Student Management System

By Gayatri Patil

28/10/2024

Introduction

Purpose

1. Simplifies and enhances the management of student records.
2. Reduces manual efforts.
3. Increases efficiency.
4. Provides a centralized platform for managing student data.

Importance

1. Enables quick access to student information.
2. Ensures data consistency and accuracy.
3. Streamlines administrative tasks, allowing staff to focus on higher priorities.
4. Supports informed decision-making with readily available data.

Technologies Used

Database

- **MySQL** for persistent storage of user and student data.

Programming Language

- **Python** for application logic, interaction with MySQL, and handling authentication.

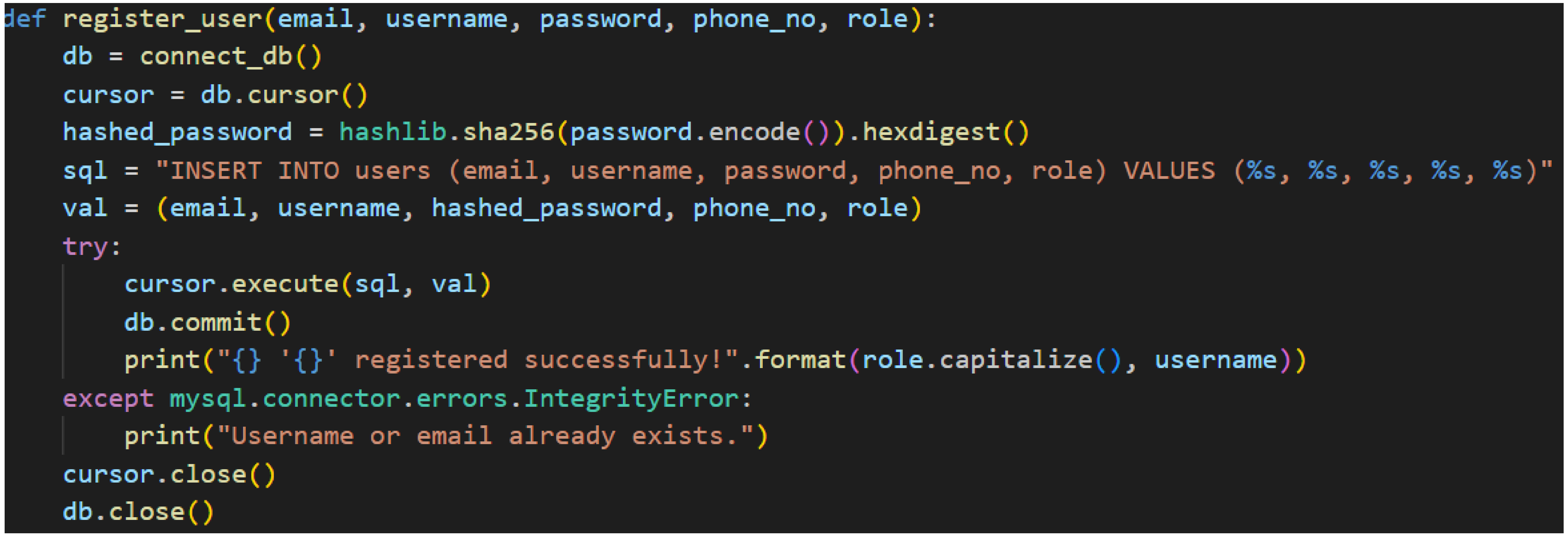
Libraries

- **MySQL Connector** to connect and execute queries in the MySQL database.
- **Hashlib** for hashing passwords securely before storage.
- **CSV** for exporting student data to a CSV file format.

Implementation and Results

1. User Registration

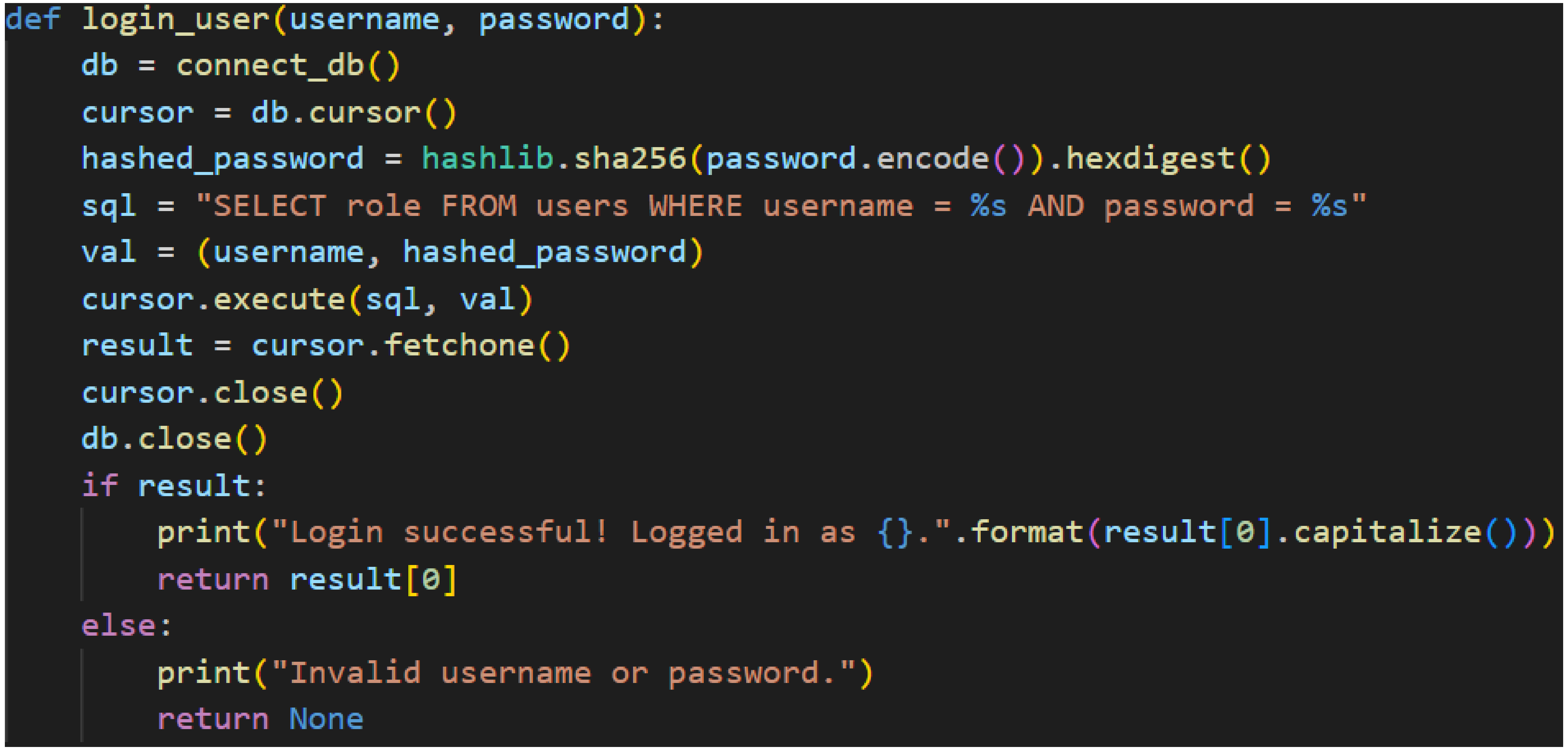
Register a new user and hash the password and use exception handling to prevent duplicate usernames or emails.



```
def register_user(email, username, password, phone_no, role):
    db = connect_db()
    cursor = db.cursor()
    hashed_password = hashlib.sha256(password.encode()).hexdigest()
    sql = "INSERT INTO users (email, username, password, phone_no, role) VALUES (%s, %s, %s, %s, %s)"
    val = (email, username, hashed_password, phone_no, role)
    try:
        cursor.execute(sql, val)
        db.commit()
        print("{} '{}' registered successfully!".format(role.capitalize(), username))
    except mysql.connector.errors.IntegrityError:
        print("Username or email already exists.")
    cursor.close()
    db.close()
```

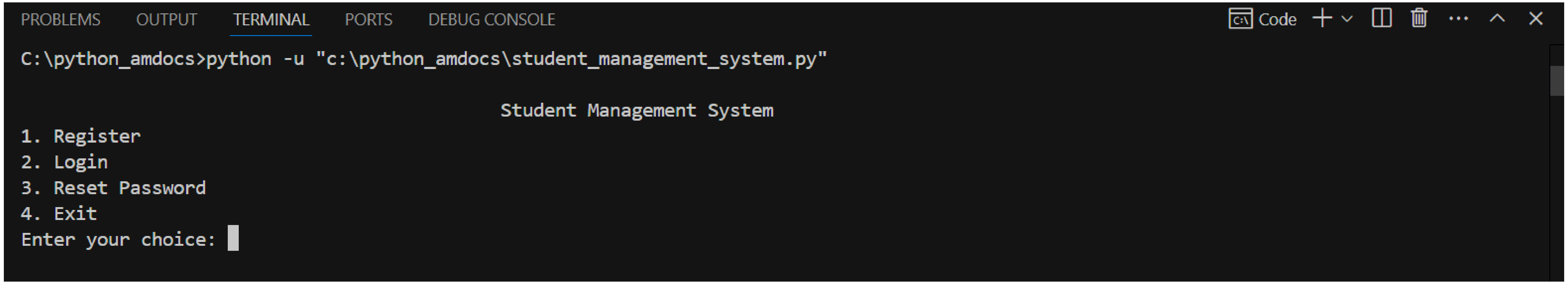
2. User Login

While login passwords are hashed and then matched with stored hashed values, ensuring data security. login_user function returns the role, enabling role-based functionality.

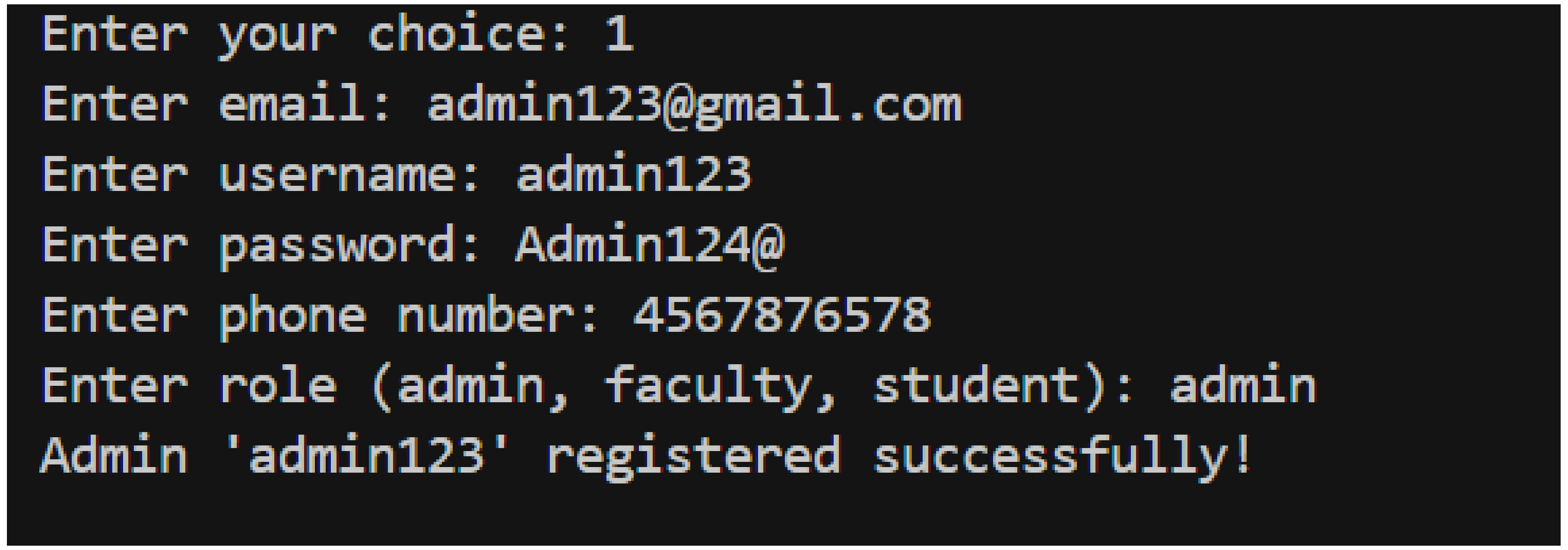


```python
def login_user(username, password):
    db = connect_db()
    cursor = db.cursor()
    hashed_password = hashlib.sha256(password.encode()).hexdigest()
    sql = "SELECT role FROM users WHERE username = %s AND password = %s"
    val = (username, hashed_password)
    cursor.execute(sql, val)
    result = cursor.fetchone()
    cursor.close()
    db.close()
    if result:
        print("Login successful! Logged in as {}.".format(result[0].capitalize()))
        return result[0]
    else:
        print("Invalid username or password.")
        return None
```

Registration and Login output

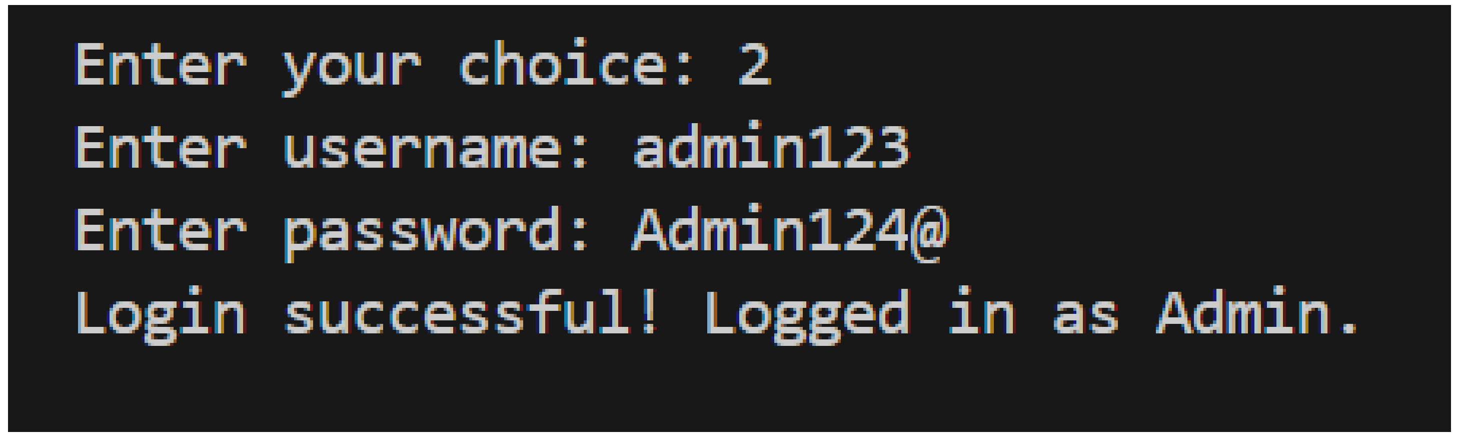


```
C:\python_amdocs>python -u "c:\python_amdocs\student_management_system.py"

                                            Student Management System
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 
```



```
Enter your choice: 1
Enter email: admin123@gmail.com
Enter username: admin123
Enter password: Admin124@
Enter phone number: 4567876578
Enter role (admin, faculty, student): admin
Admin 'admin123' registered successfully!
```

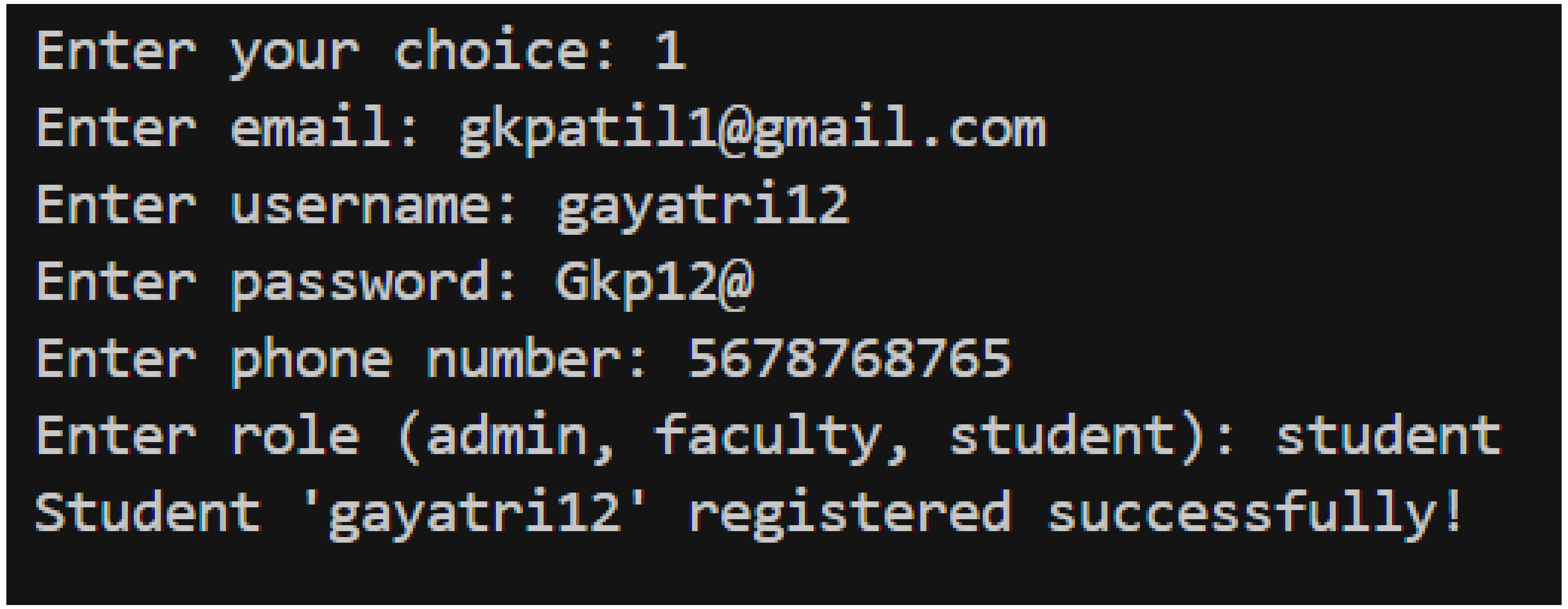
Admin Registration



```
Enter your choice: 2
Enter username: admin123
Enter password: Admin124@
Login successful! Logged in as Admin.
```

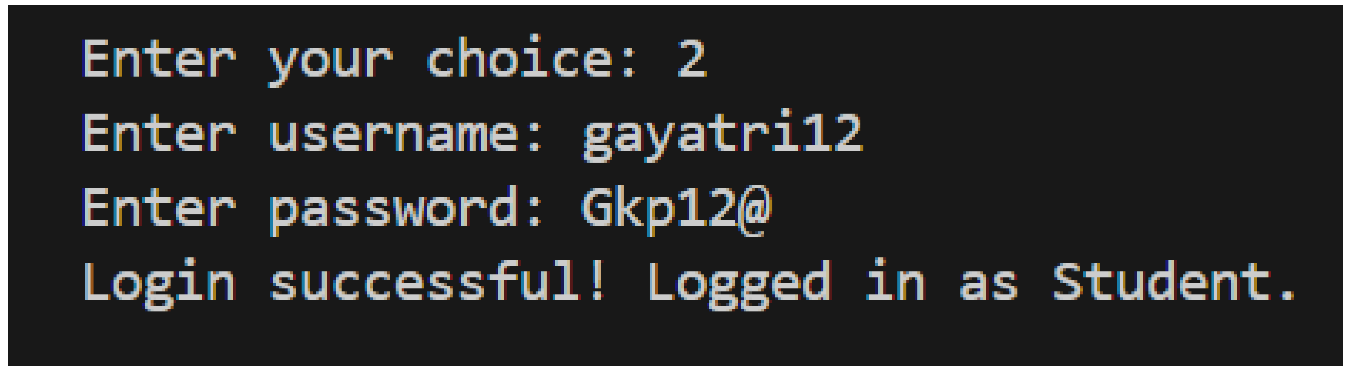
Admin Login

Registration and Login output



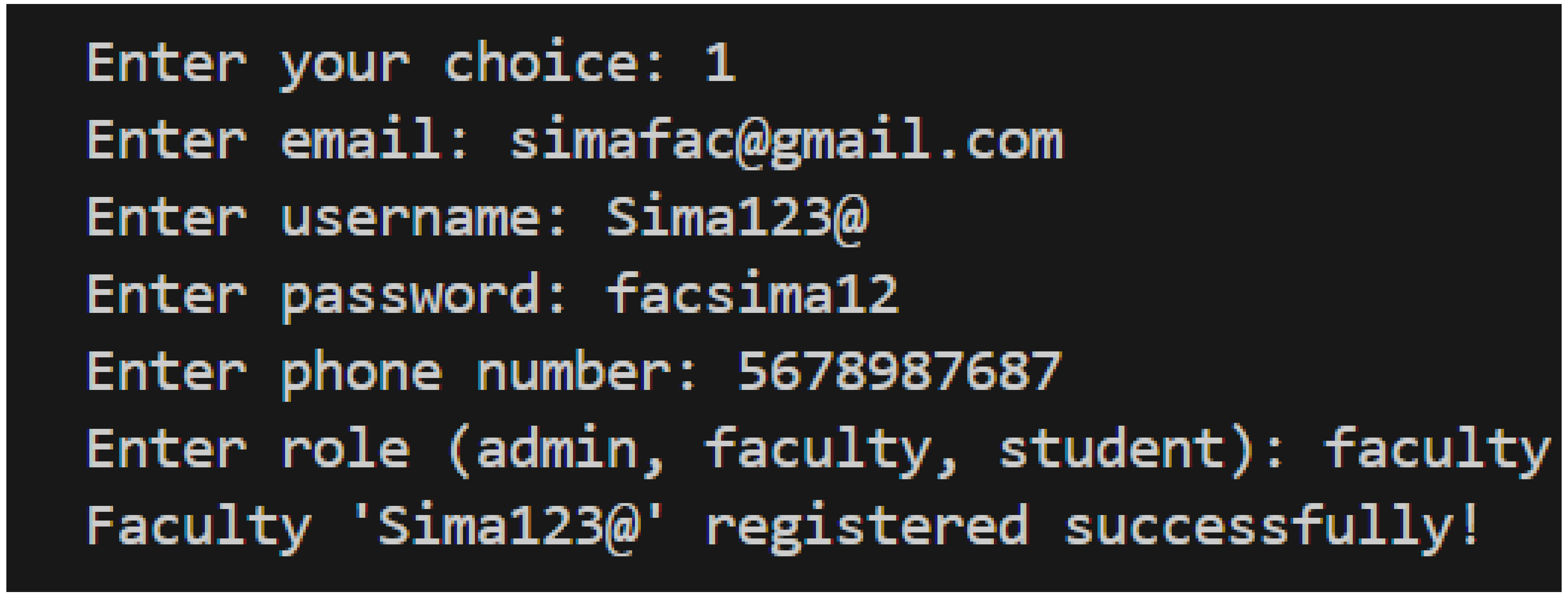
```
Enter your choice: 1
Enter email: gkpatil1@gmail.com
Enter username: gayatri12
Enter password: Gkp12@
Enter phone number: 5678768765
Enter role (admin, faculty, student): student
Student 'gayatri12' registered successfully!
```

Student Registration



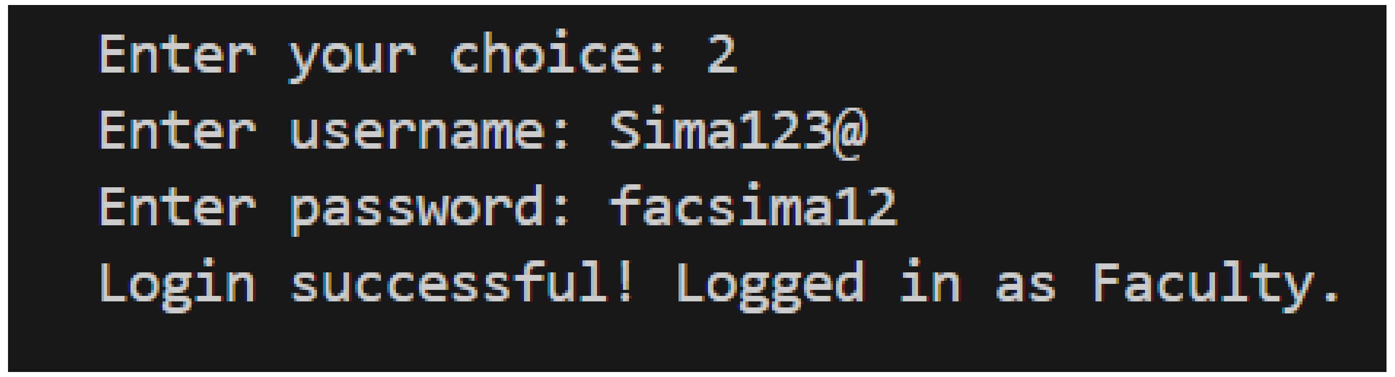
```
Enter your choice: 2
Enter username: gayatri12
Enter password: Gkp12@
Login successful! Logged in as Student.
```

Student Login



```
Enter your choice: 1
Enter email: simafac@gmail.com
Enter username: Sima123@
Enter password: facsima12
Enter phone number: 5678987687
Enter role (admin, faculty, student): faculty
Faculty 'Sima123@' registered successfully!
```

Faculty Registration

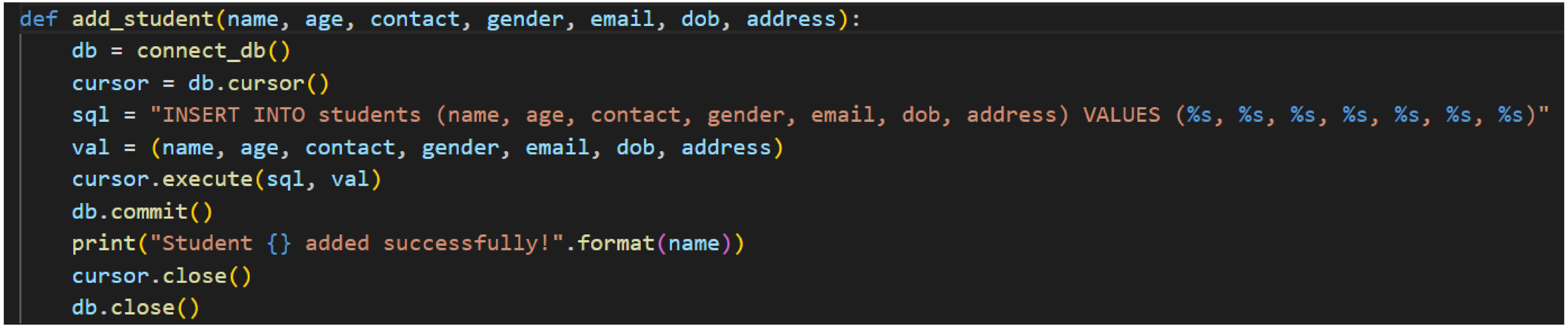


```
Enter your choice: 2
Enter username: Sima123@
Enter password: facsima12
Login successful! Logged in as Faculty.
```

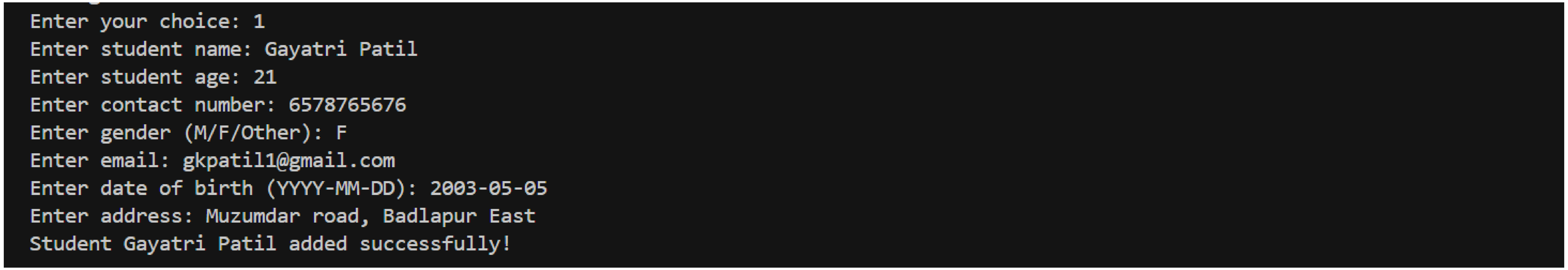
Faculty Login

3. Student Management

3.1. Add a student record.



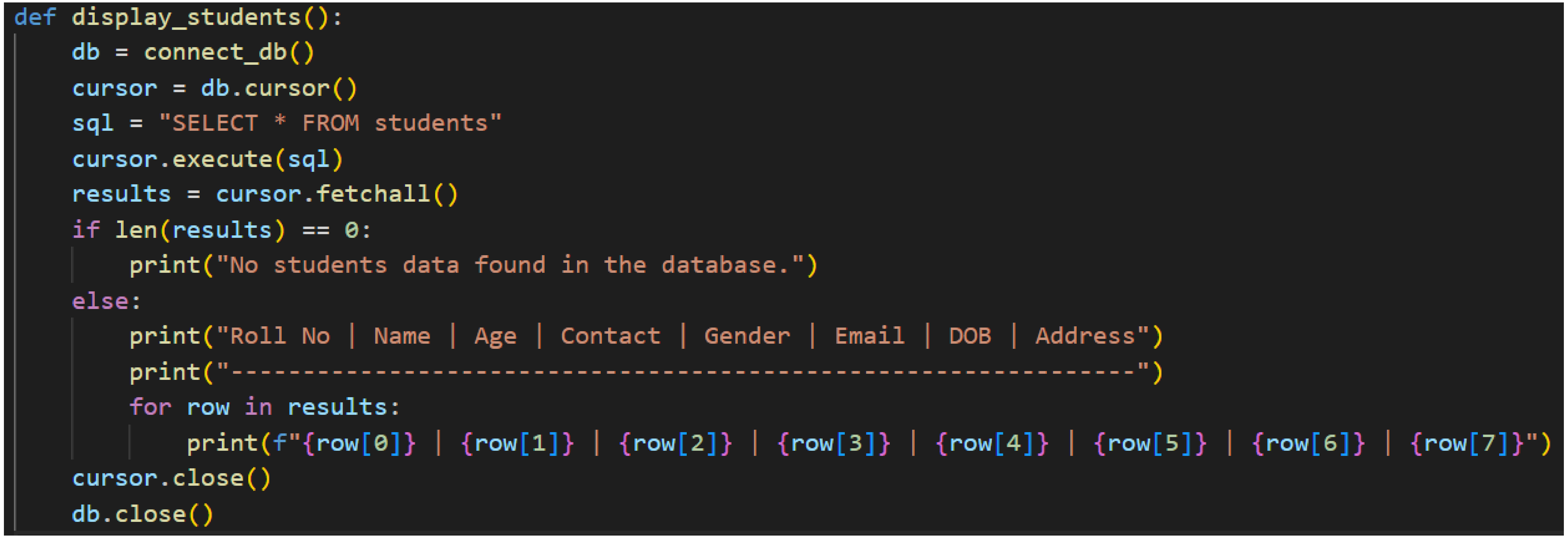
```python
def add_student(name, age, contact, gender, email, dob, address):
    db = connect_db()
    cursor = db.cursor()
    sql = "INSERT INTO students (name, age, contact, gender, email, dob, address) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    val = (name, age, contact, gender, email, dob, address)
    cursor.execute(sql, val)
    db.commit()
    print("Student {} added successfully!".format(name))
    cursor.close()
    db.close()
```



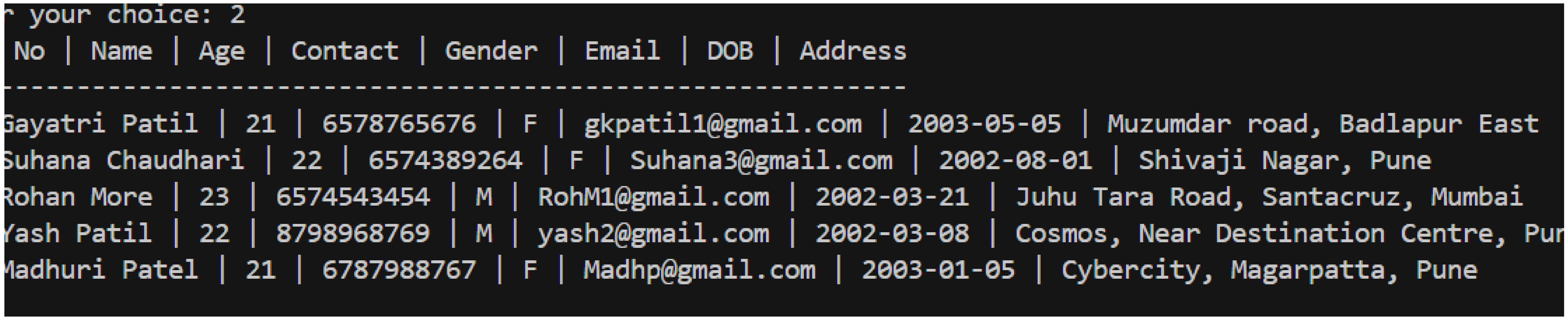
```
Enter your choice: 1
Enter student name: Gayatri Patil
Enter student age: 21
Enter contact number: 6578765676
Enter gender (M/F/Other): F
Enter email: gkpatil1@gmail.com
Enter date of birth (YYYY-MM-DD): 2003-05-05
Enter address: Muzumdar road, Badlapur East
Student Gayatri Patil added successfully!
```

Student Record Added

3.2.Display all student records



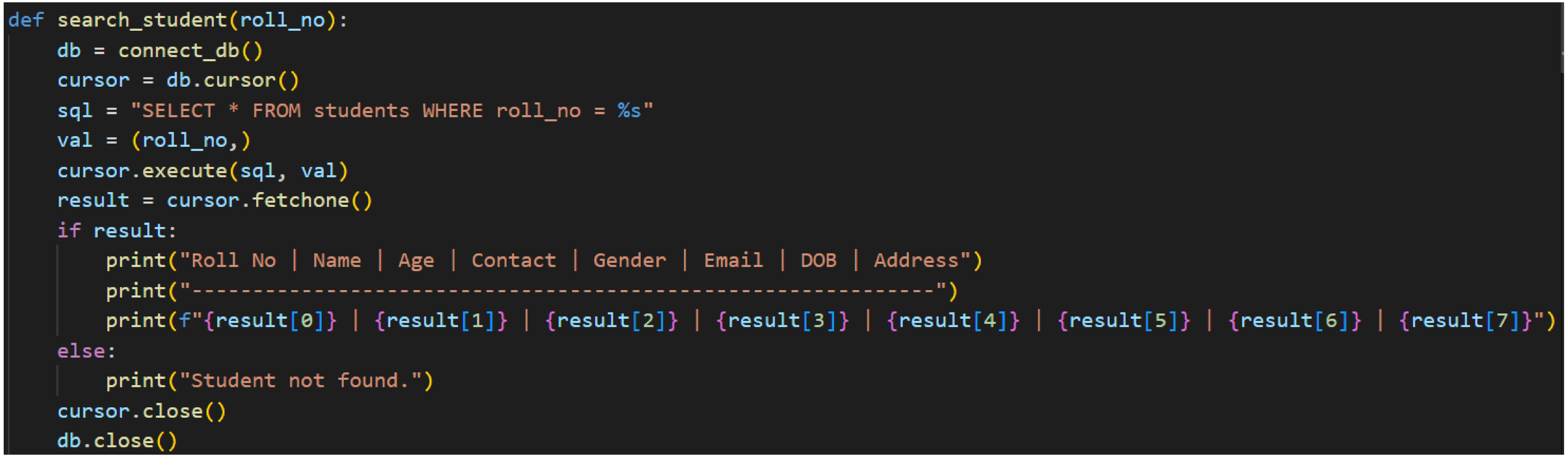
```
def display_students():
    db = connect_db()
    cursor = db.cursor()
    sql = "SELECT * FROM students"
    cursor.execute(sql)
    results = cursor.fetchall()
    if len(results) == 0:
        print("No students data found in the database.")
    else:
        print("Roll No | Name | Age | Contact | Gender | Email | DOB | Address")
        print("---------------------------------------------------------------")
        for row in results:
            print(f"{row[0]} | {row[1]} | {row[2]} | {row[3]} | {row[4]} | {row[5]} | {row[6]} | {row[7]}")
    cursor.close()
    db.close()
```



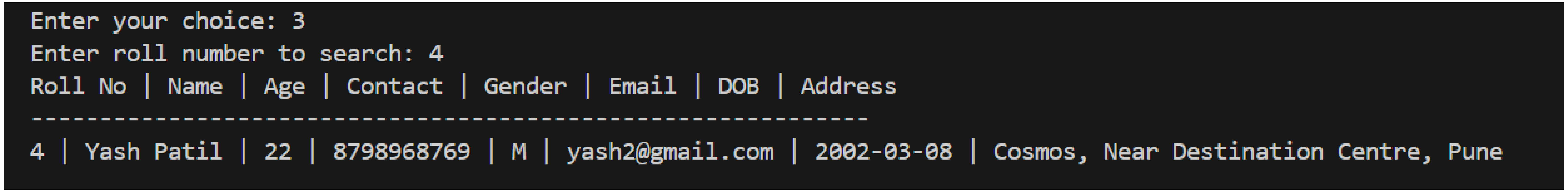
```
r your choice: 2
 No | Name | Age | Contact | Gender | Email | DOB | Address
---------------------------------------------------------------
Gayatri Patil | 21 | 6578765676 | F | gkpatil1@gmail.com | 2003-05-05 | Muzumdar road, Badlapur East
Suhana Chaudhari | 22 | 6574389264 | F | Suhana3@gmail.com | 2002-08-01 | Shivaji Nagar, Pune
Rohan More | 23 | 6574543454 | M | RohM1@gmail.com | 2002-03-21 | Juhu Tara Road, Santacruz, Mumbai
Yash Patil | 22 | 8798968769 | M | yash2@gmail.com | 2002-03-08 | Cosmos, Near Destination Centre, Pur
Madhuri Patel | 21 | 6787988767 | F | Madhp@gmail.com | 2003-01-05 | Cybercity, Magarpatta, Pune
```

Displayed all student records

3.3. Search for a student by roll number.



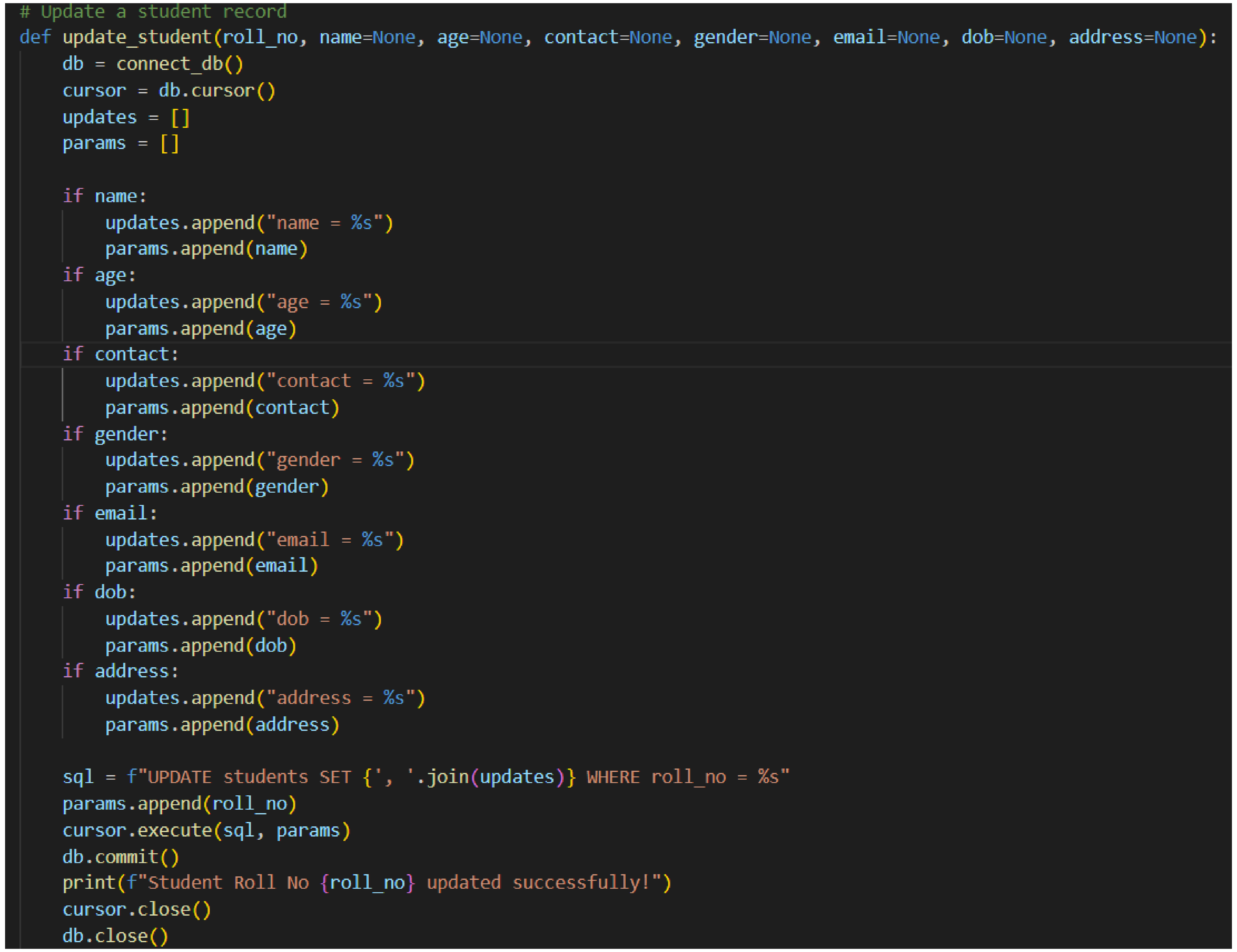
```python
def search_student(roll_no):
    db = connect_db()
    cursor = db.cursor()
    sql = "SELECT * FROM students WHERE roll_no = %s"
    val = (roll_no,)
    cursor.execute(sql, val)
    result = cursor.fetchone()
    if result:
        print("Roll No | Name | Age | Contact | Gender | Email | DOB | Address")
        print("-------------------------------------------------------------")
        print(f"{result[0]} | {result[1]} | {result[2]} | {result[3]} | {result[4]} | {result[5]} | {result[6]} | {result[7]}")
    else:
        print("Student not found.")
    cursor.close()
    db.close()
```



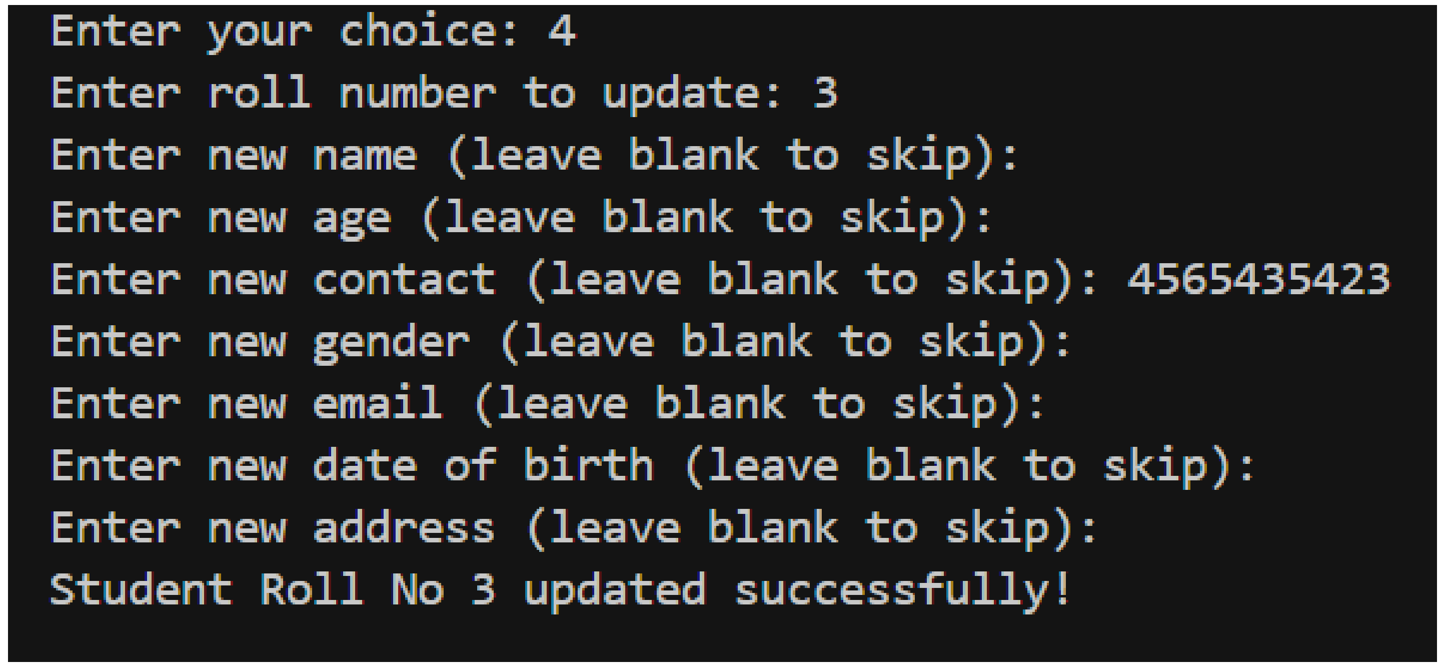
```
Enter your choice: 3
Enter roll number to search: 4
Roll No | Name | Age | Contact | Gender | Email | DOB | Address
-------------------------------------------------------------
4 | Yash Patil | 22 | 8798968769 | M | yash2@gmail.com | 2002-03-08 | Cosmos, Near Destination Centre, Pune
```

Searched student by roll no. - 4

3.4. Update a student record.



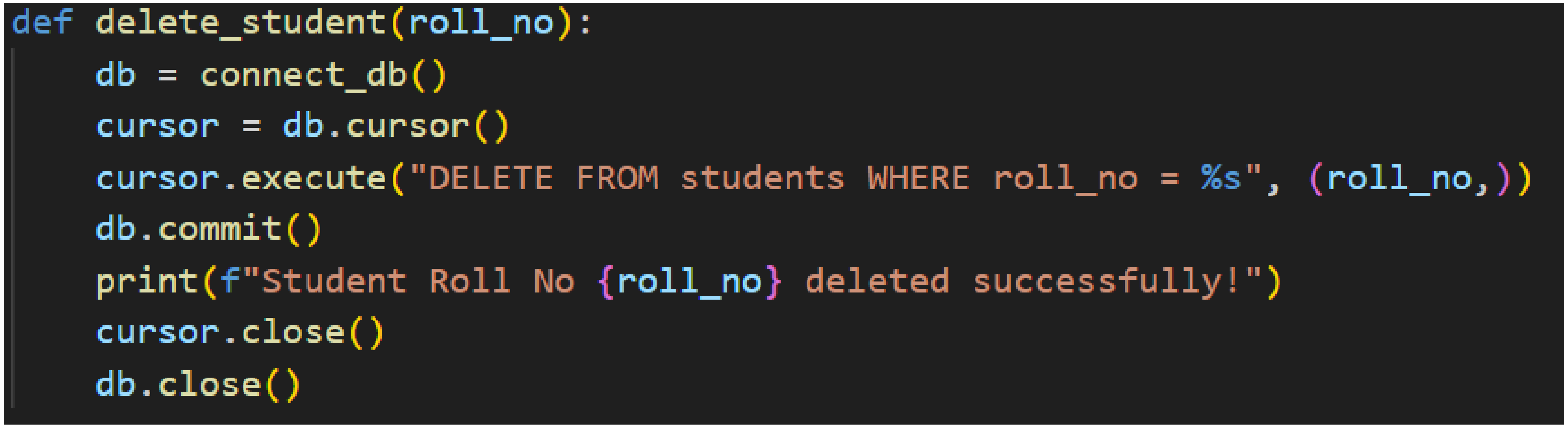
```python
# Update a student record
def update_student(roll_no, name=None, age=None, contact=None, gender=None, email=None, dob=None, address=None):
    db = connect_db()
    cursor = db.cursor()
    updates = []
    params = []

    if name:
        updates.append("name = %s")
        params.append(name)
    if age:
        updates.append("age = %s")
        params.append(age)
    if contact:
        updates.append("contact = %s")
        params.append(contact)
    if gender:
        updates.append("gender = %s")
        params.append(gender)
    if email:
        updates.append("email = %s")
        params.append(email)
    if dob:
        updates.append("dob = %s")
        params.append(dob)
    if address:
        updates.append("address = %s")
        params.append(address)

    sql = f"UPDATE students SET {', '.join(updates)} WHERE roll_no = %s"
    params.append(roll_no)
    cursor.execute(sql, params)
    db.commit()
    print(f"Student Roll No {roll_no} updated successfully!")
    cursor.close()
    db.close()
```



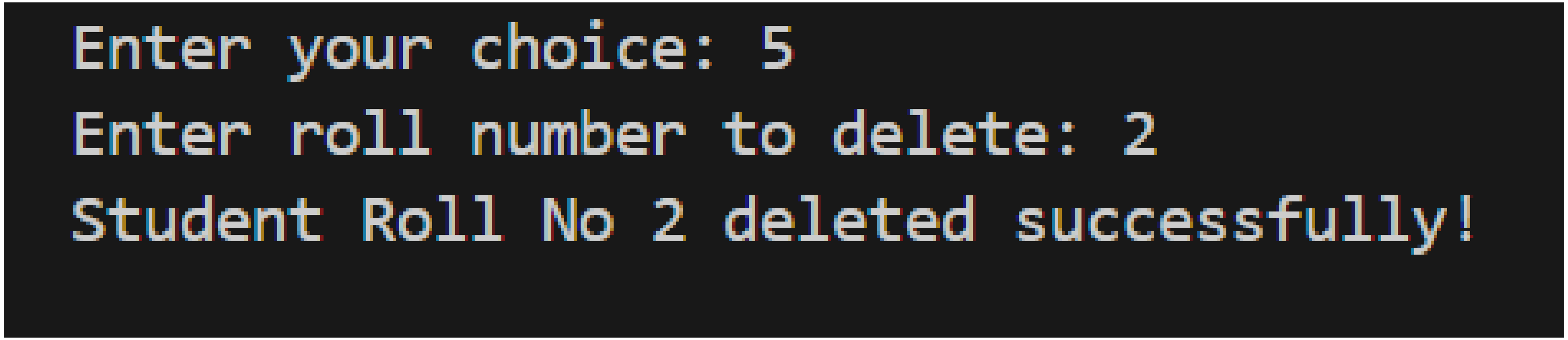
```
Enter your choice: 4
Enter roll number to update: 3
Enter new name (leave blank to skip):
Enter new age (leave blank to skip):
Enter new contact (leave blank to skip): 4565435423
Enter new gender (leave blank to skip):
Enter new email (leave blank to skip):
Enter new date of birth (leave blank to skip):
Enter new address (leave blank to skip):
Student Roll No 3 updated successfully!
```

Updated student record

3.5. Delete a student record.



```
def delete_student(roll_no):
    db = connect_db()
    cursor = db.cursor()
    cursor.execute("DELETE FROM students WHERE roll_no = %s", (roll_no,))
    db.commit()
    print(f"Student Roll No {roll_no} deleted successfully!")
    cursor.close()
    db.close()
```

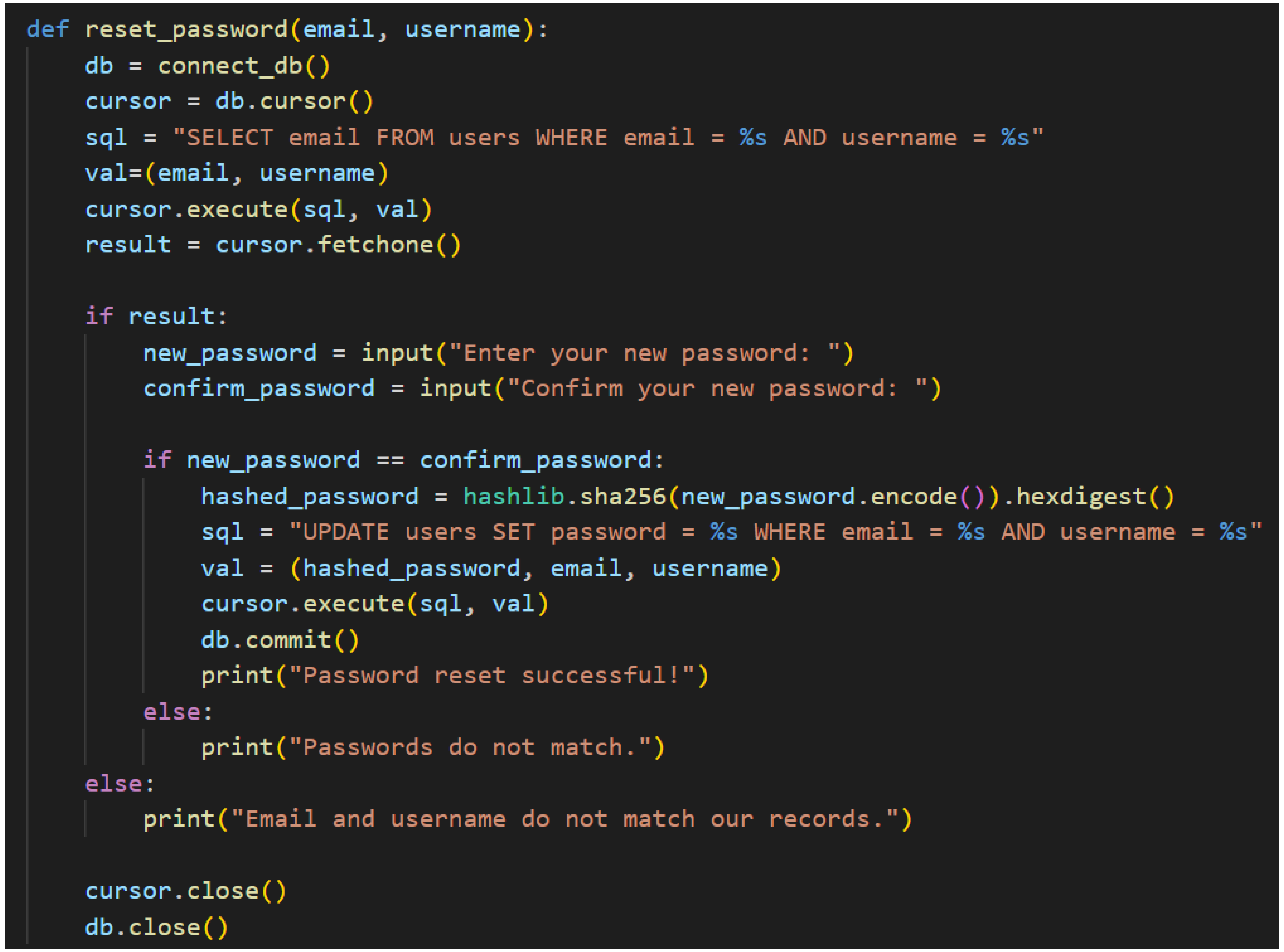


```
Enter your choice: 5
Enter roll number to delete: 2
Student Roll No 2 deleted successfully!
```

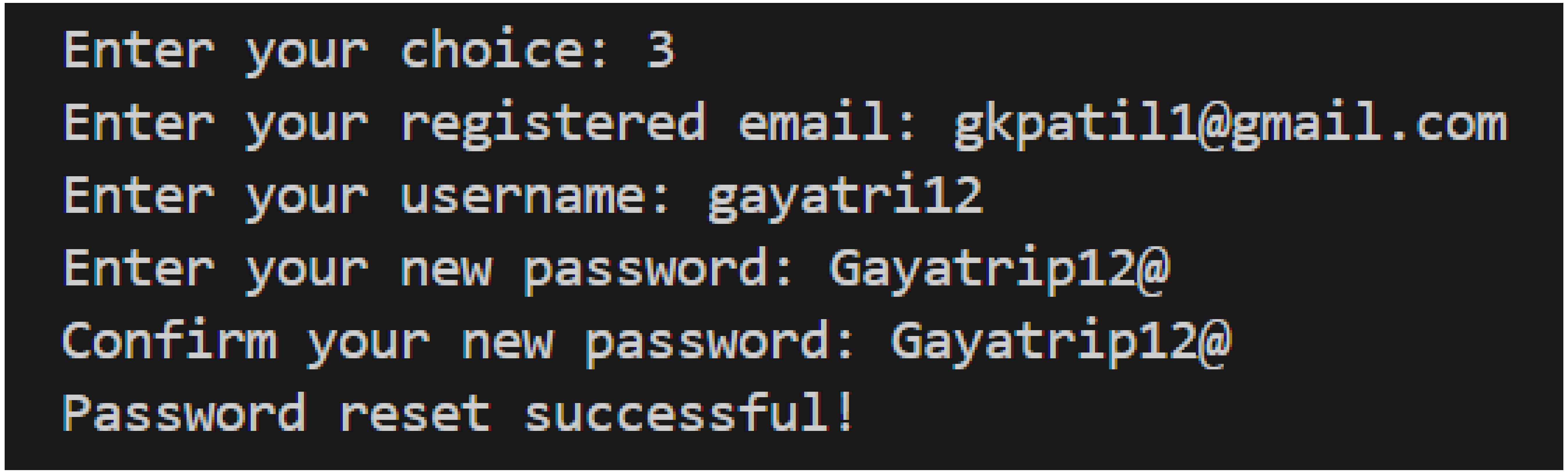
Deleted Student record

4. Password Reset Functionality

Allows users to reset passwords securely with email and username verification.

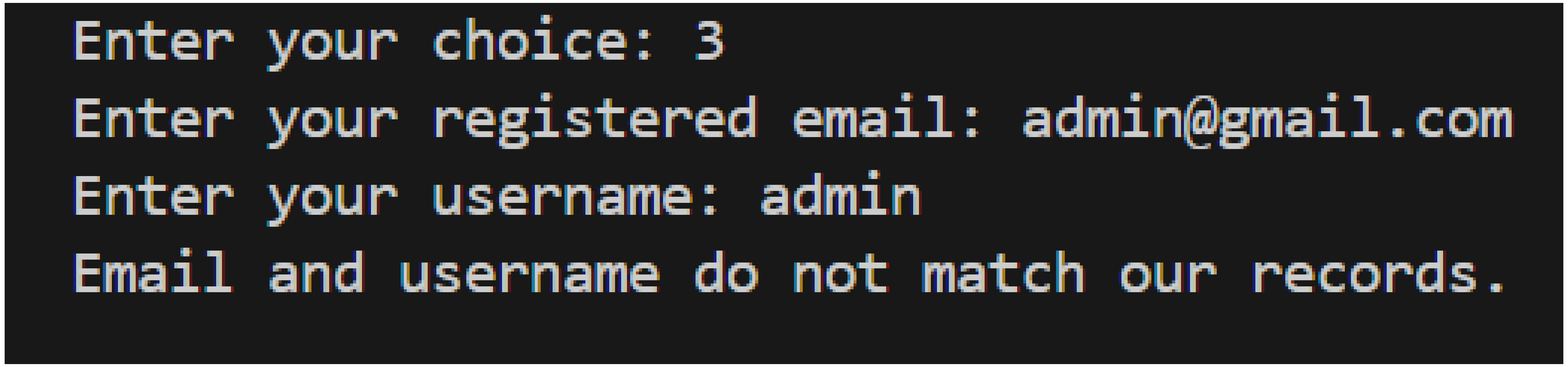


```python
def reset_password(email, username):
    db = connect_db()
    cursor = db.cursor()
    sql = "SELECT email FROM users WHERE email = %s AND username = %s"
    val=(email, username)
    cursor.execute(sql, val)
    result = cursor.fetchone()

    if result:
        new_password = input("Enter your new password: ")
        confirm_password = input("Confirm your new password: ")

        if new_password == confirm_password:
            hashed_password = hashlib.sha256(new_password.encode()).hexdigest()
            sql = "UPDATE users SET password = %s WHERE email = %s AND username = %s"
            val = (hashed_password, email, username)
            cursor.execute(sql, val)
            db.commit()
            print("Password reset successful!")
        else:
            print("Passwords do not match.")
    else:
        print("Email and username do not match our records.")

    cursor.close()
    db.close()
```



```
Enter your choice: 3
Enter your registered email: gkpatil1@gmail.com
Enter your username: gayatri12
Enter your new password: Gayatrip12@
Confirm your new password: Gayatrip12@
Password reset successful!
```

Password reset successful

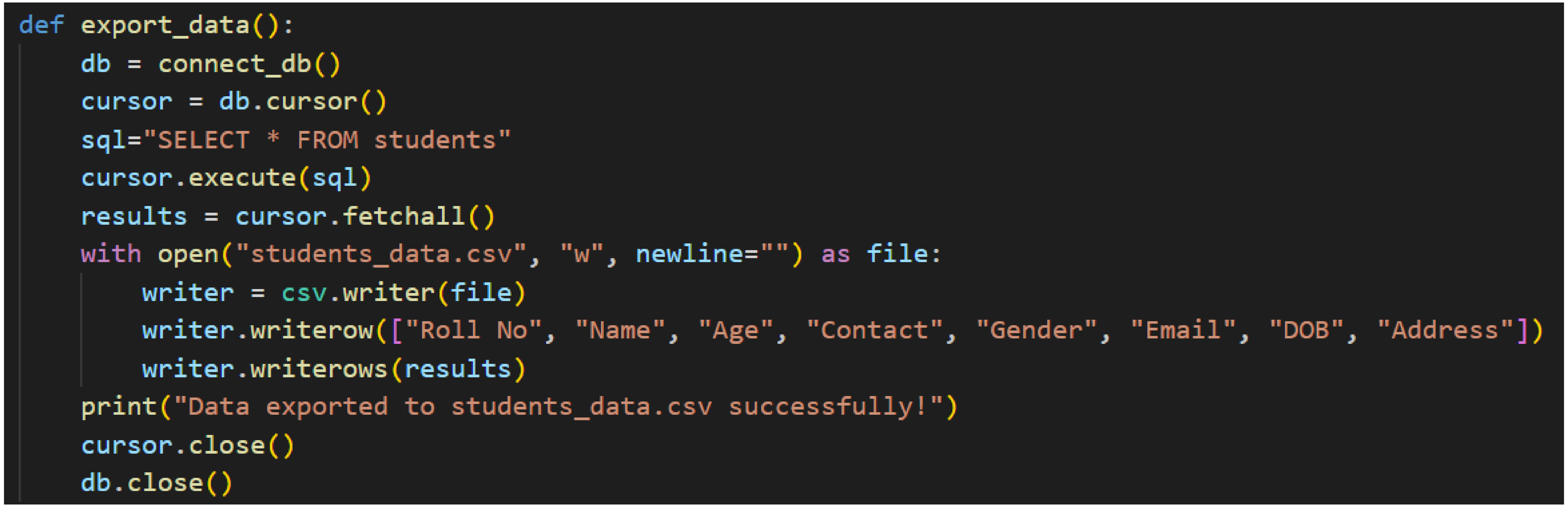


```
Enter your choice: 3
Enter your registered email: admin@gmail.com
Enter your username: admin
Email and username do not match our records.
```

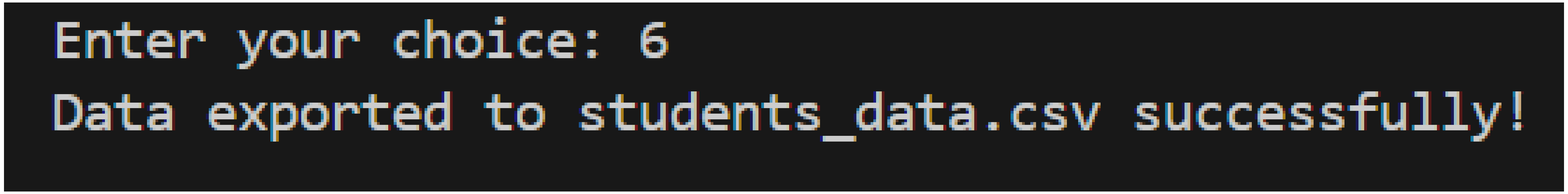
Password reset unsuccessful

5. Data Export

Provides data export to CSV format for easy analysis or external use.

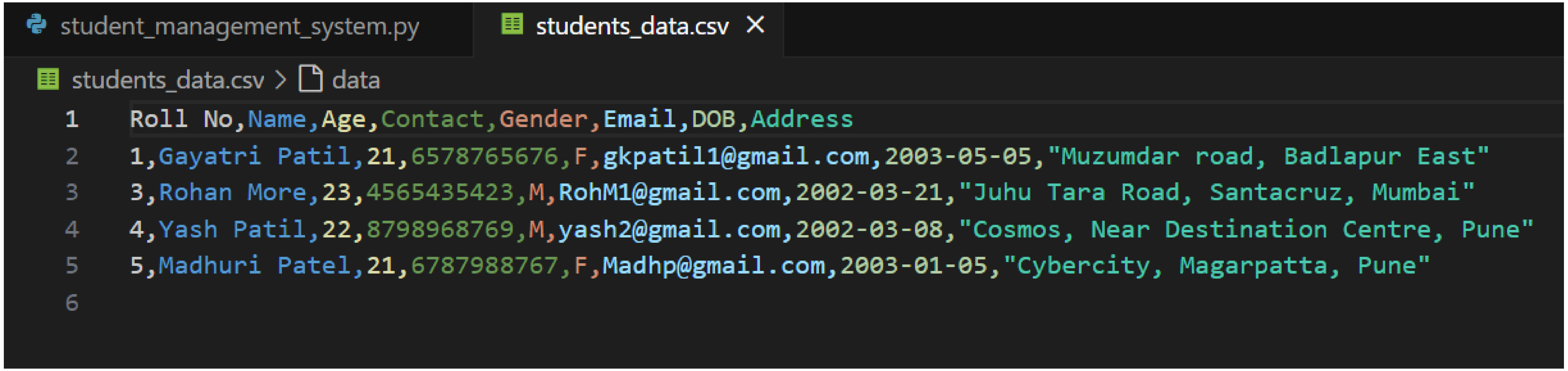


```python
def export_data():
    db = connect_db()
    cursor = db.cursor()
    sql="SELECT * FROM students"
    cursor.execute(sql)
    results = cursor.fetchall()
    with open("students_data.csv", "w", newline="") as file:
        writer = csv.writer(file)
        writer.writerow(["Roll No", "Name", "Age", "Contact", "Gender", "Email", "DOB", "Address"])
        writer.writerows(results)
    print("Data exported to students_data.csv successfully!")
    cursor.close()
    db.close()
```



```
Enter your choice: 6
Data exported to students_data.csv successfully!
```

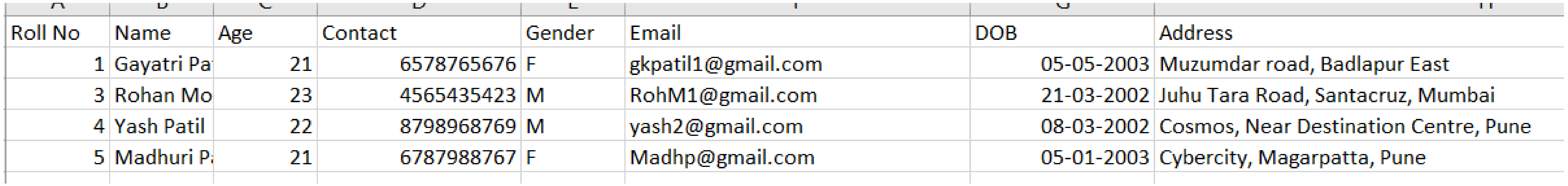
Data exported in csv format



student_management_system.py   students_data.csv ×

students_data.csv > data

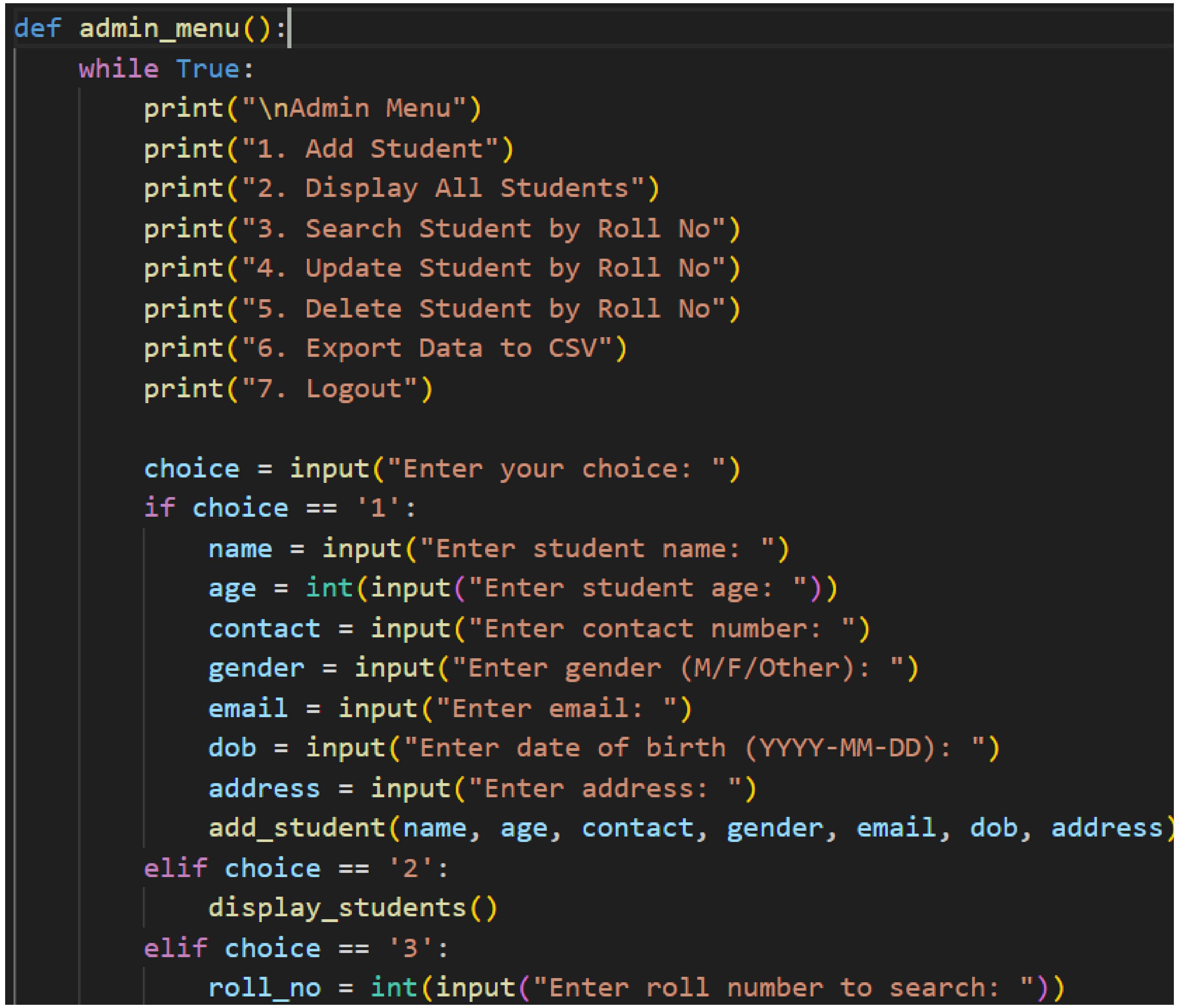
```
Roll No,Name,Age,Contact,Gender,Email,DOB,Address
1,Gayatri Patil,21,6578765676,F,gkpatil1@gmail.com,2003-05-05,"Muzumdar road, Badlapur East"
3,Rohan More,23,4565435423,M,RohM1@gmail.com,2002-03-21,"Juhu Tara Road, Santacruz, Mumbai"
4,Yash Patil,22,8798968769,M,yash2@gmail.com,2002-03-08,"Cosmos, Near Destination Centre, Pune"
5,Madhuri Patel,21,6787988767,F,Madhp@gmail.com,2003-01-05,"Cybercity, Magarpatta, Pune"
```



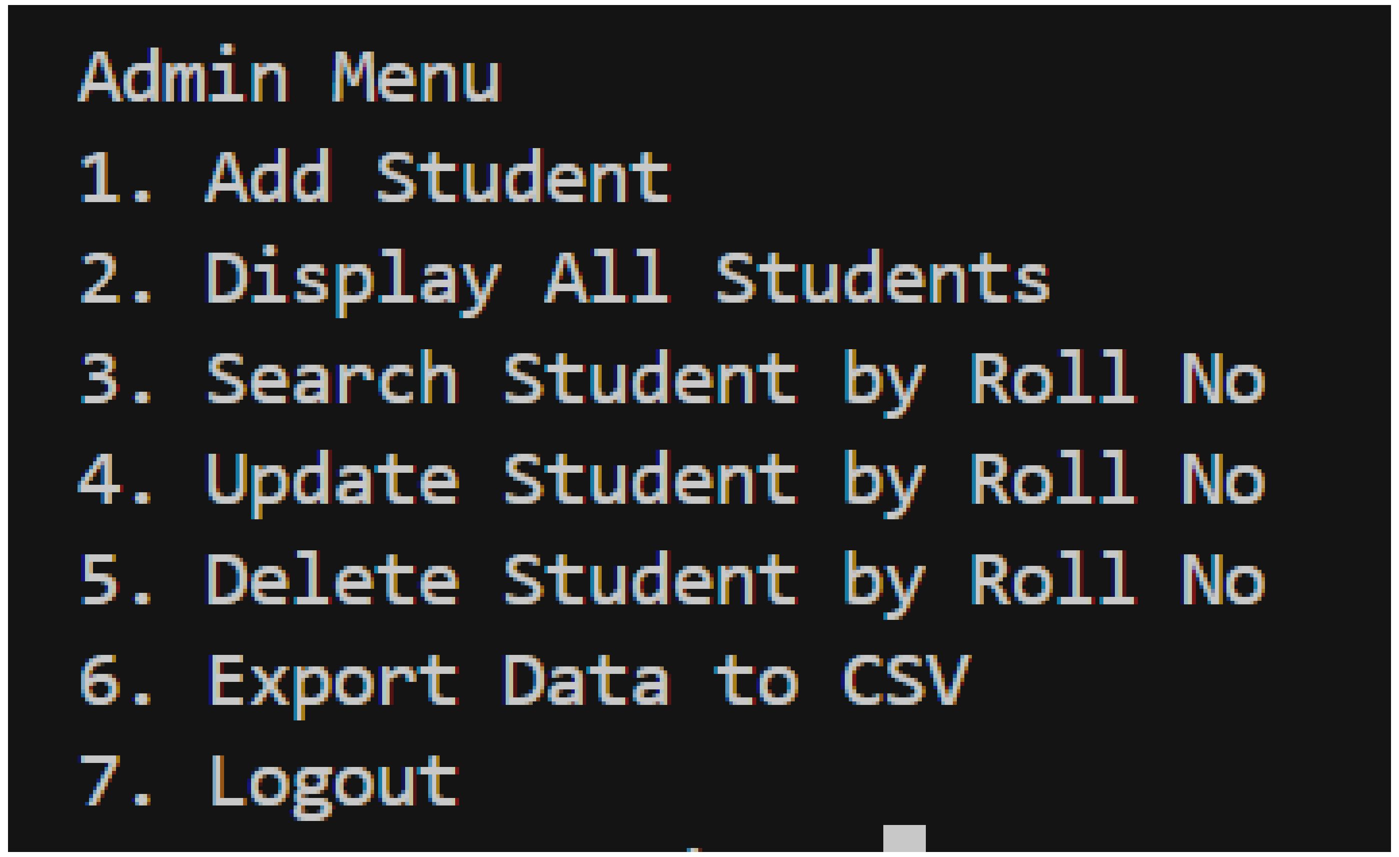
Roll No	Name	Age	Contact	Gender	Email	DOB	Address
1	Gayatri Pa	21	6578765676	F	gkpatil1@gmail.com	05-05-2003	Muzumdar road, Badlapur East
3	Rohan Mo	23	4565435423	M	RohM1@gmail.com	21-03-2002	Juhu Tara Road, Santacruz, Mumbai
4	Yash Patil	22	8798968769	M	yash2@gmail.com	08-03-2002	Cosmos, Near Destination Centre, Pune
5	Madhuri Pa	21	6787988767	F	Madhp@gmail.com	05-01-2003	Cybercity, Magarpatta, Pune

CSV FILE

6.1. Role-based functionality - Admin Menu



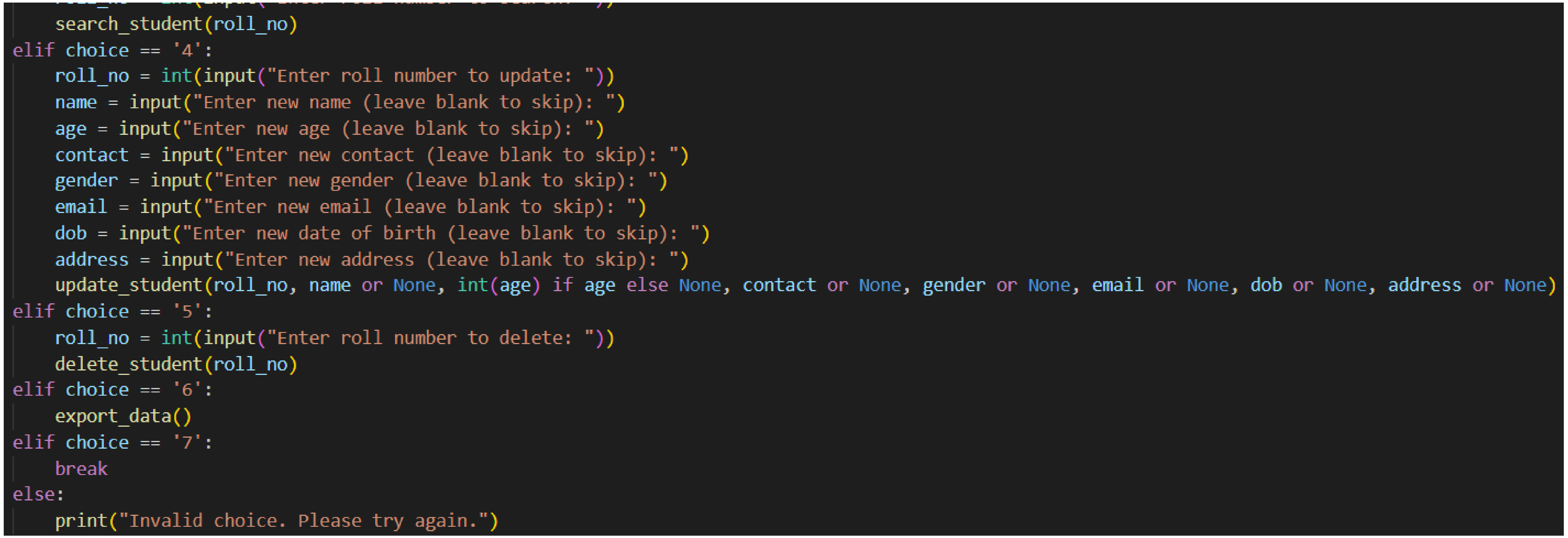
```
def admin_menu():
    while True:
        print("\nAdmin Menu")
        print("1. Add Student")
        print("2. Display All Students")
        print("3. Search Student by Roll No")
        print("4. Update Student by Roll No")
        print("5. Delete Student by Roll No")
        print("6. Export Data to CSV")
        print("7. Logout")

        choice = input("Enter your choice: ")
        if choice == '1':
            name = input("Enter student name: ")
            age = int(input("Enter student age: "))
            contact = input("Enter contact number: ")
            gender = input("Enter gender (M/F/Other): ")
            email = input("Enter email: ")
            dob = input("Enter date of birth (YYYY-MM-DD): ")
            address = input("Enter address: ")
            add_student(name, age, contact, gender, email, dob, address)
        elif choice == '2':
            display_students()
        elif choice == '3':
            roll_no = int(input("Enter roll number to search: "))
```



```
Admin Menu
1. Add Student
2. Display All Students
3. Search Student by Roll No
4. Update Student by Roll No
5. Delete Student by Roll No
6. Export Data to CSV
7. Logout
```

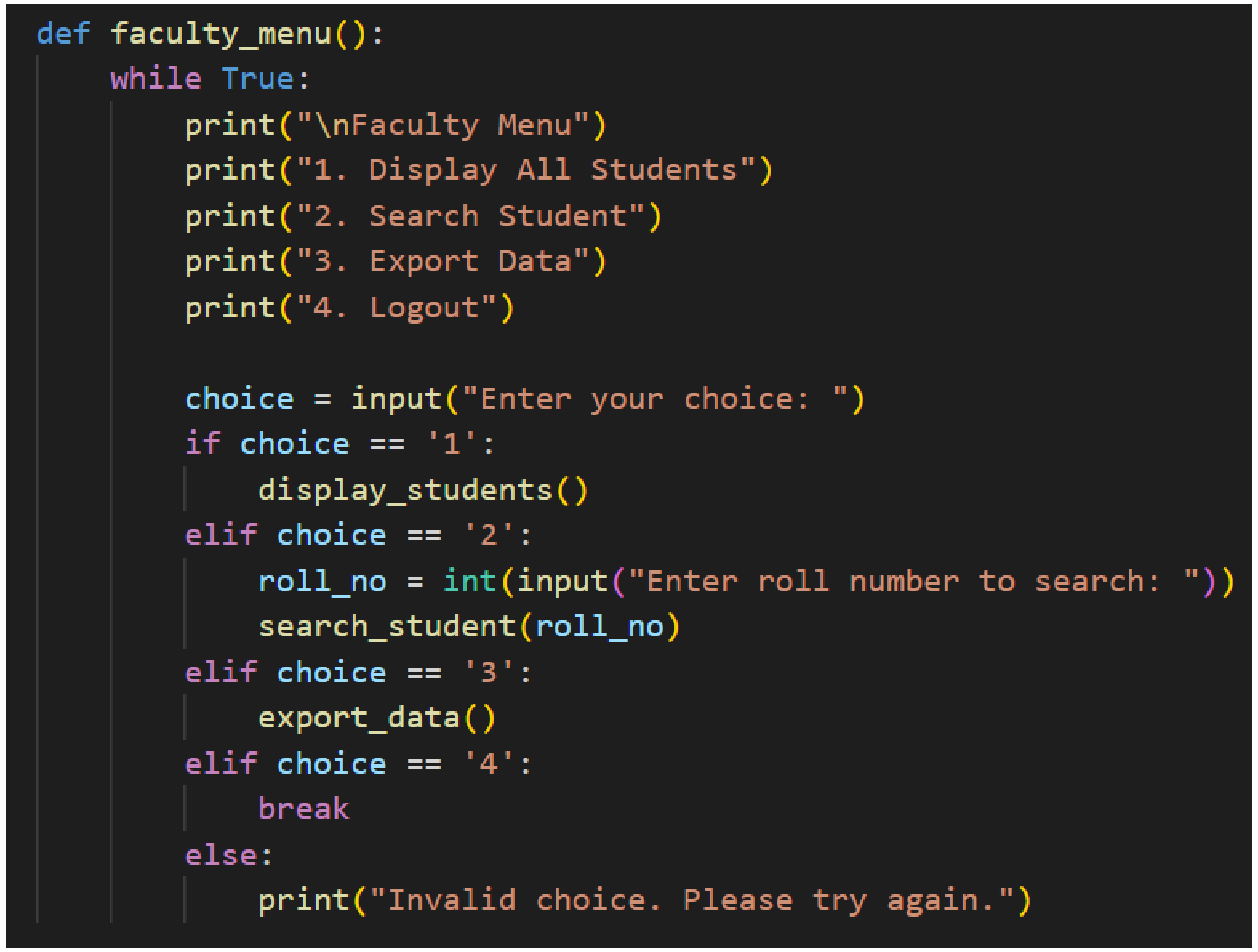
Admin Menu

6.1. Role-based functionality - Admin Menu

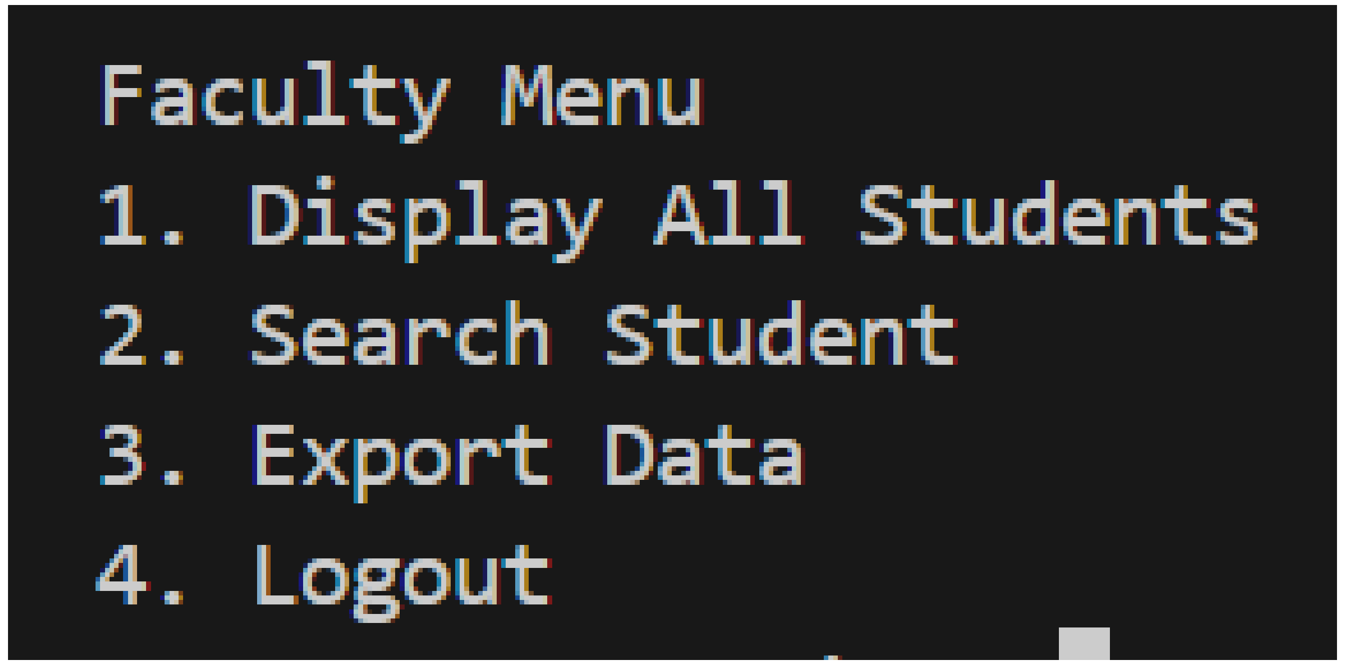


```
    search_student(roll_no)
elif choice == '4':
    roll_no = int(input("Enter roll number to update: "))
    name = input("Enter new name (leave blank to skip): ")
    age = input("Enter new age (leave blank to skip): ")
    contact = input("Enter new contact (leave blank to skip): ")
    gender = input("Enter new gender (leave blank to skip): ")
    email = input("Enter new email (leave blank to skip): ")
    dob = input("Enter new date of birth (leave blank to skip): ")
    address = input("Enter new address (leave blank to skip): ")
    update_student(roll_no, name or None, int(age) if age else None, contact or None, gender or None, email or None, dob or None, address or None)
elif choice == '5':
    roll_no = int(input("Enter roll number to delete: "))
    delete_student(roll_no)
elif choice == '6':
    export_data()
elif choice == '7':
    break
else:
    print("Invalid choice. Please try again.")
```

6.2. Role-based functionality - Faculty Menu



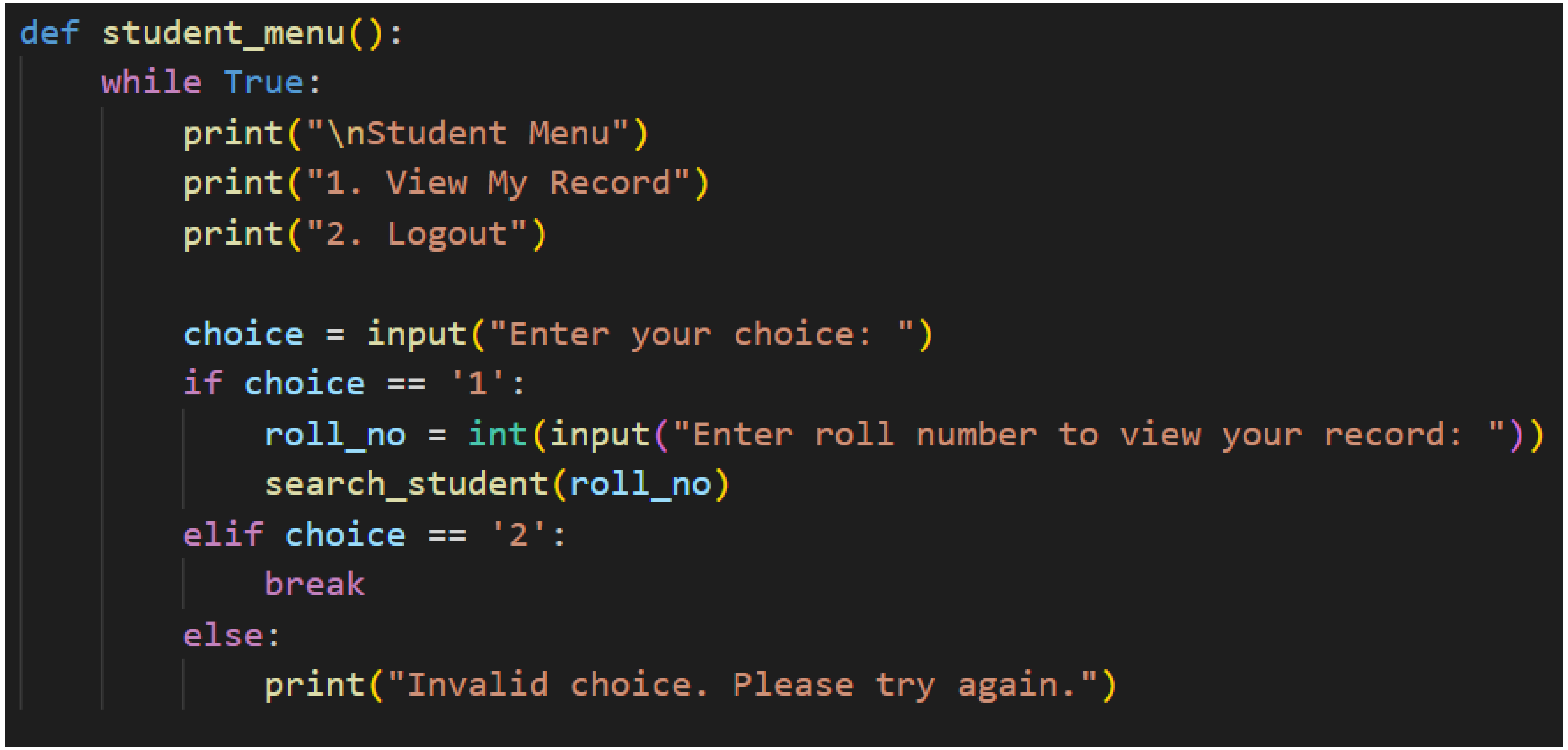
```python
def faculty_menu():
    while True:
        print("\nFaculty Menu")
        print("1. Display All Students")
        print("2. Search Student")
        print("3. Export Data")
        print("4. Logout")

        choice = input("Enter your choice: ")
        if choice == '1':
            display_students()
        elif choice == '2':
            roll_no = int(input("Enter roll number to search: "))
            search_student(roll_no)
        elif choice == '3':
            export_data()
        elif choice == '4':
            break
        else:
            print("Invalid choice. Please try again.")
```



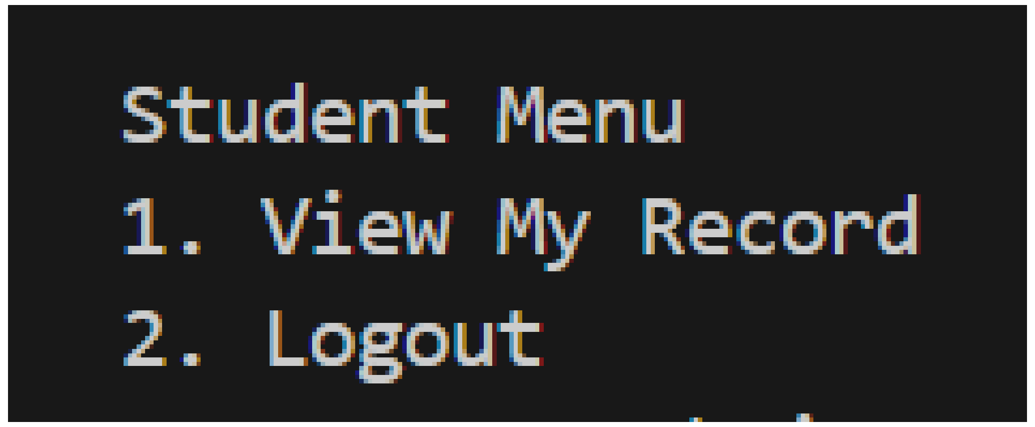
```
Faculty Menu
1. Display All Students
2. Search Student
3. Export Data
4. Logout
```

Faculty Menu

6.3. Role-based functionality - Student Menu



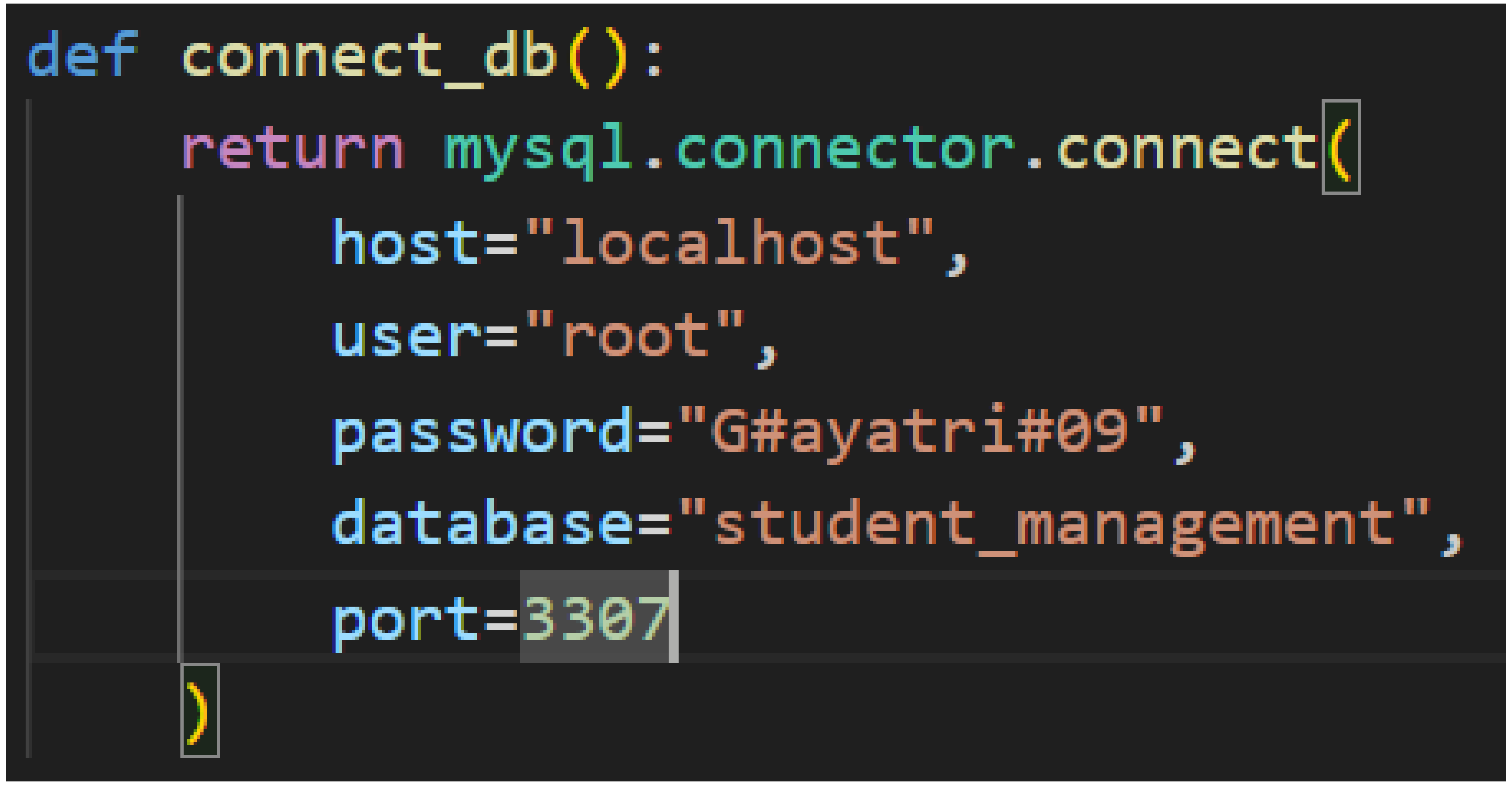
```python
def student_menu():
    while True:
        print("\nStudent Menu")
        print("1. View My Record")
        print("2. Logout")

        choice = input("Enter your choice: ")
        if choice == '1':
            roll_no = int(input("Enter roll number to view your record: "))
            search_student(roll_no)
        elif choice == '2':
            break
        else:
            print("Invalid choice. Please try again.")
```



```
Student Menu
1. View My Record
2. Logout
```

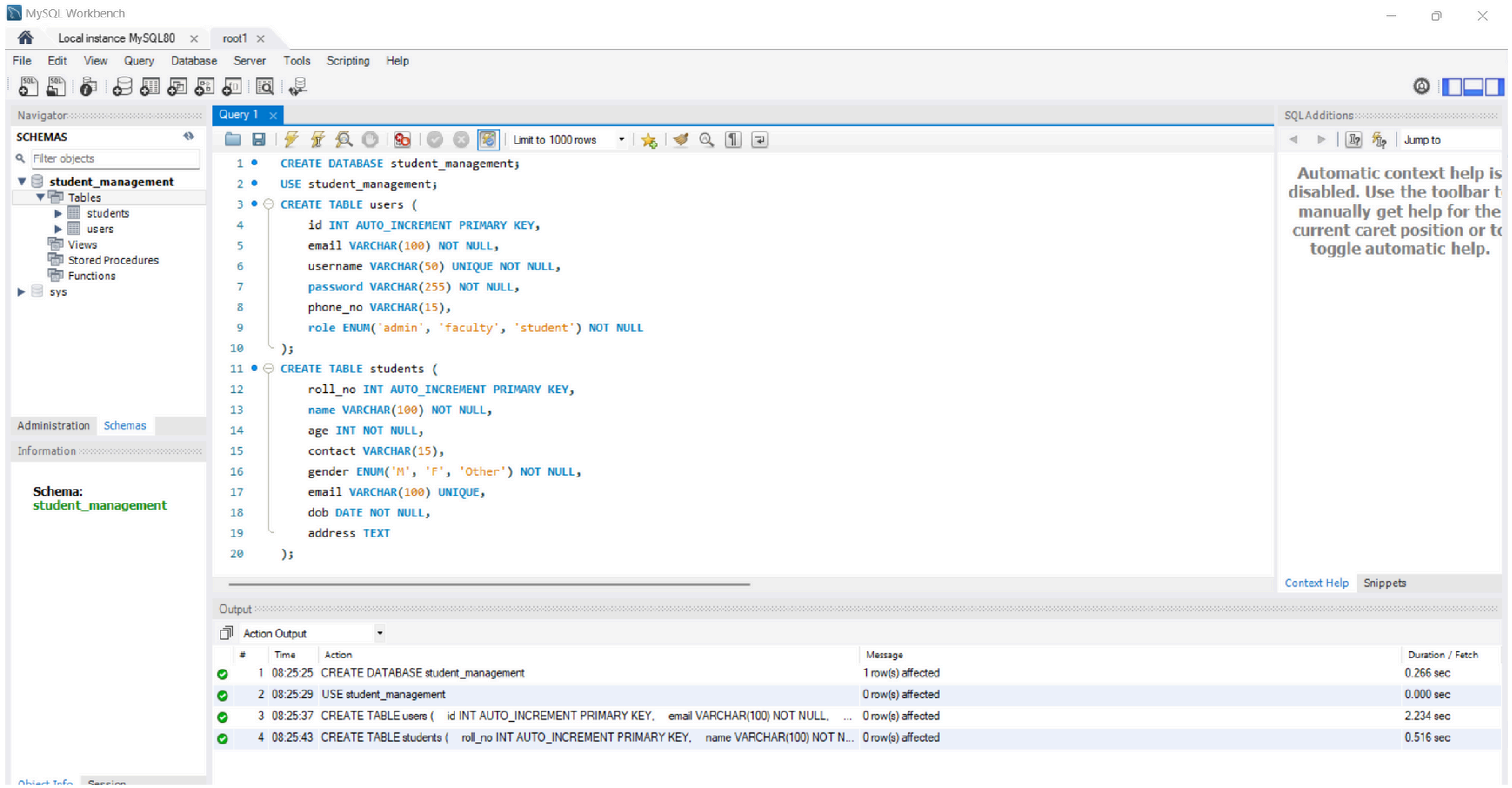
Student Menu

7. Establishing connection to the MySQL database



```
def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="G#ayatri#09",
        database="student_management",
        port=3307
    )
```

8. MySQL Workbench

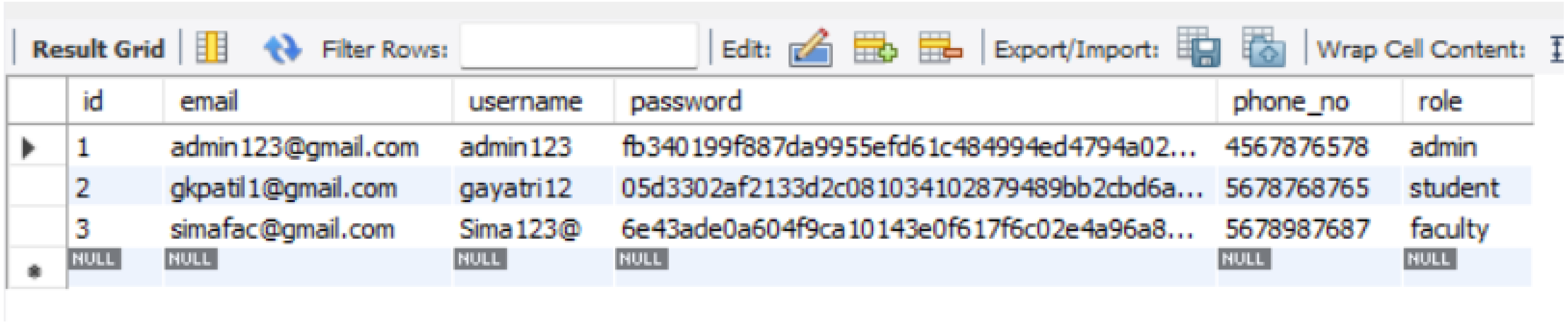


```sql
CREATE DATABASE student_management;
USE student_management;
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(100) NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone_no VARCHAR(15),
    role ENUM('admin', 'faculty', 'student') NOT NULL
);
CREATE TABLE students (
    roll_no INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    contact VARCHAR(15),
    gender ENUM('M', 'F', 'Other') NOT NULL,
    email VARCHAR(100) UNIQUE,
    dob DATE NOT NULL,
    address TEXT
);
```

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

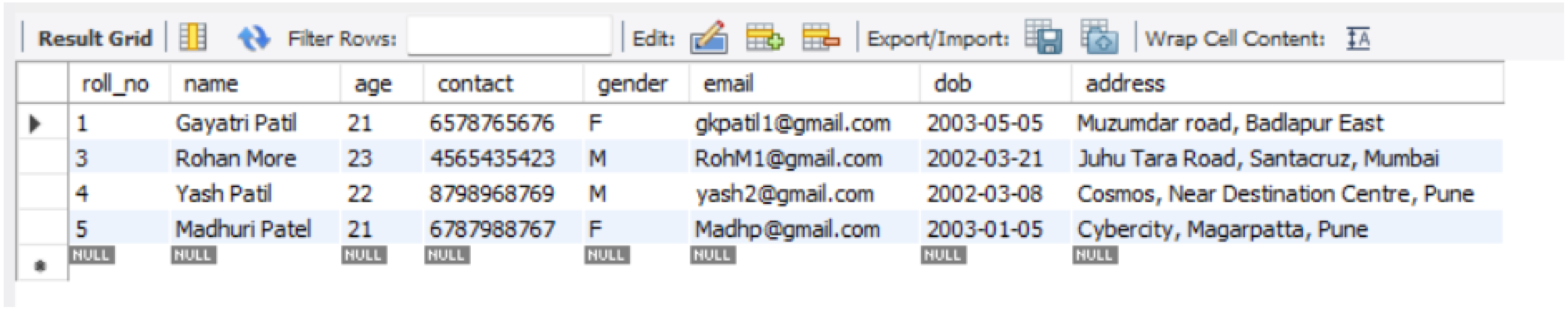
Schema: student_management

	#	Time	Action	Message	Duration / Fetch
✓	1	08:25:25	CREATE DATABASE student_management	1 row(s) affected	0.266 sec
✓	2	08:25:29	USE student_management	0 row(s) affected	0.000 sec
✓	3	08:25:37	CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, email VARCHAR(100) NOT NULL, ...	0 row(s) affected	2.234 sec
✓	4	08:25:43	CREATE TABLE students (roll_no INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT N...	0 row(s) affected	0.516 sec



id	email	username	password	phone_no	role
1	admin123@gmail.com	admin123	fb340199f887da9955efd61c484994ed4794a02...	4567876578	admin
2	gkpatil1@gmail.com	gayatri12	05d3302af2133d2c081034102879489bb2cbd6a...	5678768765	student
3	simafac@gmail.com	Sima123@	6e43ade0a604f9ca10143e0f617f6c02e4a96a8...	5678987687	faculty
NULL	NULL	NULL	NULL	NULL	NULL

Users Table



roll_no	name	age	contact	gender	email	dob	address
1	Gayatri Patil	21	6578765676	F	gkpatil1@gmail.com	2003-05-05	Muzumdar road, Badlapur East
3	Rohan More	23	4565435423	M	RohM1@gmail.com	2002-03-21	Juhu Tara Road, Santacruz, Mumbai
4	Yash Patil	22	8798968769	M	yash2@gmail.com	2002-03-08	Cosmos, Near Destination Centre, Pune
5	Madhuri Patel	21	6787988767	F	Madhp@gmail.com	2003-01-05	Cybercity, Magarpatta, Pune
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Students Table

Outcomes

1. Simplified student record management processes.
2. Enhanced data consistency with validation checks.
3. Enabled efficient student data retrieval with fast search.
4. Developed role-based menus for user-specific access.
5. Secured sensitive data with password hashing and access controls.
6. Enabled student data export to CSV for easy reporting.
7. Designed for scalability and future enhancements.

Thank You