



**CHANDIGARH
UNIVERSITY**
Discover. Learn. Empower.

**NAAC
GRADE A+**
Accredited University



UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University

Mini Project ON Academic Performance Management

Submitted By:

Name: Atul Sinha

Balram Kumar

UID: 24MCA20010

24MCA20020

Branch: MCA

Section: 24MCA 1-A

Subject: PL/SQL

Semester: 1st

Submitted To:

Ms. Gagandeep Kaur



**CHANDIGARH
UNIVERSITY**

Discover. Learn. Empower.

**NAAC
GRADE A+**
Accredited University



UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University

University Institute of Computing
Chandigarh University, Gharuan, Mohali

Table of Contents

Topic	Page No.
ABSTRACT	3
CHAPTER-1: Introduction	3
CHAPTER-2: Background	4
CHAPTER-3: Objective	4
CHAPTER-4: Technology Used	5
CHAPTER-5: Requirements	6
CHAPTER-6: System Design	7
CHAPTER-7: Code	8
CHAPTER-8: Output	16
CHAPTER-9: Conclusion	19
CHAPTER-10: References	19

ABSTRACT

This project implements a graphical user interface (GUI) application for managing student academic records, leveraging the MySQL database for data storage and retrieval. Built with Python, the application employs the Tkinter library to create an intuitive interface that allows users to add, update, delete, and visualize student information. Key functionalities include inserting new student records, modifying existing entries, and removing students from the database. The application also features a visualization tool that provides graphical representations of student grades across various courses, enhancing data interpretation through bar charts generated by Matplotlib.

The system connects to a MySQL database, ensuring persistent data management and facilitating structured query operations. The database consists of two main tables: Students and Enrollments, linked to provide comprehensive insights into student performance. The application is designed to streamline academic performance management, making it accessible to educators and administrators for efficient student record handling.

1. Introduction

In today's educational environment, the management of student academic records has become increasingly complex due to the growing volume of data and the need for accurate tracking of student performance. Institutions face challenges in maintaining and updating records, which can hinder their ability to make informed decisions about curriculum development and student support. As a response to these challenges, this project introduces a comprehensive student academic performance management system that leverages modern technologies to streamline record-keeping processes.

The application is developed using Python and employs the Tkinter library to create a user-friendly graphical interface. This interface allows users to easily navigate through various functionalities, including adding, updating, and deleting student records. By connecting to a MySQL database, the system ensures data integrity and facilitates structured query operations, enabling users to retrieve and manipulate student information efficiently. The design focuses on usability, ensuring that educators and administrators can manage student data with minimal training and maximum effectiveness.

An essential feature of this system is its capability to visualize student performance data. By utilizing Matplotlib for graphical representation, the

application generates insightful bar charts that depict student grades across different courses. This feature not only enhances data interpretation but also empowers educators to identify trends and patterns in student performance. Overall, this project aims to provide an efficient solution for academic record management, ultimately supporting educational institutions in their efforts to foster student success and improve administrative processes.

2. Background

The effective management of student academic records has become a crucial concern for educational institutions as they navigate the complexities of an increasingly data-driven environment. With the rapid growth of student populations and the diversification of academic programs, schools and universities are confronted with the challenge of accurately tracking a wide range of student performance metrics. Traditional methods of record-keeping, often reliant on cumbersome paper systems or fragmented digital tools, can lead to inefficiencies, inaccuracies, and significant delays in retrieving essential information. These shortcomings not only impede administrative efficiency but also negatively impact the academic experience of students, underscoring the urgent need for integrated digital solutions that can streamline these processes. The emergence of relational database management systems, such as MySQL, has transformed how educational institutions approach data management by providing a robust and scalable framework for storing and organizing structured information. This allows for real-time access to student records, facilitating informed decision-making and promoting accountability within academic programs. Furthermore, the integration of data visualization techniques into these management systems has proven invaluable in enhancing data interpretation, enabling educators and administrators to quickly identify trends and patterns in student performance. Visual tools such as graphs and charts make complex data more accessible, fostering a deeper understanding of academic outcomes and allowing for targeted interventions where needed. In response to these challenges and opportunities, this project aims to develop a user-friendly application that combines the strengths of a relational database with effective data visualization capabilities, ultimately providing a comprehensive solution for managing student academic performance and supporting educational institutions in their mission to foster student success.

3. Objective

The primary objective of this project is to develop a comprehensive student academic performance management system that addresses the challenges faced

by educational institutions in managing student records. The specific objectives include:

- i. **Streamlined Record Management:** To create an application that allows for the efficient addition, updating, and deletion of student records, enabling administrative staff to manage data effortlessly and accurately.
- ii. **Data Integrity and Security:** To implement a robust database system using MySQL that ensures the integrity, security, and accessibility of student data, protecting sensitive information while providing authorized users with the necessary access.
- iii. **Enhanced Data Visualization:** To integrate data visualization tools that allow educators to generate graphical representations of student performance, facilitating better interpretation of academic data and enabling the identification of trends and patterns.
- iv. **User-Friendly Interface:** To design an intuitive graphical user interface using Tkinter that enhances user experience, making it easy for administrators and educators to navigate the application and perform essential functions with minimal training.
- v. **Informed Decision-Making:** To provide a platform that supports data-driven decision-making by allowing users to analyze student performance metrics and develop targeted interventions to improve academic outcomes.
- vi. **Comprehensive Reporting:** To develop reporting features that enable users to generate summaries and reports of student performance, aiding in institutional assessments and academic planning.

4. Technology Used

This student academic performance management system is developed using a combination of modern technologies to ensure efficiency, reliability, and usability. The key technologies utilized in this project include:

- i. **Python:** The core programming language used for developing the application, chosen for its versatility and extensive libraries that facilitate rapid development and integration with various systems.
- ii. **Tkinter:** A standard GUI toolkit for Python, Tkinter is employed to create a user-friendly graphical interface that allows users to interact seamlessly with the application. Its intuitive design makes it easy for administrators and educators to navigate through functionalities.
- iii. **MySQL:** A widely-used relational database management system that provides a robust platform for storing and managing student data. MySQL

ensures data integrity, security, and efficient querying capabilities, making it ideal for handling academic records.

- iv. **Pandas:** A powerful data manipulation and analysis library for Python, Pandas is utilized to handle data retrieval and processing from the MySQL database. It enables easy manipulation of structured data and simplifies the task of preparing data for visualization.
- v. **Matplotlib:** A popular plotting library in Python, Matplotlib is employed for creating visual representations of student performance data. This library facilitates the generation of graphs and charts, enabling educators to quickly interpret academic metrics.
- vi. **SQL:** The structured query language (SQL) is used for executing database operations such as data retrieval, insertion, updates, and deletions. SQL queries allow for efficient interaction with the MySQL database, ensuring seamless data management.

5. Requirements

Functional Requirements

- **User Authentication:** Secure login for users to access the application.
- **Student Record Management:**
 - Add, update, and delete student records.
 - Validate input data.
- **Data Retrieval:**
 - Display a list of all students in a table format.
 - Allow selection for detailed views and updates.
- **Grade Visualization:** Generate graphical representations of student grades.
- **Reporting:** Provide options to generate performance summary reports.

Non-Functional Requirements

- **Usability:** Intuitive interface with clear error messages.
- **Performance:** Efficient handling of large datasets with quick response times.
- **Security:** Protect sensitive data with secure authentication and encryption.
- **Compatibility:** Support multiple operating systems and screen resolutions.
- **Maintainability:** Well-documented code for easy future updates.
- **Scalability:** Accommodate growth in user numbers and data volume.

6. System Design

The system design for the student academic performance management application encompasses both the architectural and component design aspects to ensure a robust, scalable, and user-friendly solution.

i. Architectural Design

The application follows a three-tier architecture, which includes:

- **Presentation Layer:** This layer consists of the user interface developed using Tkinter. It allows users to interact with the system through forms and menus for managing student records, visualizing grades, and generating reports. The interface is designed to be intuitive and responsive, providing easy navigation for users.
- **Application Layer:** This middle layer contains the business logic of the application, which includes the core functionalities for managing student records, such as adding, updating, deleting, and retrieving student data. It processes user inputs, performs necessary validations, and coordinates communication between the presentation and data layers.
- **Data Layer:** The data layer utilizes MySQL as the relational database management system to store and manage student data. This layer handles all database interactions, including executing SQL queries for data manipulation and retrieval. It ensures data integrity and security while allowing for efficient storage and access of records.

ii. Component Design

The key components of the system include:

- **User Interface Components:**
 - Forms for adding and updating student information.
 - A table view (Treeview) for displaying the list of students and their details.
 - Graphical components for visualizing student grades using Matplotlib.
- **Database Design:**
 - **Student Table:** Contains fields such as student_id (Primary Key), first_name, last_name, age, gender, enrollment_year, and major.
 - **Course Table:** Stores information about courses, including course_id (Primary Key) and course_name.

- **Enrollments Table:** Links students to their courses, containing fields such as enrollment_id (Primary Key), student_id (Foreign Key), course_id (Foreign Key), and grade.
- **Functionality Components:**
 - **Record Management:** Functions for adding, updating, deleting, and retrieving student records.
 - **Data Visualization:** Functions that query the database and generate visual representations of grades for individual students.
 - **Reporting Module:** Functions that compile performance summaries and generate reports for educators and administrators.

7. Code

Code of MySQL

```
CREATE DATABASE StudentsAcademic;  
Use StudentsAcademic;
```

```
CREATE TABLE Students (  
    student_id INT PRIMARY KEY AUTO_INCREMENT,  
    first_name VARCHAR(50) NOT NULL,  
    last_name VARCHAR(50) NOT NULL,  
    age INT CHECK (age >= 0), -- Assuming age cannot be negative  
    gender ENUM('M', 'F', 'Other'),  
    enrollment_year YEAR NOT NULL,  
    major VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Courses (  
    course_id INT PRIMARY KEY AUTO_INCREMENT,  
    course_name VARCHAR(100) NOT NULL,  
    course_code VARCHAR(10) UNIQUE NOT NULL,  
    credits INT CHECK (credits > 0), -- Assuming credits must be positive  
    semester ENUM('Fall', 'Spring', 'Summer', 'Winter') NOT NULL  
);
```

```
CREATE TABLE Enrollments (  
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,  
    student_id INT,  
    course_id INT,  
    grade ENUM('A', 'B', 'C', 'D', 'F'),
```

```
semester ENUM('Fall', 'Spring', 'Summer', 'Winter'),  
FOREIGN KEY (student_id) REFERENCES Students(student_id) ON  
DELETE CASCADE,  
FOREIGN KEY (course_id) REFERENCES Courses(course_id) ON  
DELETE CASCADE  
);
```

```
CREATE TABLE Attendance (  
attendance_id INT PRIMARY KEY AUTO_INCREMENT,  
student_id INT,  
course_id INT,  
date DATE NOT NULL,  
status ENUM('Present', 'Absent') NOT NULL,  
FOREIGN KEY (student_id) REFERENCES Students(student_id) ON  
DELETE CASCADE,  
FOREIGN KEY (course_id) REFERENCES Courses(course_id) ON  
DELETE CASCADE  
);
```

```
INSERT INTO Students (first_name, last_name, age, gender, enrollment_year,  
major) VALUES  
( 'Aarav', 'Patel', 20, 'M', 2022, 'Computer Science'),  
( 'Vivaan', 'Sharma', 21, 'M', 2021, 'Mechanical Engineering'),  
( 'Anaya', 'Verma', 19, 'F', 2023, 'Biology'),  
( 'Diya', 'Reddy', 22, 'F', 2020, 'Psychology'),  
( 'Arjun', 'Iyer', 21, 'M', 2021, 'Mathematics'),  
( 'Aisha', 'Khan', 20, 'F', 2022, 'Commerce'),  
( 'Karan', 'Nair', 19, 'M', 2023, 'Electrical Engineering'),  
( 'Sneha', 'Mehta', 22, 'F', 2020, 'Chemistry'),  
( 'Raj', 'Singh', 21, 'M', 2021, 'Information Technology'),  
( 'Priya', 'Ghosh', 20, 'F', 2022, 'Fashion Design'),  
( 'Ravi', 'Jain', 23, 'M', 2019, 'Civil Engineering'),  
( 'Neha', 'Bhatia', 19, 'F', 2023, 'Physics'),  
( 'Siddharth', 'Choudhury', 20, 'M', 2022, 'Statistics'),  
( 'Isha', 'Malhotra', 21, 'F', 2021, 'English Literature'),  
( 'Rahul', 'Gupta', 20, 'M', 2022, 'Economics');
```

```
INSERT INTO Courses (course_name, course_code, credits, semester)  
VALUES  
( 'Introduction to Programming', 'CS101', 3, 'Fall'),  
( 'Data Structures', 'CS102', 3, 'Spring'),
```

('Calculus I', 'MATH101', 4, 'Fall'),
('Physics I', 'PHYS101', 4, 'Spring'),
('Chemistry I', 'CHEM101', 3, 'Fall'),
('Biology I', 'BIO101', 3, 'Spring'),
('Thermodynamics', 'MECH101', 4, 'Fall'),
('Linear Algebra', 'MATH102', 3, 'Spring'),
('Microeconomics', 'ECO101', 3, 'Fall'),
('Web Development', 'CS201', 4, 'Spring');

INSERT INTO Enrollments (student_id, course_id, grade, semester) VALUES

(1, 1, 'A', 'Fall'),
(1, 2, 'B', 'Spring'),
(2, 1, 'C', 'Fall'),
(2, 3, 'A', 'Spring'),
(3, 3, 'B', 'Fall'),
(3, 5, 'A', 'Spring'),
(4, 2, 'A', 'Fall'),
(4, 4, 'B', 'Spring'),
(5, 1, 'C', 'Fall'),
(5, 2, 'B', 'Spring'),
(6, 3, 'A', 'Fall'),
(6, 4, 'C', 'Spring'),
(7, 5, 'B', 'Fall'),
(7, 6, 'A', 'Spring'),
(8, 2, 'A', 'Fall'),
(8, 3, 'B', 'Spring'),
(9, 4, 'C', 'Fall'),
(9, 5, 'A', 'Spring'),
(10, 1, 'B', 'Fall'),
(10, 3, 'A', 'Spring'),
(11, 2, 'C', 'Fall'),
(11, 4, 'B', 'Spring'),
(12, 1, 'A', 'Fall'),
(12, 5, 'C', 'Spring'),
(13, 2, 'B', 'Fall'),
(13, 6, 'A', 'Spring'),
(14, 3, 'C', 'Fall'),
(14, 4, 'B', 'Spring'),
(15, 1, 'A', 'Fall'),
(15, 2, 'B', 'Spring');

INSERT INTO Attendance (student_id, course_id, date, status) VALUES

(1, 1, '2023-09-01', 'Present'),
(1, 1, '2023-09-02', 'Absent'),
(1, 2, '2023-09-01', 'Present'),
(2, 1, '2023-09-01', 'Present'),
(2, 1, '2023-09-02', 'Present'),
(2, 3, '2023-09-01', 'Absent'),
(3, 3, '2024-01-15', 'Present'),
(3, 5, '2024-01-16', 'Present'),
(4, 2, '2023-09-01', 'Present'),
(4, 4, '2023-09-02', 'Present'),
(5, 1, '2023-09-01', 'Present'),
(5, 2, '2023-09-02', 'Absent'),
(6, 3, '2023-09-01', 'Present'),
(6, 4, '2023-09-02', 'Present'),
(7, 5, '2023-09-01', 'Present'),
(7, 6, '2023-09-02', 'Present'),
(8, 2, '2023-09-01', 'Present'),
(8, 3, '2023-09-02', 'Absent'),
(9, 4, '2023-09-01', 'Present'),
(9, 5, '2023-09-02', 'Present'),
(10, 1, '2023-09-01', 'Present'),
(10, 3, '2023-09-02', 'Absent'),
(11, 2, '2023-09-01', 'Present'),
(11, 4, '2023-09-02', 'Present'),
(12, 1, '2023-09-01', 'Present'),
(12, 5, '2023-09-02', 'Absent'),
(13, 2, '2023-09-01', 'Present'),
(13, 6, '2023-09-02', 'Present'),
(14, 3, '2023-09-01', 'Absent'),
(14, 4, '2023-09-02', 'Present'),
(15, 1, '2023-09-01', 'Present'),
(15, 2, '2023-09-02', 'Present');

Code of Python for GUI and Connecting to the Database

```
import mysql.connector  
import tkinter as tk  
from tkinter import ttk, messagebox  
import matplotlib.pyplot as plt  
import pandas as pd
```

```
def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Mysql@khg10",
        database="StudentsAcademic",
    )

def fetch_data():
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM Students")
    rows = cursor.fetchall()
    conn.close()
    return rows

def insert_student(first_name, last_name, age, gender, enrollment_year, major):
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute(
        "INSERT INTO Students (first_name, last_name, age, gender,
enrollment_year, major) VALUES (%s, %s, %s, %s, %s, %s)",
        (first_name, last_name, age, gender, enrollment_year, major),
    )
    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Student added successfully!")

def update_student(
    student_id, first_name, last_name, age, gender, enrollment_year, major
):
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute(
        "UPDATE Students SET first_name=%s, last_name=%s, age=%s,
gender=%s, enrollment_year=%s, major=%s WHERE student_id=%s",
        (first_name, last_name, age, gender, enrollment_year, major, student_id),
    )
    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Student updated successfully!")
```

```
def delete_student(student_id):
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM Students WHERE student_id=%s",
(student_id,))
    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Student deleted successfully!")

def visualize_student(student_id):
    conn = connect_db()
    query = """
SELECT c.course_name, e.grade FROM Enrollments e
JOIN Courses c ON e.course_id = c.course_id
WHERE e.student_id = %s"""
    df = pd.read_sql(query, conn, params=(student_id,))
    conn.close()

    grade_mapping = {"A": 4, "B": 3, "C": 2, "D": 1, "F": 0}
    df["numeric_grade"] = df["grade"].map(grade_mapping)

    # Create a bar chart
    plt.bar(df["course_name"], df["numeric_grade"], color="blue")
    plt.xlabel("Courses")
    plt.ylabel("Grades (Numeric)")
    plt.title("Student Grades Visualization")
    plt.xticks(rotation=45)
    plt.show()

def load_students():
    for row in tree.get_children():
        tree.delete(row)
    for student in fetch_data():
        tree.insert("", "end", values=student)

root = tk.Tk()
root.title("Academic Performance Management")
tk.Label(root, text="First Name").grid(row=0, column=0)
first_name_var = tk.StringVar()
tk.Entry(root, textvariable=first_name_var).grid(row=0, column=1)
```

```
tk.Label(root, text="Last Name").grid(row=1, column=0)
last_name_var = tk.StringVar()
tk.Entry(root, textvariable=last_name_var).grid(row=1, column=1)
tk.Label(root, text="Age").grid(row=2, column=0)
age_var = tk.IntVar()
tk.Entry(root, textvariable=age_var).grid(row=2, column=1)
tk.Label(root, text="Gender").grid(row=3, column=0)
gender_var = tk.StringVar()
tk.Combobox(root, textvariable=gender_var, values=["M", "F", "Other"]).grid(
    row=3, column=1
)
tk.Label(root, text="Enrollment Year").grid(row=4, column=0)
enrollment_year_var = tk.IntVar()
tk.Entry(root, textvariable=enrollment_year_var).grid(row=4, column=1)
tk.Label(root, text="Major").grid(row=5, column=0)
major_var = tk.StringVar()
tk.Entry(root, textvariable=major_var).grid(row=5, column=1)

def add_student():
    insert_student(
        first_name_var.get(),
        last_name_var.get(),
        age_var.get(),
        gender_var.get(),
        enrollment_year_var.get(),
        major_var.get(),
    )
    load_students()

def update_student_info():
    selected_item = tree.selection()
    if selected_item:
        student_id = tree.item(selected_item, "values")[0]
        update_student(
            student_id,
            first_name_var.get(),
            last_name_var.get(),
            age_var.get(),
            gender_var.get(),
            enrollment_year_var.get(),
            major_var.get(),
```



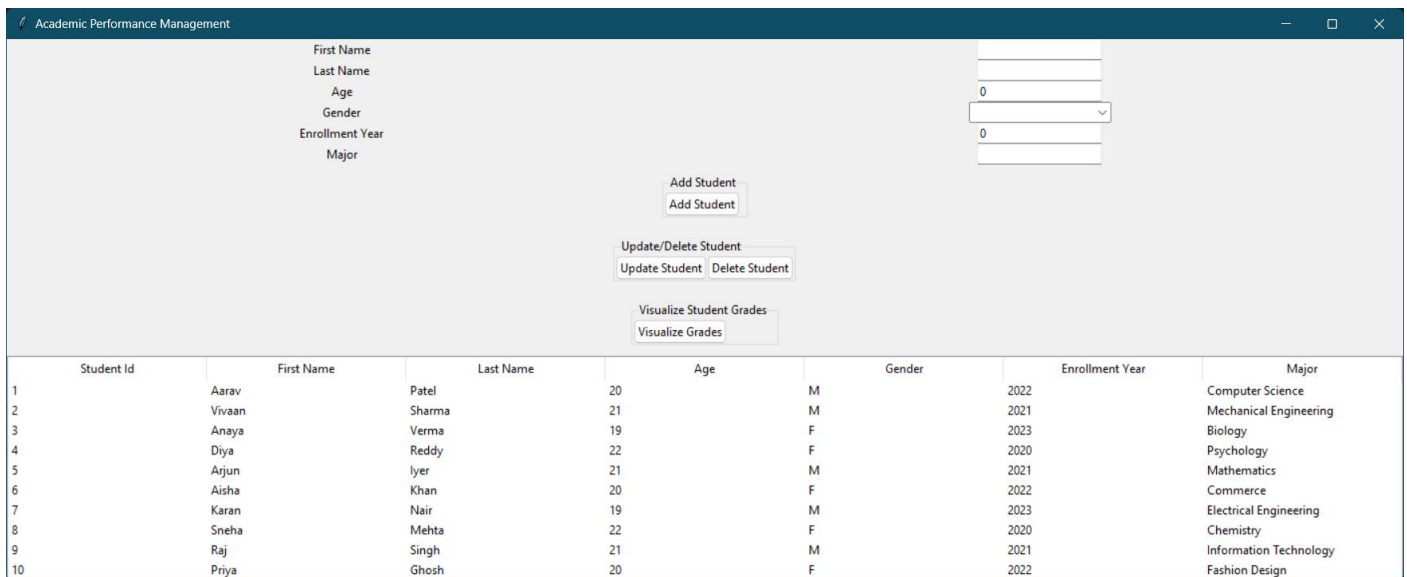
```
)  
load_students()  
  
def delete_student_info():  
    selected_item = tree.selection()  
    if selected_item:  
        student_id = tree.item(selected_item, "values")[0]  
        delete_student(student_id)  
        load_students()  
  
def visualize_data():  
    selected_item = tree.selection()  
    if selected_item:  
        student_id = tree.item(selected_item, "values")[0]  
        visualize_student(student_id)  
  
add_frame = ttk.LabelFrame(root, text="Add Student")  
add_frame.grid(row=6, column=0, columnspan=2, padx=10, pady=10)  
tk.Button(add_frame, text="Add Student", command=add_student).grid(row=0,  
column=0)  
update_delete_frame = ttk.LabelFrame(root, text="Update/Delete Student")  
update_delete_frame.grid(row=7, column=0, columnspan=2, padx=10,  
pady=10)  
tk.Button(  
    update_delete_frame, text="Update Student", command=update_student_info  
)grid(row=0, column=0)  
tk.Button(  
    update_delete_frame, text="Delete Student", command=delete_student_info  
)grid(row=0, column=1)  
visualize_frame = ttk.LabelFrame(root, text="Visualize Student Grades")  
visualize_frame.grid(row=8, column=0, columnspan=2, padx=10, pady=10)  
tk.Button(visualize_frame, text="Visualize Grades",  
command=visualize_data).grid(  
    row=0, column=0  
)  
columns = (  
    "student_id",  
    "first_name",  
    "last_name",  
    "age",  
    "gender",
```

```

"enrollment_year",
"major",
)
tree = ttk.Treeview(root, columns=columns, show="headings")
for col in columns:
    tree.heading(col, text=col.replace("_", " ").title())
tree.grid(row=9, column=0, columnspan=2)
load_students()
root.mainloop()

```

8. Output



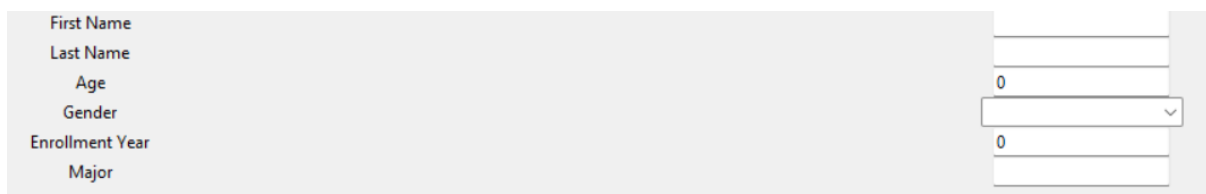
Academic Performance Management

First Name
Last Name
Age
Gender
Enrollment Year
Major

Add Student
Add Student
Update/Delete Student
Update Student Delete Student
Visualize Student Grades
Visualize Grades

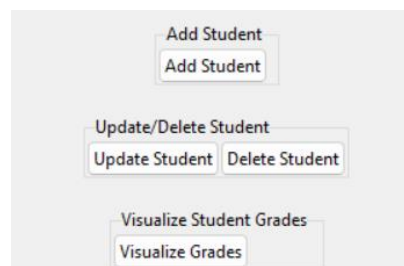
Student Id	First Name	Last Name	Age	Gender	Enrollment Year	Major
1	Aarav	Patel	20	M	2022	Computer Science
2	Vivaan	Sharma	21	M	2021	Mechanical Engineering
3	Anaya	Verma	19	F	2023	Biology
4	Diya	Reddy	22	F	2020	Psychology
5	Arjun	Iyer	21	M	2021	Mathematics
6	Aisha	Khan	20	F	2022	Commerce
7	Karan	Nair	19	M	2023	Electrical Engineering
8	Sneha	Mehra	22	F	2020	Chemistry
9	Raj	Singh	21	M	2021	Information Technology
10	Priya	Ghosh	20	F	2022	Fashion Design

Home Page of the Program



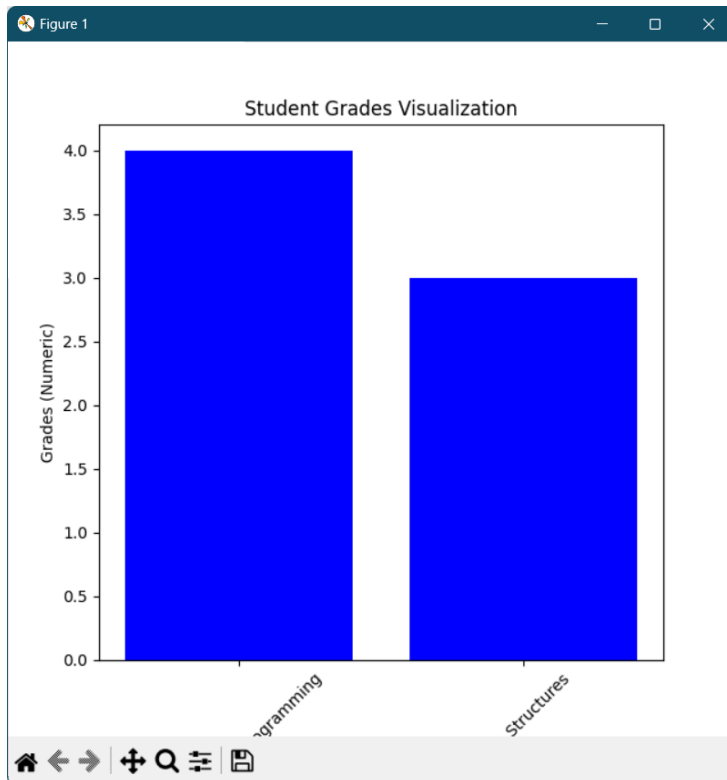
First Name
Last Name
Age
Gender
Enrollment Year
Major

Entry Boxes to enter a new record into the database



Add Student
Add Student
Update/Delete Student
Update Student Delete Student
Visualize Student Grades
Visualize Grades

Different action Buttons to carry out various operations



Grade Visualization of a Student

Tables View using MySQL

student_id	first_name	last_name	age	gender	enrollment_year	major
1	Aarav	Patel	20	M	2022	Computer Science
2	Vivaan	Sharma	21	M	2021	Mechanical Engineering
3	Anaya	Verma	19	F	2023	Biology
4	Diya	Reddy	22	F	2020	Psychology
5	Arjun	Iyer	21	M	2021	Mathematics
6	Aisha	Khan	20	F	2022	Commerce
7	Karan	Nair	19	M	2023	Electrical Engineering
8	Sneha	Mehta	22	F	2020	Chemistry
9	Raj	Singh	21	M	2021	Information Technology
10	Priya	Ghosh	20	F	2022	Fashion Design
11	Ravi	Jain	23	M	2019	Civil Engineering
12	Neha	Bhatia	19	F	2023	Physics
13	Siddharth	Choudhury	20	M	2022	Statistics
14	Isha	Malhotra	21	F	2021	English Literature
15	Rahul	Gupta	20	M	2022	Economics
NULL	NULL	NULL	NULL	NULL	NULL	NULL

All the records from Students Table

All the records from Students Table

course_id	course_name	course_code	credits	semester
1	Introduction to Programming	CS101	3	Fall
2	Data Structures	CS102	3	Spring
3	Calculus I	MATH101	4	Fall
4	Physics I	PHYS101	4	Spring
5	Chemistry I	CHEM101	3	Fall
6	Biology I	BIO101	3	Spring
7	Thermodynamics	MECH101	4	Fall
8	Linear Algebra	MATH102	3	Spring
9	Microeconomics	ECO101	3	Fall
10	Web Development	CS201	4	Spring
NULL	NULL	NULL	NULL	NULL

Various operations can be performed using SQL Queries to fetch the desired result:

SELECT

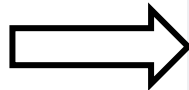
```
s.student_id,  
s.first_name,  
s.last_name,  
c.course_name,  
e.grade
```

FROM

```
Courses c
```

RIGHT JOIN

```
Enrollments e ON c.course_id = e.course_id
```



	student_id	first_name	last_name	course_name	grade
▶	1	Aarav	Patel	Introduction to Programming	A
	1	Aarav	Patel	Data Structures	B
	2	Vivaan	Sharma	Introduction to Programming	C
	2	Vivaan	Sharma	Calculus I	A
	3	Anaya	Verma	Calculus I	B
	3	Anaya	Verma	Chemistry I	A
	4	Diya	Reddy	Data Structures	A
	4	Diya	Reddy	Physics I	B
	5	Arjun	Iyer	Introduction to Programming	C
	5	Arjun	Iyer	Data Structures	B
	6	Aisha	Khan	Calculus I	A
	6	Aisha	Khan	Physics I	C
	7	Karan	Nair	Chemistry I	B
	7	Karan	Nair	Biology I	A
	8	Sneha	Mehta	Data Structures	A

SELECT

```
s.student_id,  
s.first_name,  
s.last_name,  
c.course_name,  
e.grade
```

FROM

```
Students s
```

LEFT JOIN

```
Enrollments e ON s.student_id = e.student_id
```

LEFT JOIN

```
Courses c ON e.course_id = c.course_id
```

UNION

SELECT

```
s.student_id,  
s.first_name,  
s.last_name,  
c.course_name,  
e.grade
```

FROM

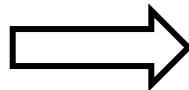
```
Courses c
```

RIGHT JOIN

```
Enrollments e ON c.course_id = e.course_id
```

RIGHT JOIN

```
Students s ON e.student_id = s.student_id;
```



	student_id	first_name	last_name	course_name	grade
▶	1	Aarav	Patel	Introduction to Programming	A
	1	Aarav	Patel	Data Structures	B
	2	Vivaan	Sharma	Introduction to Programming	C
	2	Vivaan	Sharma	Calculus I	A
	3	Anaya	Verma	Calculus I	B
	3	Anaya	Verma	Chemistry I	A
	4	Diya	Reddy	Data Structures	A
	4	Diya	Reddy	Physics I	B
	5	Arjun	Iyer	Introduction to Programming	C
	5	Arjun	Iyer	Data Structures	B
	6	Aisha	Khan	Calculus I	A
	6	Aisha	Khan	Physics I	C
	7	Karan	Nair	Chemistry I	B
	7	Karan	Nair	Biology I	A
	8	Sneha	Mehta	Data Structures	A
	8	Sneha	Mehta	Calculus I	B
	9	Raj	Singh	Physics I	C
	9	Raj	Singh	Chemistry I	A
	10	Priya	Ghosh	Introduction to Programming	B

9. Conclusion

The Academic Performance Management project successfully demonstrates how database-driven applications can be developed to efficiently manage student data and academic records. By leveraging MySQL as the database backend and a Python GUI for interaction, the project offers a user-friendly platform for administrators to perform CRUD operations, visualize academic performance, and make data-driven decisions.

Key highlights of the project include:

- **Comprehensive Data Management:** The system supports adding, updating, deleting, and retrieving student data and grades, creating a centralized database that is both organized and accessible.
- **Enhanced Data Insights with Visualization:** The visualization feature provides intuitive insights into individual student performance, helping educators and administrators quickly identify academic strengths and areas for improvement.
- **Modular and Scalable Design:** By structuring the GUI into clearly defined sections and using MySQL to store and manage data, the system is easily extensible. New features, such as additional reports or analysis modules, could be integrated with minimal adjustments.
- **Real-World Application Potential:** The project simulates a real-world academic management system and can be further extended for large-scale educational institutions with more complex requirements. It showcases how relational databases combined with GUI applications can streamline academic workflows.

Overall, this project exemplifies how database integration and data visualization can significantly enhance academic record management and provide valuable insights for educational institutions, improving the decision-making process regarding student performance and academic progress.

10. References

- **MySQL Documentation.** (n.d.). *MySQL 8.0 Reference Manual*. Retrieved from <https://dev.mysql.com/doc/refman/8.0/en/>
- **Tkinter Documentation.** (n.d.). *Tkinter: Python's Standard GUI Toolkit*. Python Software Foundation. Retrieved from <https://docs.python.org/3/library/tkinter.html>

- **Pandas Documentation.** (n.d.). *Pandas Documentation — Data Analysis Library*. Retrieved from <https://pandas.pydata.org/docs/>
- **Matplotlib Documentation.** (n.d.). *Matplotlib: Visualization with Python*. Retrieved from <https://matplotlib.org/stable/contents.html>
- *Database Design Concepts.* (2020). Tutorialspoint. Retrieved from https://www.tutorialspoint.com/dbms/dbms_database_design.htm
- **Real Python.** (2022). *Building a Simple CRUD Application with Python and MySQL*. Retrieved from <https://realpython.com/python-mysql/>
- **W3Schools.** (n.d.). *SQL Joins - SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/sql_join.asp
- **GeeksforGeeks.** (n.d.). *MySQL Joins and Types of Joins*. Retrieved from <https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-join/>
- **Stack Overflow.** (n.d.). *Handling CRUD Operations with MySQL and Tkinter*. Retrieved from various community discussions and examples on CRUD operations and data visualization techniques using MySQL and Python.
- **Kumar, N., & Verma, R.** (2021). *A Study on Academic Performance Analysis using Data Visualization Techniques*. *International Journal of Education and Learning*.