**ALL SAINT’S CHURCH SR.SEC. SCHOOL**

**M.I. ROAD , JAIPUR**

**A PROJECT REPORT ON**

**SCHOOL MANAGEMENT SYSTEM**

**SUBJECT: INFORMATICS PRACTICES (065)**

**Session: 2023-2024**

**SUBMITTED BY- SUBMITTED TO-**

**Mohd. Shayyan Mrs. Sharon Hiskiel**

**CERTIFICATE**

**This is to certify that Mohd. Shayyan of class XII SCIENCE has sucessfully completed the project on the topic School Management System, In Partial fulfilment of the requirement for the AISSCE Partal Examination of the subject code Informatics Practices(065).**

**The project work reported here is as per the guidelines of CBSE for AISSCE Practical Examination and it s done under the supervision Mrs. Sharon Hiskie, PGT COMPUTER. The project work, carried out by Mohd.Shayyan is not a form of any other project work.**

**Internal Examiner Principal**

**External Examiner School Seal**

**ACKNOWLEDGMENT**

**I would like to express my special thanks to my teacher Mrs. Sharon Hiskiel for mentoring me throughout this project work. I also thank our respected principal Mrs. Shabnam Haque for her motivation and guidance throughout the year.**

**My project is titled as “School Management System” and it has enabled me to do a lot of research and I came to Know about so many new things in software design and development.**

**Also, I would also like to thank my parents who motivated and supported me during my work.**

**Mohd. Shayyan**

**XII SCIENCE**

**INDEX**

1. **Python Introduction**
2. **MySQL Introduction**
3. **Hardware Requirements**
4. **Introduction to project** **(python)**
5. **Database schema Screenshots** **(SQL)**
6. **User Output**
7. **SQL Queries**
8. **User Interface Code**
9. **Testing**
10. **Bibliography**

**PYTHON INTRODUCTION**

**Python is a general purpose, dynamic, high-level, and interpreted programming language. Python is a high level language. It is a free and open source language. It is an interpreted language, as python programs are executed by an interpreter. Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.**

**• Pandas is a Python library used for working with data sets.**

**• It has functions for analyzing, cleaning, exploring, and manipulating data.**

**• The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.**

**• Pandas can clean messy data sets, and make them readable and relevant.**

**• Relevant data is very important in data science.**

**• Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.**

**MySQL INTRODUCTION**

**The “School Management System” created by me is based on PYTHON AND MYSQL.**

**Its an automation of the existing system which enables its user to perform few operations pertaining to management of School as listed below.**

**The Project Enables its user to:**

**1.) Add new Student, new Staff and new Fee records.**

**2.) Delete Student, Staff and Fee records.**

**3.) Update Student, Staff and Fee records.**

**4.) View Student, Staff and Fee records from the Database.**

**System Requirements**

**-------------------|Hardware Requirements |--------------**

**Processor- Intel(R) Core(TM) i5**

**7300U CPU 2.60GHz**

**2.71 GHz**

**Installed memory[RAM]- 8.00 GB (7.88 GB usable)**

**System Type- 64-bit operating system,**

**X64-based processor**

**Pen and Touch- No pen or touch input is**

**Available for this display**

**INTRODUCTION OF PROJECT**

**PROJECT TITLE-“SCHOOL MANAGEMENT”**

**DBMS: MySQL**

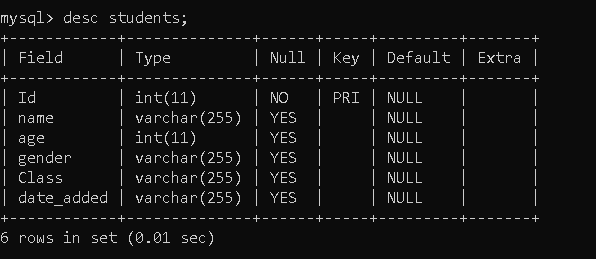
**Host : localhost**

**User: root Password: root Database: School**

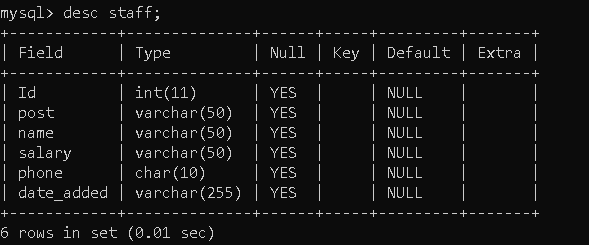
**Table Structure: As per the Screenshot given below:**

**Screenshots OF PROJECT**

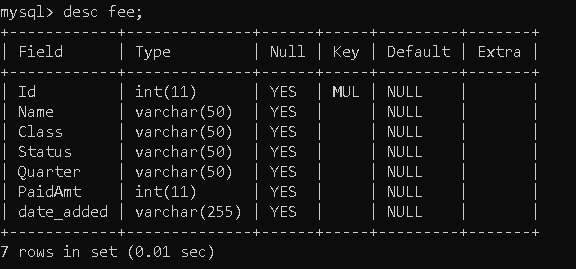
**Student table has following Schema**

****

**Staff table has following Schema**

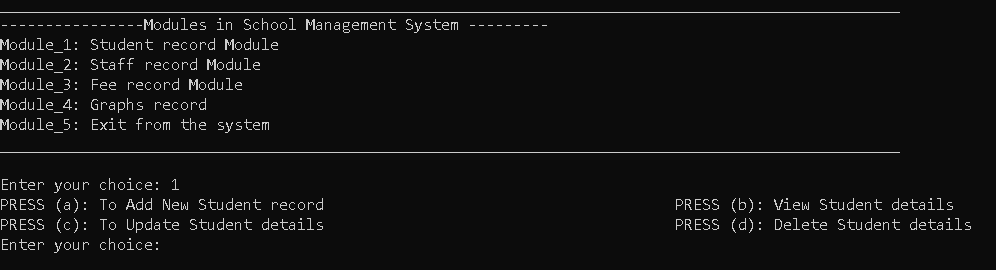
****

**Fee table has following Schema**

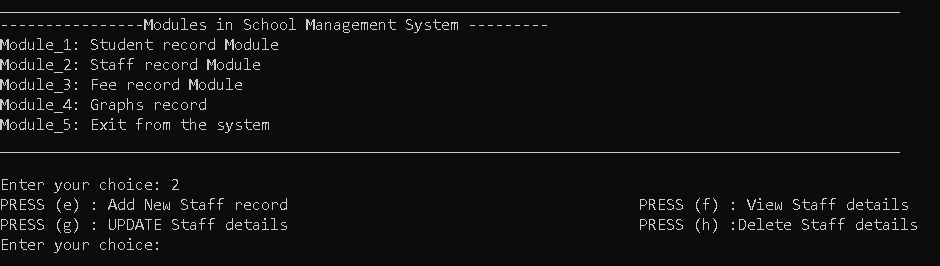
****

**USER OUTPUT**

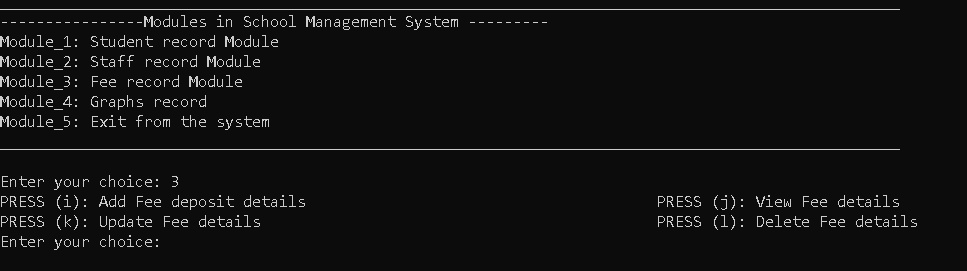
**STUDENT MODULE DETAILS:**

****

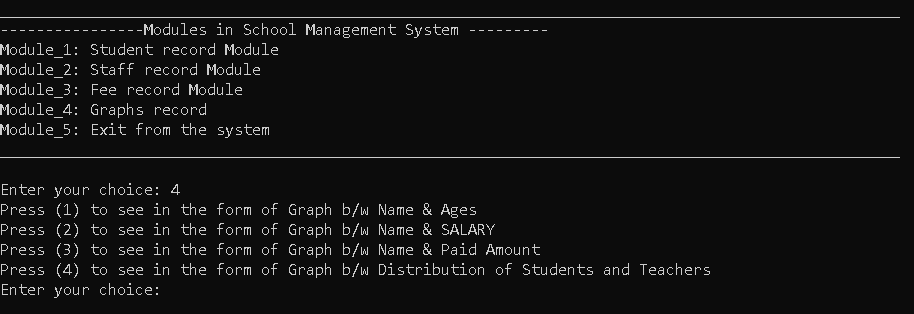
**STAFF MODULE DETAILS:**

****

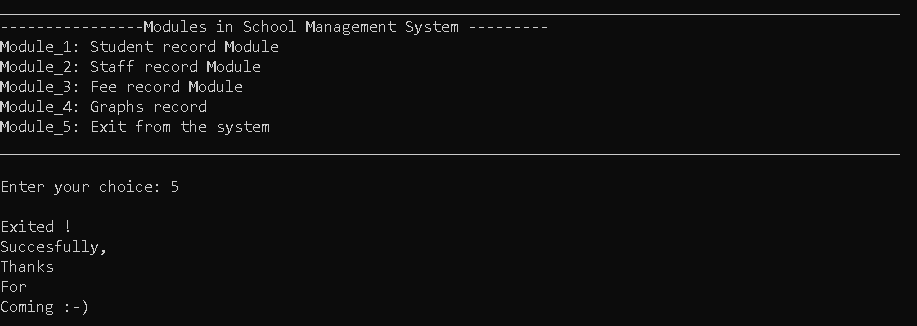
**FEE MODULE DETAILS:**

****

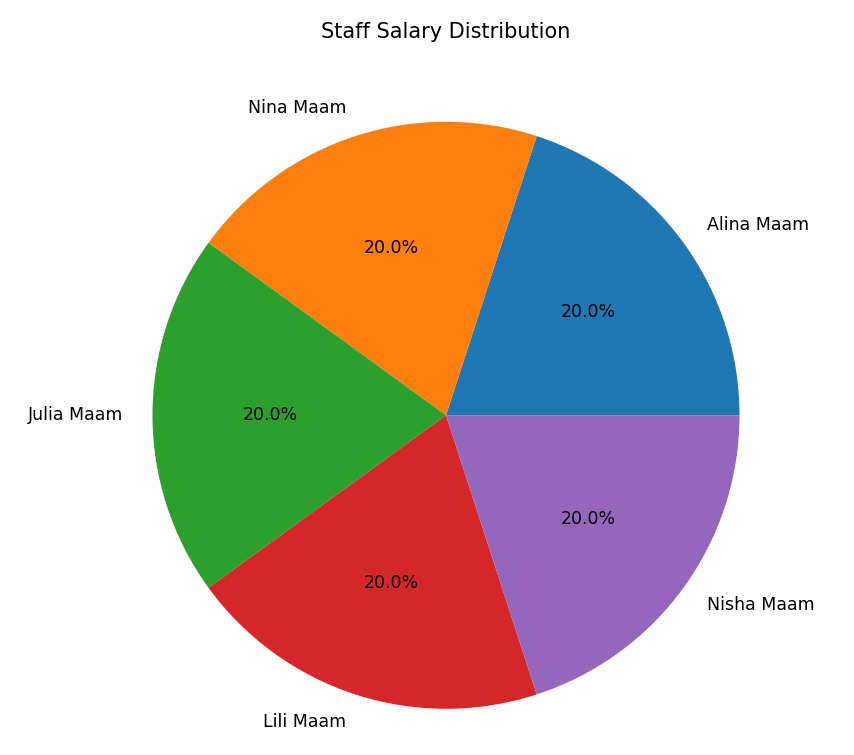
**EXIT MODULE DETAILS:**

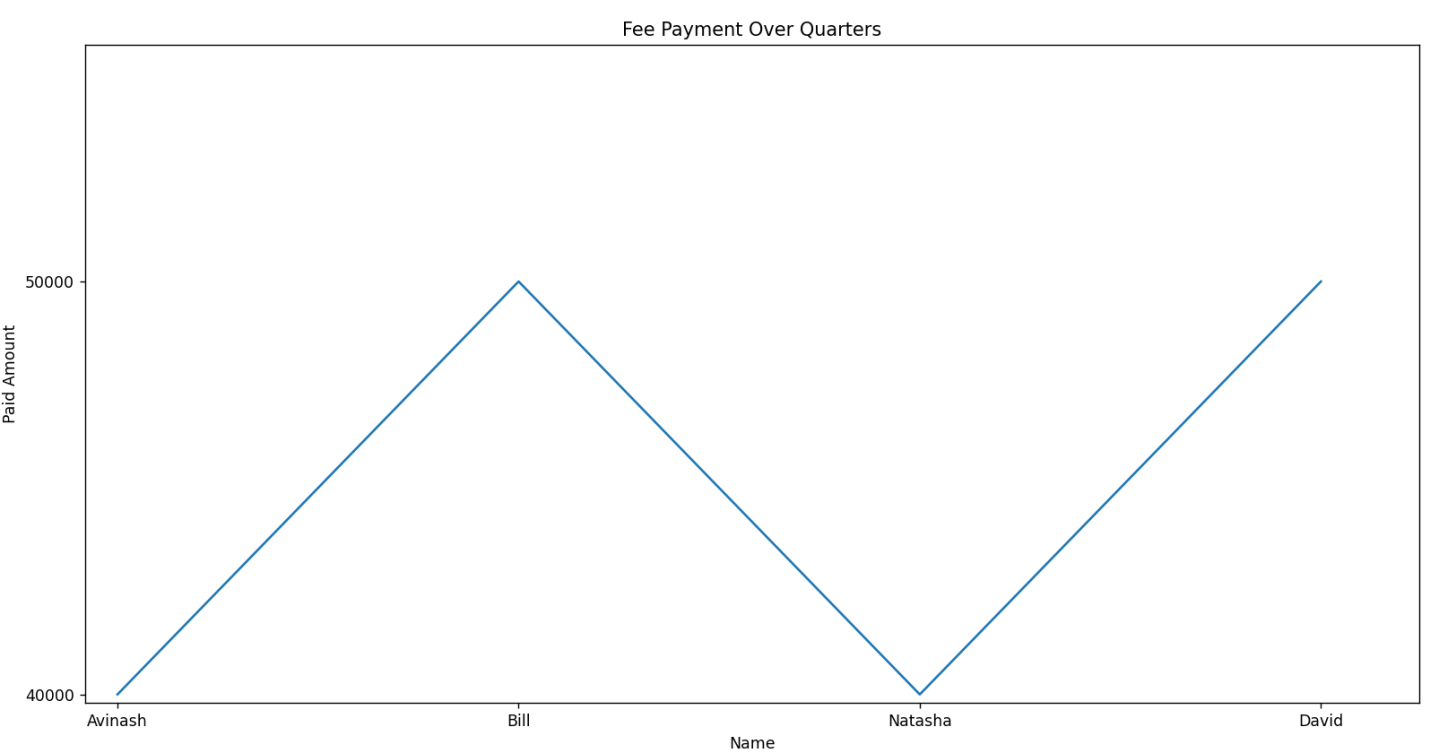
****

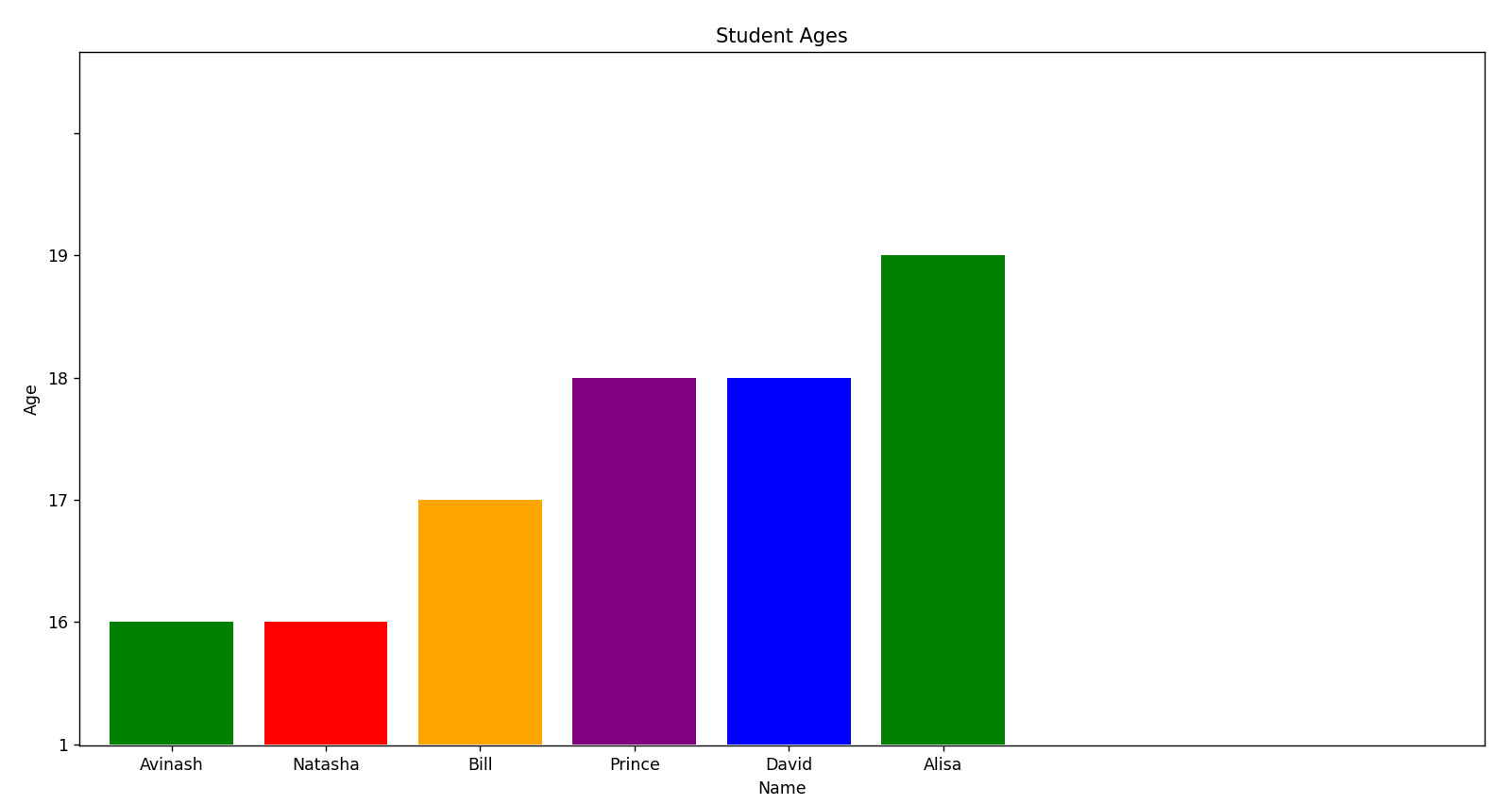
**EXIT MODULE DETAILS:**

****

**CHARTS**

**Chart b/w Staff Name & Salary: **

**Chart :Students vs Amount Paid: **

**Chart: Students vs Ages: **

**SQL QUERIES**

**Create database school;**

**use school;**

**CREATE TABLE IF NOT EXISTS students (Id INT PRIMARY KEY,name VARCHAR(255), age INT, gender VARCHAR(255), Class VARCHAR(255),date\_added VARCHAR(255)**

**Desc students;**

**use school;**

**CREATE TABLE IF NOT EXISTS Staff(Id INT,post varchar(50),name varchar(50),salary varchar(50),phone char(10),date\_added VARCHAR(255)**

**Desc Staff;**

**CREATE TABLE IF NOT EXISTS fee(Id INT,Name varchar(50),Class varchar(50),Status varchar(50),Quarter varchar(50),PaidAmt INT,date\_added VARCHAR(255),FOREIGN KEY (Id) REFERENCES students(Id)**

**Desc fee;**

**USER OUTPUT(SOURCE CODE)**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user='root',**

**password='root')**

**print(mydb,"connected to server")**

**print("\n")**

**print("-" \*100)**

**print(" Welcome to School Management System")**

**def menu():**

**print("-" \* 100)**

**print("----------------Modules in School Management System ---------")**

**print("Module\_1: Student record Module ")**

**print("Module\_2: Staff record Module")**

**print("Module\_3: Fee record Module")**

**print("Module\_4: Exit from the system")**

**print("\_" \* 100)**

**# Get the user's choice:**

**# if option first:**

**def getchoice():**

**while True:**

**create\_database()**

**create\_students()**

**create\_Staff()**

**create\_fee()**

**menu()**

**ch = input("Enter your choice: ")**

**if ch=='1':**

**print("PRESS (a): To Add New Student record PRESS (b): View Student details")**

**print("PRESS (c): To Update Student details PRESS (d):Delete Student details")**

**ch = input("Enter your choice: ")**

**create\_students()**

**if ch=='a':**

**add\_student()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif ch=='b':**

**view\_students()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif ch=='c':**

**update\_student()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif ch=='d':**

**delete\_student()**

**input("Press ENTER KEY to continue.....")**

**print()**

**## if option Second:**

**elif ch=='2':**

**print("PRESS (e) : Add New Staff record PRESS (f) : View Staff details | ")**

**print("PRESS (g) : Delete Staff details PRESS (h) : UPDATE Staff details ")**

**opp =input("Enter your choice: ")**

**create\_Staff()**

**if opp=='e':**

**add\_staff()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='f':**

**view\_staff()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='g':**

**update\_staff()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='h':**

**delete\_staff()**

**input("Press ENTER KEY to continue.....")**

**print()**

**### if option Third:**

**elif ch=='3':**

**print("PRESS (i): Add Fee deposit details PRESS (j): View Fee details ")**

**print("PRESS (k): Update Fee details PRESS (l): Delete Fee details")**

**opp = input("Enter your choice: ")**

**create\_fee()**

**if opp=='i':**

**fee()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='j':**

**view\_fee()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='k':**

**update\_fee()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp=='l':**

**delete\_fee()**

**input("Press ENTER KEY to continue.....")**

**print()**

**#### if option Fourth:**

**elif ch=='4':**

**print("Press (1) to see in the form of Graph b/w Name & Ages ")**

**print("Press (2) to see in the form of Graph b/w Name & SALARY ")**

**print("Press (3) to see in the form of Graph b/w Name & Paid Amount ")**

**print("Press (4) to see in the form of Graph b/w Distribution of Students and Teachers ")**

**ch = input("Enter your choice: ")# Get the user's choice**

**if ch == '1':**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM students")**

**result = cursor.fetchall()**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**df = pd.DataFrame({'ID': lst1, 'Name': lst2, 'Age': lst3, 'Gender': lst4, 'Class': lst5})**

**# Sort the dataframe by Age in ascending order**

**df\_sorted = df.sort\_values(by='Age')**

**# Get the sorted values for 'Name' and 'Age'**

**Name = df\_sorted['Name'].tolist()**

**Age = df\_sorted['Age'].tolist()**

**# Create the bar chart**

**plt.bar(Name, Age, color=['blue', 'green', 'red', 'orange', 'purple'])**

**plt.xlabel('Name')**

**plt.ylabel('Age')**

**plt.title('Student Ages')**

**# Set the y-axis limits and ticks**

**plt.ylim(0, 18) # Set the y-axis limits from 0 to 18**

**plt.yticks(range(19)) # Set the y-axis ticks from 0 to 18**

**plt.show()**

**# Plotting pie chart**

**elif ch == '2':**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM staff")**

**result = cursor.fetchall()**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**df=pd.DataFrame({'ID':lst1,'POST':lst2,'NAME':lst3,'SALARY':lst4,'PHONE':lst5})**

**plt.pie(df['SALARY'], labels=df['NAME'], autopct='%1.1f%%')**

**plt.title('Staff Salary Distribution')**

**plt.show()**

**#Plotting line chart**

**elif ch == '3':**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM fee")**

**result = cursor.fetchall()**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**lst6 = [row[5] for row in result\_list]**

**df=pd.DataFrame({'Id':lst1,'Name':lst2,'Class':lst3,'Status':lst4,'Quarter':lst5,'PaidAmt':lst6 })**

**# Sort the DataFrame by Quarter in ascending order**

**df.sort\_values(by='PaidAmt')**

**Name = df['Name']**

**PaidAmt = df['PaidAmt']**

**plt.plot(Name, PaidAmt)**

**plt.xlabel('Name')**

**plt.ylabel('Paid Amount')**

**plt.title('Fee Payment Over Quarters')**

**plt.show()**

**# Plotting pie chart No.of Students & Teachers:**

**elif ch == '4':**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM students")**

**result = cursor.fetchall()**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**df = pd.DataFrame({'ID': lst1, 'Name': lst2, 'Age': lst3, 'Gender': lst4, 'Class': lst5})**

**cursor.execute("SELECT \* FROM staff")**

**result = cursor.fetchall()**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**df=pd.DataFrame({'ID':lst1,'POST':lst2,'NAME':lst3,'SALARY':lst4,'PHONE':lst5})**

**# Count the number of students**

**cursor.execute("SELECT COUNT(\*) FROM students")**

**num\_students = cursor.fetchone()[0]**

**# Count the number of staff (teachers)**

**cursor.execute("SELECT COUNT(\*) FROM staff")**

**num\_teachers = cursor.fetchone()[0]**

**# Create a DataFrame for the data**

**data = {'Category': ['Students', 'Teachers'], 'Count': [num\_students, num\_teachers]}**

**df = pd.DataFrame(data)**

**# Plotting the pie chart**

**plt.figure(figsize=(6, 6))**

**plt.pie(df['Count'], labels=df['Category'], autopct='%1.1f%%', startangle=140)**

**plt.title('Distribution of Students and Teachers')**

**plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.**

**# Show the chart**

**plt.show()**

**#### if option fiveth:**

**elif ch=='5':**

**print()**

**print("Exited !")**

**print("Succesfully,")**

**print("Thanks")**

**print("For")**

**print("Coming :-)")**

**print()**

**print()**

**print()**

**print()**

**ADD STUDENT RECORD**

**# Creating the table if it doesn't exist**

**def create\_students():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS students (Id INT PRIMARY KEY,name VARCHAR(255), age INT, gender VARCHAR(255), Class VARCHAR(255),date\_added VARCHAR(255))')**

**def add\_student():**

**Id = input("Enter Id of student: ")**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM students WHERE Id = %s", (Id,))**

**existing\_students = cursor.fetchone()**

**if existing\_students:**

**print("student with this Id already exists. Please enter a different Id.")**

**else:**

**name = input("Enter student Name: ")**

**age = input("Enter student's age: ")**

**gender = input("Enter student gender(m/f): ")**

**Class = input("Enter student Class: ")**

**now = datetime.now()**

**date\_time = now.strftime("%Y-%m-%d %H:%M:%S")**

**# Inserting Values**

**sql = "INSERT INTO students (Id, name, age, gender, Class, date\_added) VALUES (%s, %s, %s , %s, %s, %s)"**

**val = (Id, name, age, gender, Class, date\_time)**

**cursor.execute(sql, val)# Executing the SQL query**

**mydb.commit()# Committing the changes in the table**

**print(cursor.rowcount, "record(s) inserted.")**

**DELETE STUDENT RECORD**

**# Define the function to delete student details**

**def delete\_student():**

**Id = input("Enter student Id: ")**

**cursor = mydb.cursor()**

**sql = "delete from students where Id = %s"**

**val = (Id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**VIEW STUDENT RECORD**

**# Define the function to view student details**

**def view\_students():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM students")**

**result = cursor.fetchall()**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (i) to see the Separate Index values")**

**print("Press (l) to see in the form of list")**

**ch = input("Enter your choice: ")**

**if ch=='i':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**print("Id is:", lst1)**

**lst2 = [row[1] for row in result\_list]**

**print('Name is:', lst2)**

**lst3 = [row[2] for row in result\_list]**

**print('Age is', lst3)**

**lst4 = [row[3] for row in result\_list]**

**print('Gender is: ', lst4)**

**lst5 = [row[4] for row in result\_list]**

**print('Class is', lst5)**

**lst6 = [row[5] for row in result\_list]**

**print('Date\_&\_Time', lst6)**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**lst6 = [row[5] for row in result\_list]**

**df = pd.DataFrame({'Id': lst1, 'Name': lst2, 'Age': lst3, 'Gender': lst4, 'Class': lst5,'Date\_&\_Time': lst6})**

**print(df.to\_markdown())**

**UPDATE STUDENT RECORD**

**# Define the function to update student details**

**def update\_student():**

**Id = input("Enter student's Id: ")**

**name = input("Enter student's Name: ")**

**age = input("Enter studenlt's age: ")**

**gender = input("Enter student's gender(m/f): ")**

**Class = input("Enter student's Class: ")**

**cursor = mydb.cursor()**

**sql\_up = "update students set name = %s, age = %s, gender = %s, Class = %s where Id = %s"**

**val\_up = (name, age, gender, Class,Id)**

**cursor.execute(sql\_up, val\_up)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**ADD STAFF RECORD**

**# CREATING A TABLE**

**def create\_Staff():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Staff(Id INT,post varchar(50),name varchar(50),salary varchar(50),phone char(10),date\_added VARCHAR(255)** **,FOREIGN KEY (Id) REFERENCES students(Id))')**

**# Define the function to add a new staff**

**def add\_staff():**

**Id = input("Enter staff ID: ")**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM staff WHERE Id = %s", (Id,))**

**existing\_staff = cursor.fetchone()**

**if existing\_staff:**

**print("Staff with this ID already exists. Please enter a different ID.")**

**else:**

**post = input("Enter staff Post: ")**

**name = input("Enter staff Name: ")**

**salary = input("Enter staff Salary: ")**

**phone = input("Enter staff Phone no: ")**

**now = datetime.now()**

**date\_time = now.strftime("%Y-%m-%d %H:%M:%S")**

**# Inserting Values**

**sql = "INSERT INTO staff (Id, post, name, salary, phone, date\_added) VALUES (%s, %s, %s, %s, %s, %s)"**

**val = (Id, post, name, salary, phone, date\_time)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**UPDATE STAFF RECORD**

**# Define the function to update staff details**

**def update\_staff():**

**Id=input("Enter staff ID: ")**

**post=input("Enter staff Post: ")**

**name = input("Enter staff Name: ")**

**salary = input("Enter staff Salary: ")**

**phone = input("Enter staff Phone no: ")**

**cursor = mydb.cursor()**

**- sql = "UPDATE staff set Id = %s , post= %s, name = %s, salary = %s, phone = %s WHERE Id = %s"**

**val = (Id,post,name,salary, phone)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**DELETE STAFF RECORD**

**# Define the function to delete staff details**

**def delete\_staff():**

**Id = input("Enter staff ID: ")**

**cursor = mydb.cursor()**

**sql = "DELETE FROM staff WHERE Id = %s"**

**val = (Id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**VIEW STAFF RECORD**

**# Define the function to view student details**

**def view\_staff():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM staff")**

**result = cursor.fetchall()**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (i) to see the Separate Index values")**

**print("Press (l) to see in the form of list")**

**ch = input("Enter your choice: ")# Get the user's choice**

**if ch=='i':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**print("Id is:", lst1)**

**lst2 = [row[1] for row in result\_list]**

**print('Post is:', lst2)**

**lst3 = [row[2] for row in result\_list]**

**print('Name is:', lst3)**

**lst4 = [row[3] for row in result\_list]**

**print('Salary is: ', lst4)**

**lst5 = [row[4] for row in result\_list]**

**print('Phone\_no is: ', lst5)**

**lst6 = [row[5] for row in result\_list]**

**print('Date\_&\_Time is: ', lst6)**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**lst6 = [row[5] for row in result\_list]**

**df=pd.DataFrame({'ID':lst1,'POST':lst2,'NAME':lst3,'SALARY':lst4,'PHONE':lst5,'Date\_&\_Time': lst6})**

**print(df.to\_markdown())**

**ADD FEE RECORD**

**# CREATING A TABLE**

**def create\_fee():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS fee(Id INT,Name varchar(50),Class varchar(50),Status varchar(50),Quarter varchar(50),PaidAmt INT,date\_added VARCHAR(255),FOREIGN KEY (Id) REFERENCES students(Id))')**

**# Define the function to add Fee details**

**def fee():**

**Id = input("Enter Payer's ID: ")**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM fee WHERE Id = %s", (Id,))**

**existing\_fee = cursor.fetchone()**

**if existing\_fee:**

**print("fee with this ID already exists. Please enter a different Id.")**

**else:**

**Name = input("Enter Payer's Name: ")**

**Class = input("Enter Payer's Class: ")**

**Status = input("Enter Status (Paid/Due): ")**

**Quarter = input("Enter Quarter: ")**

**PaidAmt = input("Enter Paid Amount: ")**

**now = datetime.now()**

**date\_time = now.strftime("%Y-%m-%d %H:%M:%S")**

**# Inserting Values**

**sql = "INSERT INTO fee (Id, Name, Class, Status, Quarter, PaidAmt, date\_added) VALUES (%s, %s, %s, %s, %s, %s, %s)"**

**val = (Id, Name, Class, Status, Quarter, PaidAmt, date\_time)**

**cursor.execute(sql, val)**

**mydb.commit()# Committing the changes in the table**

**print(cursor.rowcount, "record(s) inserted.")**

**VIEW FEE RECORD**

**# Define the function to view Fee details**

**def view\_fee():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM fee")**

**result = cursor.fetchall()**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (i) to see the Separate Index values")**

**print("Press (l) to see in the form of list")**

**# Get the user's choice**

**ch = input("Enter your choice: ")**

**if ch=='i':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**print("Id is:", lst1)**

**lst2 = [row[1] for row in result\_list]**

**print('Name is:', lst2)**

**lst3 = [row[2] for row in result\_list]**

**print('Class is:', lst3)**

**lst4 = [row[3] for row in result\_list]**

**print('Status is: ', lst4)**

**lst5 = [row[4] for row in result\_list]**

**print('Quarter is:', lst5)**

**lst6 = [row[5] for row in result\_list]**

**print('PaidAmt is:', lst6)**

**lst7 = [row[6] for row in result\_list]**

**print('Date\_&\_Time is:', lst7)**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**lst5 = [row[4] for row in result\_list]**

**lst6 = [row[5] for row in result\_list]**

**lst7 = [row[6] for row in result\_list]**

**df=pd.DataFrame({'Id':lst1,'Name':lst2,'Class':lst3,'Status':lst4,'Quarter':lst5,'PaidAmt':lst6,'Date\_&\_Time': lst7 })**

**print(df.to\_markdown())**

**DELETE FEE RECORD**

**# Define the function to delete Fee details**

**def delete\_fee():**

**Id = input("Enter student Id: ")**

**cursor = mydb.cursor()**

**sqle = "DELETE FROM fee WHERE Id = %s"**

**vale = (Id,)**

**cursor.execute(sqle, vale)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**UPDATE FEE RECORD**

**# Define the function to update Fee details**

**def update\_fee():**

**Id = input("Enter student Id: ")**

**Name = input("Enter student Name: ")**

**Class = input("Enter student Class: ")**

**Status = input("Enter student Status(Paid/Due): ")**

**Quarter = input("Enter student Quarter: ")**

**PaidAmt = input("Enter student PaidAmount: ")**

**cursor = mydb.cursor()**

**sqlx = "UPDATE fee SET Name = %s, Class = %s, Status = %s, Quarter = %s,PaidAmt = %s WHERE Id = %s"**

**valx = (Name,Class,Status,Quarter,PaidAmt,Id)**

**cursor.execute(sqlx, valx)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**CONCLUSION**

**Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. Software testing also provides an, independent view of the software to allow the business to appreciate and understand the risk at implementation of the software.**

**Test techniques include, but are not limited to the process of executing a program or an application with the intent of finding software bugs.**

**It can also be stated as the process of validating and verifying that**

**a software program/Application / product meets the business**

**and technical requirements that guided the its design and**

**development, so that it works as expected and can be**

**implemented with the same characteristics. Software**

**testing, depending on the testing method employed, can**

**be implemented at any time in the development process however**

**the most test effort is employed after the requirements have**

**been defined and coding process has been completed.**

**Future Scope of Project**

**Some suggestions for the “Future Scope of Project”:**

**1.Integration with Online Platforms:Explore the possibility of integration the School Management System wth online platforms for a more seamless experience.This could include student and parent portals, online fee payment gateways and communication platforms.**

**2.** **Enhanced Security Measures:** **Implement advanced security features to protect sensitive student and staff information. This may include two-factor authentication, encrypted databases, and regular security audits.**

**3.** **Parent-Teacher Communication Platform:** **Develop a dedicated platform for effective communication between parents and teachers.**

**This could include features such as messaging, progress reports and scheduling parent-teacher meetings.**

**4. Customization and Scalability:** **Ensure that the system is easily customizable to meet the unique needs of different schools. Make it scalable so that it can accommodate the growth of data and users**

**Over time.**

**5. User Training and Support:** **Provide comprehensive user training modules and support systems for administrators,** **teachers, and parents to ensure smooth adoption and effective use of the School**

**Management System.**

**BIBLOGRAPHY**

* **GOOGLE**
* **www.wikipedia.com**
* **www.geeksforgeeks.org**
* **NCERT**
* **KIPS**
* **SUMITA ARORA**
* **PREETI ARORA**