aim:

The aim of the *Student Management System* project using PL/SQL is to design and implement a robust database system that efficiently manages and organizes student information. This system will utilize PL/SQL (Procedural Language/SQL) to perform data processing, complex queries, and automated tasks within the database, providing a streamlined solution for educational institutions to handle various student-related operations.

Task to be done:

A. Requirement Analysis

- **Gather Requirements**:
  - Identify key data points (e.g., student details, courses, grades, enrollments).
  - Determine required functionalities, including CRUD operations and report generation.
- **Define Functional Requirements**:
  - List functionalities for data entry, updates, and retrieval.
  - Document automation requirements, such as automated calculations for GPA.

B. Database Design

- **Define Database Schema**:
  - Identify and create tables (Students, Courses, Enrollments, Grades).
- **Establish Relationships**:
  - Set up primary and foreign keys.
  - Define relationships between tables (e.g., one-to-many between Students and Enrollments).
- **Define Constraints**:
  - Set data constraints (e.g., NOT NULL, UNIQUE) for data integrity.

C. Implement PL/SQL Code

- **Create CRUD Procedures**:
  - Write PL/SQL procedures for adding, updating, and deleting records in each table.
- **Build Custom Functions**:
  - Develop functions to retrieve specific information, such as enrolled courses and grade summaries.
- **Add Input Validation and Error Handling**:
  - Implement input validation for data integrity (e.g., valid age ranges).
  - Include error handling for scenarios like constraint violations.

D. Code for experiment/practical:

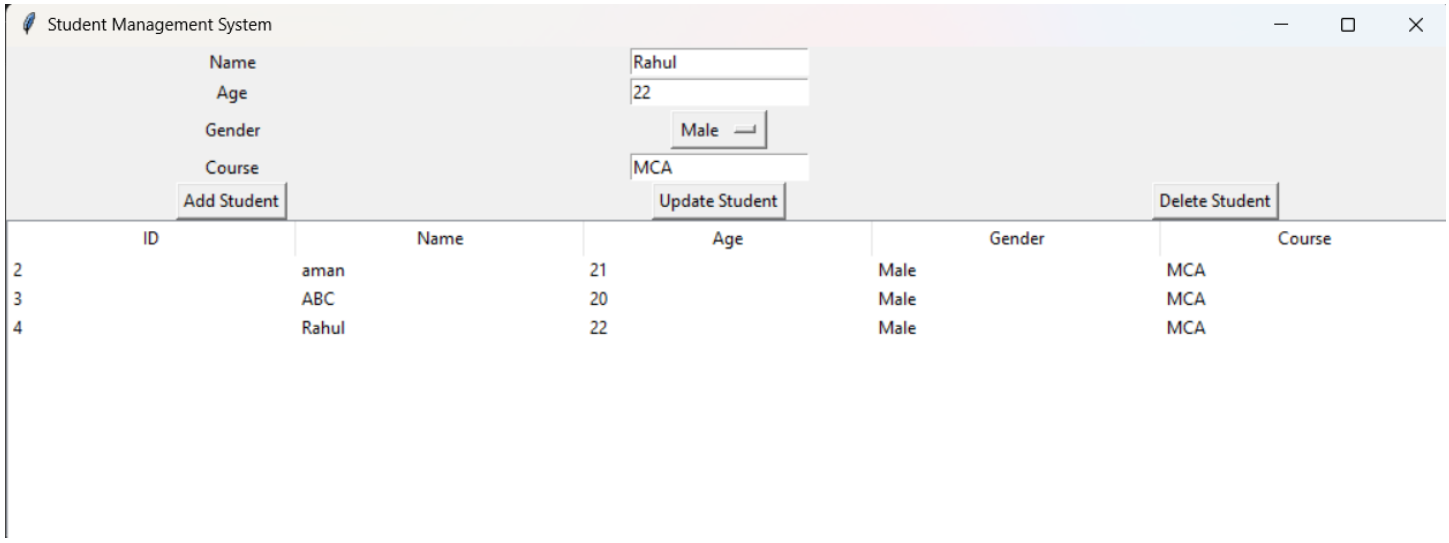
```
import sqlite3
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
# Set up the SQLite Database
def initialize_db():
    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            age INTEGER,
            gender TEXT,
            course TEXT
        )
    """)
    conn.commit()
    conn.close()
# Define Functions for CRUD Operations
def add_student():
    name = entry_name.get()
    age = entry_age.get()
    gender = gender_var.get()
    course = entry_course.get()

    if name and age and gender and course:
        conn = sqlite3.connect('students.db')
        cursor = conn.cursor()
        cursor.execute("INSERT INTO students (name, age, gender, course) VALUES (?, ?, ?, ?)",
                       (name, age, gender, course))
        conn.commit()
```

```
        conn.close()
        messagebox.showinfo("Success", "Student added successfully!")
        display_students()
    else:
        messagebox.showwarning("Input Error", "All fields are required.")
# Function to display Student in Treeview
def display_students():
    for row in tree.get_children():
        tree.delete(row)

    conn = sqlite3.connect('students.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM students")
    rows = cursor.fetchall()
    for row in rows:
        tree.insert("", END, values=row)
    conn.close()

# Function to update Student's data
def update_student():
    selected_item = tree.selection()
    if selected_item:
        student_id = tree.item(selected_item)['values'][0]
        name = entry_name.get()
        age = entry_age.get()
        gender = gender_var.get()
        course = entry_course.get()

        conn = sqlite3.connect('students.db')
        cursor = conn.cursor()
        cursor.execute("UPDATE students SET name=?, age=?, gender=?, course=? WHERE id=?",
                       (name, age, gender, course, student_id))
        conn.commit()
        conn.close()
```

```
        messagebox.showinfo("Success", "Student data updated successfully!")
        display_students()
# Function to delete student's data
def delete_student():
    selected_item = tree.selection()
    if selected_item:
        student_id = tree.item(selected_item)['values'][0]
        conn = sqlite3.connect('students.db')
        cursor = conn.cursor()
        cursor.execute("DELETE FROM students WHERE id=?", (student_id,))
        conn.commit()
        conn.close()
        messagebox.showinfo("Success", "Student deleted successfully!")
        display_students()
    else:
        messagebox.showwarning("Selection Error", "No student selected.")
# Set Up the Tkinter GUI
root = Tk()
root.title("Student Management System")

# Input fields
Label(root, text="Name").grid(row=0, column=0)
entry_name = Entry(root)
entry_name.grid(row=0, column=1)

Label(root, text="Age").grid(row=1, column=0)
entry_age = Entry(root)
entry_age.grid(row=1, column=1)

Label(root, text="Gender").grid(row=2, column=0)
gender_var = StringVar()
OptionMenu(root, gender_var, "Male", "Female", "Other").grid(row=2, column=1)

Label(root, text="Course").grid(row=3, column=0)
```

```
entry_course = Entry(root)
entry_course.grid(row=3, column=1)

# Buttons for CRUD operations
Button(root, text="Add Student", command=add_student).grid(row=4, column=0)
Button(root, text="Update Student", command=update_student).grid(row=4, column=1)
Button(root, text="Delete Student", command=delete_student).grid(row=4, column=2)

# Treeview for displaying students
tree = ttk.Treeview(root, columns=("ID", "Name", "Age", "Gender", "Course"), show="headings")
tree.heading("ID", text="ID")
tree.heading("Name", text="Name")
tree.heading("Age", text="Age")
tree.heading("Gender", text="Gender")
tree.heading("Course", text="Course")
tree.grid(row=5, column=0, columnspan=3)

# Initialize database and display data
initialize_db()
display_students()

root.mainloop()
```

E. Output:



F. Learning outcomes:

- Proficiency in Database Design: Gain hands-on experience in designing a relational database, defining table structures, relationships, primary and foreign keys, and setting up data integrity constraints.
- PL/SQL Programming Skills: Develop strong PL/SQL programming skills by writing and implementing stored procedures, functions, triggers, and other scripts to perform data manipulation and enforce business logic.
- Understanding of Data Integrity and Constraints: Learn how to maintain data integrity through constraints, relational integrity, and triggers, ensuring accurate and reliable data within a multi-table system.
- Automation of Database Operations: Gain practical experience in automating repetitive tasks, calculations, and data updates using PL/SQL triggers, thereby improving the efficiency and reliability of data management processes.
- Debugging and Performance Optimization: Learn techniques for testing, debugging, and optimizing PL/SQL code to ensure efficiency, accuracy, and scalability, especially when handling large datasets and complex queries.
- Effective Reporting and Data Retrieval: Develop skills in data querying and reporting by creating complex SQL queries and PL/SQL scripts, as well as designing views, to facilitate easy and meaningful data retrieval for end-users.

Github Link: https://github.com/Amansharma5228/PL-SQL-Project