ALL SAINT'S CHURCH SR.SEC. SCHOOL

M.I. ROAD , JAIPUR

A PROJECT REPORT ON

HOSPITAL MANAGEMENT SYSTEM

SUBJECT: INFORMATICS PRACTICES (065)

Session: 2023-2024

Name of Participants:- Sufiyan Qureshi

Mohd.Faizan

Ayan Khan

Exam Roll No. - _____

Submitted By:
Sufiyan Qureshi

Submitted To:
Mrs. Sharon Hiskiel (PGT IP)

CERTIFICATE

This is to certify that Sufiyan 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 Hiskiel, PGT COMPUTER. The project work, carried out by us 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 "Hospital 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.

Sufiyan

(XII)

INDEX

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.

PANDAS INTRODUCTION

- **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.**

MATPLOTLIB INTRODUCTION

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram, etc.

Matplotlib comes with a wide variety of plots. Plots help to understand trends, and patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

MySQL INTRODUCTION

MySQL is a relational database management system(RDBMS) developed by Oracle that is based on structured query language(SQL). A database is a structured collection of data.It may be anything from a simple shopping list to a picture gallery or a place to hold the vast amounts of information in a corporate network.In particular,a reletional database is a digital store collecting data and organizing it according to the relational modal. In this modal,tables consist of rows and columns and relationship between data elements all follow a strict logical structure. An RDBMS is simply te set of software tools used to actually implement,manage and query such a database

System Requirements

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

Device name	**DESKTOP-LFQNHSQ**
Processor	**Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz**
Installed	**RAM 8.00 GB (7.88 GB usable)**
Device ID	**A123CFF6-C1D0-41EE-A3A4-21D7FAEBE73B**
Product ID	**00330-50419-68709-AAOEM**
System type	**64-bit operating system, x64-based processor**
Pen and touch	**Touch support with 10 touch points**

INTRODUCTION OF PROJECT

PROJECT TITLE-(HOSPITAL MANAGEMENT)

DBMS: MySQL

Host : localhost

User: root
Password: root
Database: HOSPITAL

Table Structure: As per the Screenshot given below:

Screenshots OF PROJECT

Appointments table has following Schema

```
mysql> desc appointments;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| Id         | int(11)      | NO   | PRI | NULL    | auto_increment |
| patient_id | int(11)      | YES  |     | NULL    |                |
| doctor_id  | int(11)      | YES  |     | NULL    |                |
| date       | date         | YES  |     | NULL    |                |
| time       | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Hosp table has following Schema

```
mysql> desc hospitals;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| hospital_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| name        | varchar(255) | YES  |     | NULL    |                |
| location    | varchar(255) | YES  |     | NULL    |                |
| capacity    | int(11)      | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Doctors table has following Schema

```
mysql> desc doctors;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| Id             | int(11)      | NO   | PRI | NULL    | auto_increment |
| name           | varchar(255) | YES  |     | NULL    |                |
| specialization | varchar(255) | YES  |     | NULL    |                |
| contact        | varchar(15)  | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Patients table has following Schema

```
mysql> desc patients;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| Id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| name    | varchar(255) | YES  |     | NULL    |                |
| age     | int(11)      | YES  |     | NULL    |                |
| gender  | varchar(10)  | YES  |     | NULL    |                |
| contact | varchar(15)  | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

USER OUTPUT

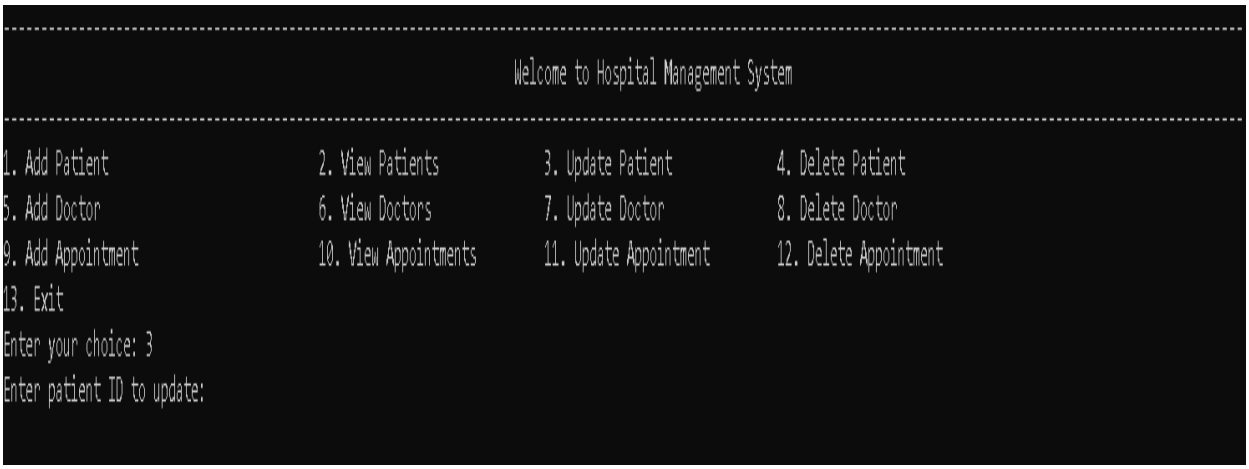
PATIENT MODULE DETAILS:

```
-----------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
-----------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 1
Enter patient name: Nick
Enter patient age: 17
Enter patient gender: Male
Enter patient contact number: 9024563258
1 record(s) inserted.
Press ENTER KEY to continue.....
```

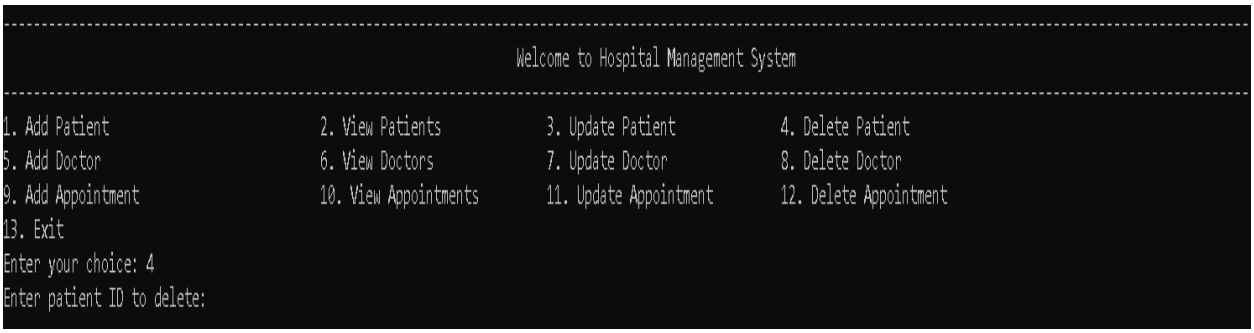
PATIENT MODULE DETAILS:

```
-----------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
-----------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 2
Press (f) to see in the form of DataFrame
Press (l) to see in the form of line graph
Press (h) to see in the form of histogram graph
Enter your choice:
```

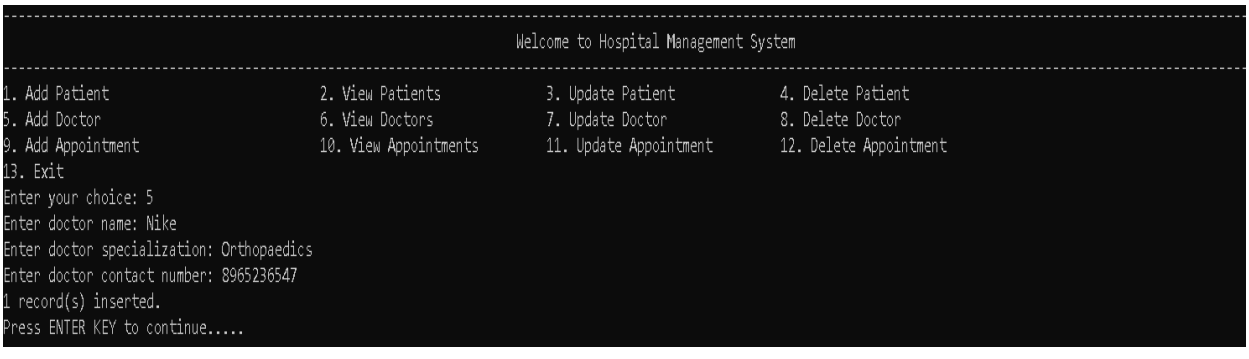
PATIENT MODULE DETAILS:



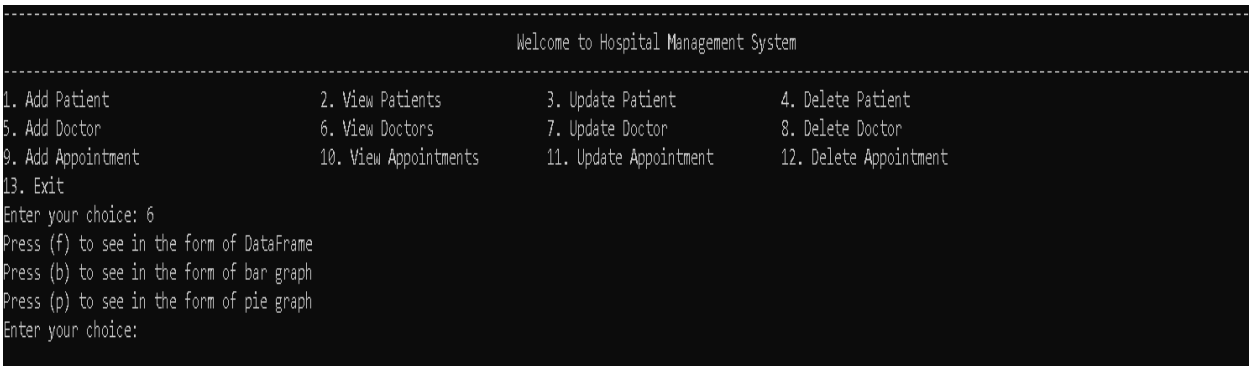
PATIENT MODULE DETAILS:



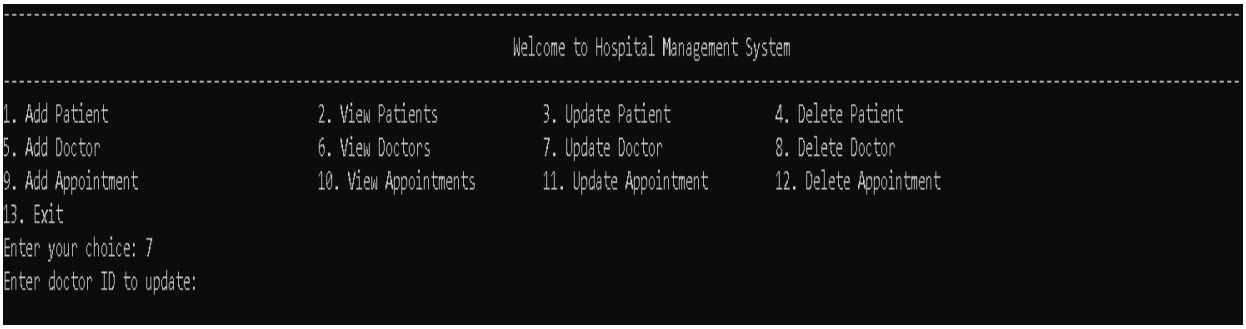
DOCTOR MODULE DETAILS:



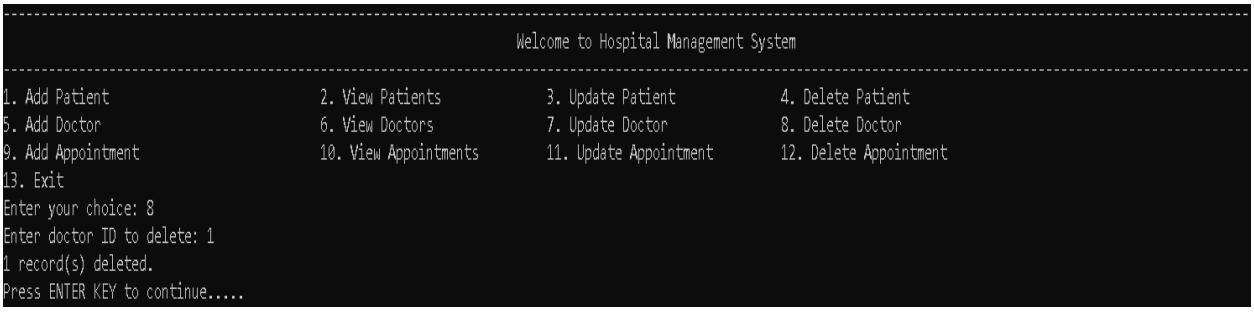
DOCTOR MODULE DETAILS:



DOCTOR MODULE DETAILS:



DOCTOR MODULE DETAILS:



APPOINTMENTS MODULE DETAILS:

```
------------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
------------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 9
Enter patient ID: 1
Enter doctor ID: 1
```

APPOINTMENTS MODULE DETAILS:

```
------------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
------------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 10
Press (f) to see in the form of DataFrame
Press (b) to see in the form of bar graph
Press (l) to see in the form of line graph
Enter your choice:
```

APPOINTMENTS MODULE DETAILS:

```
------------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
------------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 11
Enter appointment ID to update: 1
Enter patient ID: 1
Enter doctor ID: 1
```

APPOINTMENTS MODULE DETAILS:

```
-------------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
-------------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 12
Enter appointment ID to delete: 1
0 record(s) deleted.
Press ENTER KEY to continue.....
```

EXIT MODULE DETAILS:

```
-------------------------------------------------------------------------------------------------------------------------
                                                Welcome to Hospital Management System
-------------------------------------------------------------------------------------------------------------------------
1. Add Patient                  2. View Patients            3. Update Patient           4. Delete Patient
5. Add Doctor                   6. View Doctors             7. Update Doctor            8. Delete Doctor
9. Add Appointment              10. View Appointments       11. Update Appointment      12. Delete Appointment
13. Exit
Enter your choice: 13
Exited !
```

CHARTS

Chart: Patient ID vs Age

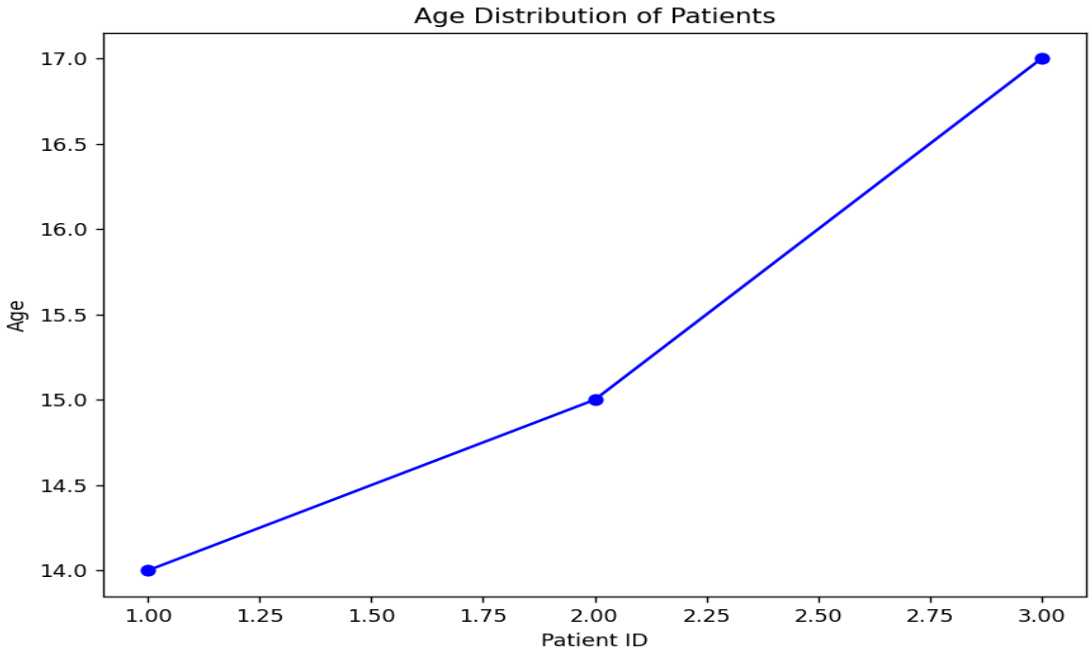


Chart: No.of Patient vs Age

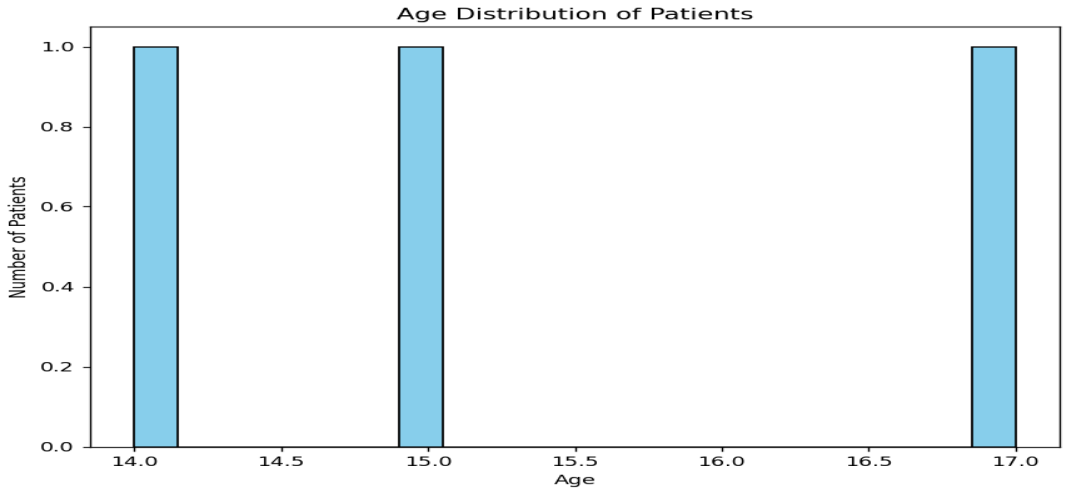


Chart: Doctor Names vs Contact Numbers

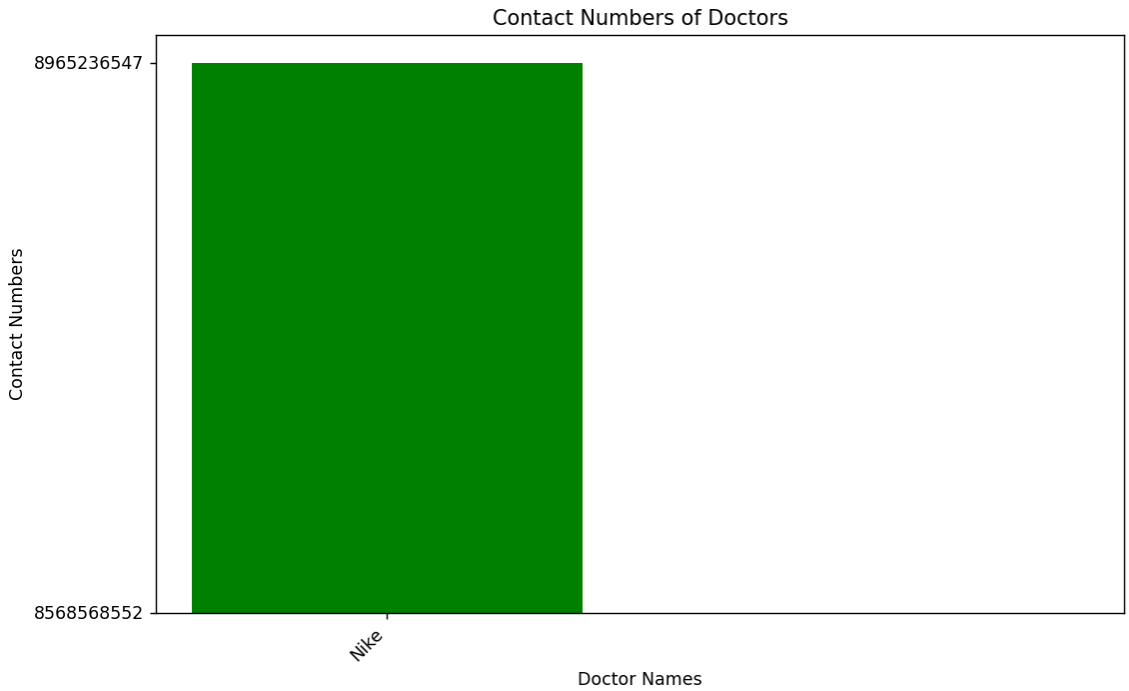


Chart: Specialization vs Doctors

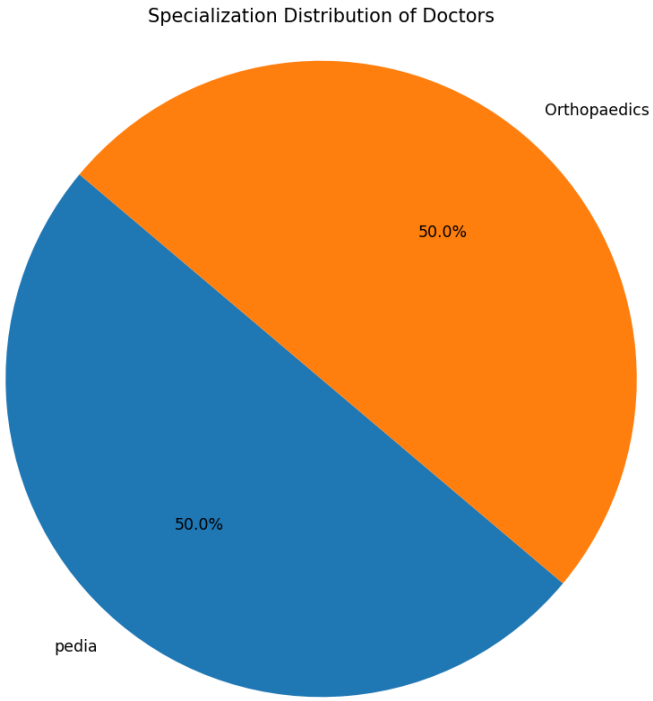


Chart: No.of Appointments vs Appointments Date

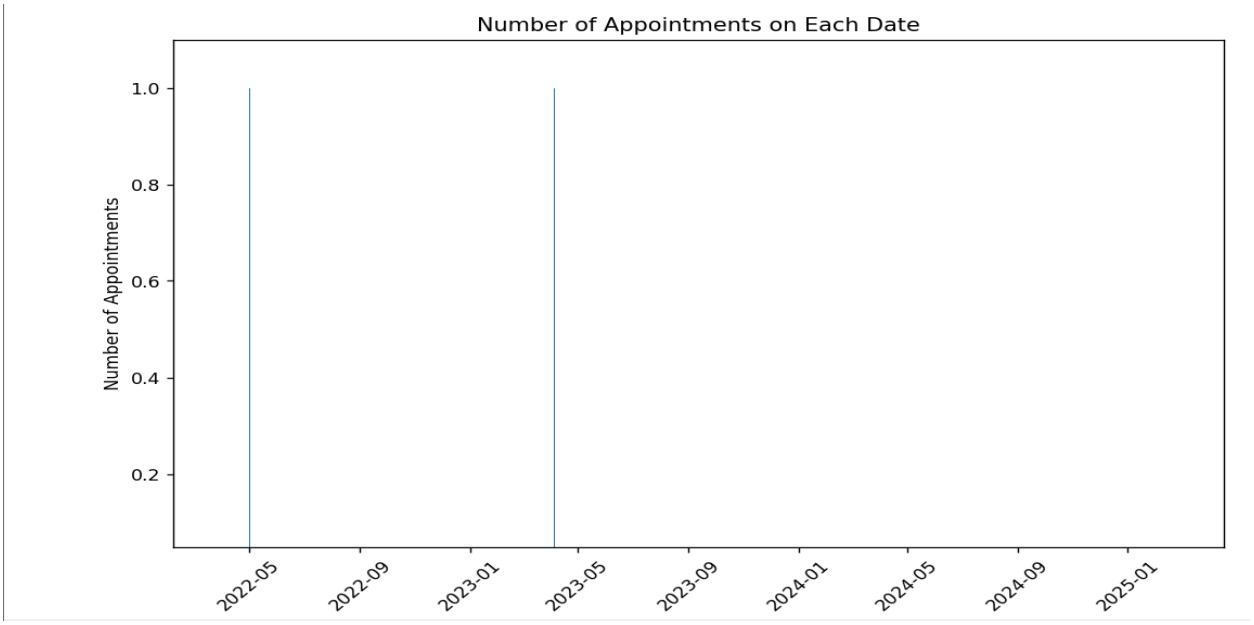
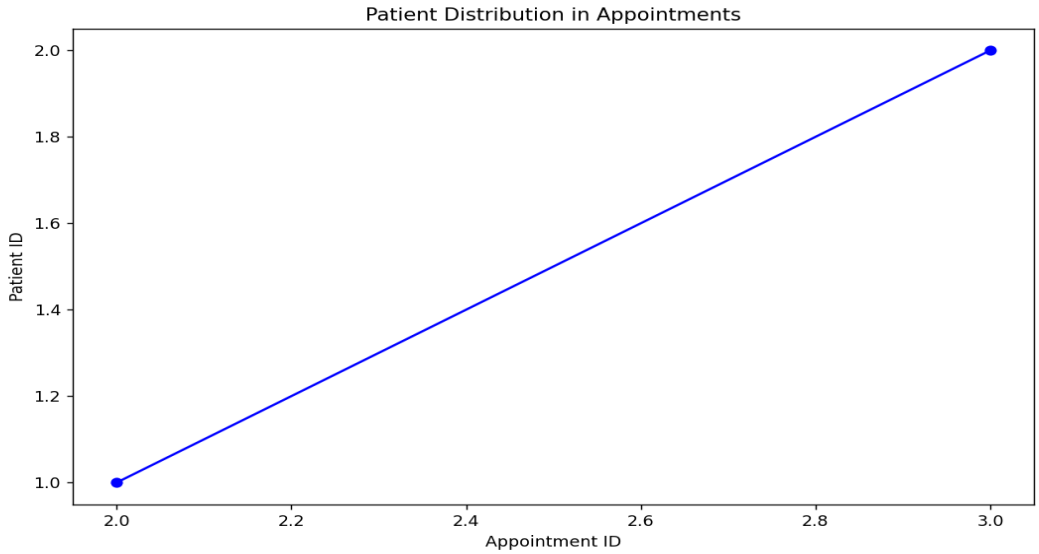


Chart: Appointment ID vs Patient ID



SQL QUERIES
USER OUTPUT(SOURCE CODE)

```
import mysql.connector
import matplotlib.pyplot as plt
import pandas as pd
# Connecting to the MySQL server
mydb = mysql.connector.connect(
    host="localhost",
    user='root',
    password='root'
)
print("-" * 165)
print(" " * 68 + "Welcome to Hospital Management System")
print("-" * 165)

# Creating a database
def create_database():
    cursor = mydb.cursor()
    cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')
    cursor.execute('USE Hosp')

# Creating Patients table
def create_patients_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')

# Creating Doctors table
def create_doctors_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')

# Creating Appointments table
def create_appointments_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO_INCREMENT
PRIMARY KEY, patient_id INT, doctor_id INT, date DATE, time VARCHAR(255))')

# Define the function to add a new patient
def add_patient():
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
```

```
    contact = input("Enter patient contact number: ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"
    val = (name, age, gender, contact)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view patient details
def view_patients():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Patients")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
        df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch=='g':# Adding line chart
        plt.figure(figsize=(8, 6))
        plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')
        plt.xlabel('Patient ID')
        plt.ylabel('Age')
        plt.title('Age Distribution of Patients')
        plt.show()


# Define the function to update patient details
def update_patient():
    patient_id = int(input("Enter patient ID to update: "))
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
    contact = input("Enter patient contact number: ")
    cursor = mydb.cursor()
    sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"
    val = (name, age, gender, contact, patient_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")
```

```
# Define the function to delete patient details
def delete_patient():
    patient_id = int(input("Enter patient ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Patients WHERE Id = %s"
    val = (patient_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new doctor
def add_doctor():
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"
    val = (name, specialization, contact)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view doctor details
def view_doctors():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Doctors")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
import mysql.connector
import matplotlib.pyplot as plt
import pandas as pd
# Connecting to the MySQL server
mydb = mysql.connector.connect(
    host="localhost",
    user='root',
    password='root'
)
print("-" * 165)
print(" " * 68 + "Welcome to Hospital Management System")
print("-" * 165)
```

```
# Creating a database
def create_database():
    cursor = mydb.cursor()
    cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')
    cursor.execute('USE Hosp')

# Creating Patients table
def create_patients_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')

# Creating Doctors table
def create_doctors_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')

# Creating Appointments table
def create_appointments_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO_INCREMENT
PRIMARY KEY, patient_id INT, doctor_id INT, date DATE, time VARCHAR(255))')

# Define the function to add a new patient
def add_patient():
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
    contact = input("Enter patient contact number: ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"
    val = (name, age, gender, contact)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view patient details
def view_patients():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Patients")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
        df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch=='g':# Adding line chart
        plt.figure(figsize=(8, 6))
        plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')
        plt.xlabel('Patient ID')
        plt.ylabel('Age')
        plt.title('Age Distribution of Patients')
        plt.show()


# Define the function to update patient details
def update_patient():
    patient_id = int(input("Enter patient ID to update: "))
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
    contact = input("Enter patient contact number: ")
    cursor = mydb.cursor()
    sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"
    val = (name, age, gender, contact, patient_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete patient details
def delete_patient():
    patient_id = int(input("Enter patient ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Patients WHERE Id = %s"
    val = (patient_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new doctor
def add_doctor():
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"
    val = (name, specialization, contact)
    cursor.execute(sql, val)
    mydb.commit()
```

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

# Define the function to view doctor details
def view_doctors():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Doctors")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
        df = pd.DataFrame({'name': lst1, 'specialization': lst2, 'contact': lst3})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch == 'g':
        # Extracting data for the pie chart
        specializations = [row[2] for row in result]
        unique_specializations = list(set(specializations))
        specialization_counts = [specializations.count(spec) for spec in unique_specializations]
        # Plotting the pie chart
        plt.figure(figsize=(8, 8))
        plt.pie(specialization_counts, labels=unique_specializations, autopct='%1.1f%%', startangle=140)
        plt.title('Specialization Distribution of Doctors')
        plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
        plt.show()


# Define the function to update doctor details
def update_doctor():
    doctor_id = int(input("Enter doctor ID to update: "))
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
    cursor = mydb.cursor()
    sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"
    val = (name, specialization, contact, doctor_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete doctor details
def delete_doctor():
    doctor_id = int(input("Enter doctor ID to delete: "))
```

```
    cursor = mydb.cursor()
    sql = "DELETE FROM Doctors WHERE Id = %s"
    val = (doctor_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new appointment
def add_appointment():
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Appointments (patient_id, doctor_id, date, time) VALUES (%s, %s, %s, %s)"
    val = (patient_id, doctor_id, date, time)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view appointment details
def view_appointments():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Appointments")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")#patient_id, doctor_id, date, time
    if 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]
        df = pd.DataFrame({'patient_id': lst1, 'doctor_id': lst2, 'date': lst3, 'time': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch == 'g':
        # Extracting dates from the 'date' column
        dates = [row[3] for row in result]

        # Counting the occurrences of each date
        date_counts = {}
        for date in dates:
            date_counts[date] = date_counts.get(date, 0) + 1

        # Sorting the dates for better visualization
```

```
        sorted_dates = sorted(date_counts.keys())

        # Creating a bar chart
        plt.figure(figsize=(10, 6))
        plt.bar(sorted_dates, [date_counts[date] for date in sorted_dates])
        plt.xlabel('Date')
        plt.ylabel('Number of Appointments')
        plt.title('Number of Appointments on Each Date')
        plt.xticks(rotation=45)
        plt.show()

# Define the function to update appointment details
def update_appointment():
    appointment_id = int(input("Enter appointment ID to update: "))
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "UPDATE Appointments SET patient_id = %s, doctor_id = %s, date = %s, time = %s WHERE
Id = %s"
    val = (patient_id, doctor_id, date, time, appointment_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete appointment details
def delete_appointment():
    appointment_id = int(input("Enter appointment ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Appointments WHERE Id = %s"
    val = (appointment_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Get the user's choice
def getchoice():
    while True:
        create_database()
        create_patients_table()
        create_doctors_table()
        create_appointments_table()

        print("1. Add Patient                2. View Patients          3. Update Patient          4. Delete
Patient")
        print("5. Add Doctor                 6. View Doctors           7. Update Doctor           8. Delete
Doctor")
        print("9. Add Appointment                10. View Appointments        11. Update Appointment
12. Delete Appointment")
```

```
        print("13. Exit")
        opp = input("Enter your choice: ")
        if opp == '1':
            add_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '2':
            view_patients()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '3':
            update_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '4':
            delete_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '5':
            add_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '6':
            view_doctors()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '7':
            update_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '8':
            delete_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '9':
            add_appointment()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '10':
            view_appointments()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '11':
            update_appointment()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '12':
            delete_appointment()
            input("Press ENTER KEY to continue.....")
            print()
```

```
        elif opp == '13':
            print('Exited !')
            break

# Recall Choice function
getchoice()
# Disconnecting from the server
mydb.close()import mysql.connector
import matplotlib.pyplot as plt
import pandas as pd
# Connecting to the MySQL server
mydb = mysql.connector.connect(
    host="localhost",
    user='root',
    password='root'
)
print("-" * 165)
print(" " * 68 + "Welcome to Hospital Management System")
print("-" * 165)

# Creating a database
def create_database():
    cursor = mydb.cursor()
    cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')
    cursor.execute('USE Hosp')

# Creating Patients table
def create_patients_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')

# Creating Doctors table
def create_doctors_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO_INCREMENT
PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')

# Creating Appointments table
def create_appointments_table():
    cursor = mydb.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO_INCREMENT
PRIMARY KEY, patient_id INT, doctor_id INT, date DATE, time VARCHAR(255))')

# Define the function to add a new patient
def add_patient():
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
    contact = input("Enter patient contact number: ")
```

```
    cursor = mydb.cursor()
    sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"
    val = (name, age, gender, contact)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view patient details
def view_patients():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Patients")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
        df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch=='g':# Adding line chart
        plt.figure(figsize=(8, 6))
        plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')
        plt.xlabel('Patient ID')
        plt.ylabel('Age')
        plt.title('Age Distribution of Patients')
        plt.show()


# Define the function to update patient details
def update_patient():
    patient_id = int(input("Enter patient ID to update: "))
    name = input("Enter patient name: ")
    age = int(input("Enter patient age: "))
    gender = input("Enter patient gender: ")
    contact = input("Enter patient contact number: ")
    cursor = mydb.cursor()
    sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"
    val = (name, age, gender, contact, patient_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")
```

```
# Define the function to delete patient details
def delete_patient():
    patient_id = int(input("Enter patient ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Patients WHERE Id = %s"
    val = (patient_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new doctor
def add_doctor():
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"
    val = (name, specialization, contact)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view doctor details
def view_doctors():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Doctors")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")
    if 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]
        df = pd.DataFrame({'name': lst1, 'specialization': lst2, 'contact': lst3})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch == 'g':
        # Extracting data for the pie chart
        specializations = [row[2] for row in result]
        unique_specializations = list(set(specializations))
        specialization_counts = [specializations.count(spec) for spec in unique_specializations]
        # Plotting the pie chart
        plt.figure(figsize=(8, 8))
        plt.pie(specialization_counts, labels=unique_specializations, autopct='%1.1f%%', startangle=140)
        plt.title('Specialization Distribution of Doctors')
```

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


# Define the function to update doctor details
def update_doctor():
    doctor_id = int(input("Enter doctor ID to update: "))
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
    cursor = mydb.cursor()
    sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"
    val = (name, specialization, contact, doctor_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete doctor details
def delete_doctor():
    doctor_id = int(input("Enter doctor ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Doctors WHERE Id = %s"
    val = (doctor_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new appointment
def add_appointment():
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Appointments (patient_id, doctor_id, date, time) VALUES (%s, %s, %s, %s)"
    val = (patient_id, doctor_id, date, time)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view appointment details
def view_appointments():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Appointments")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")#patient_id, doctor_id, date, time
    if 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]
        df = pd.DataFrame({'patient_id': lst1, 'doctor_id': lst2, 'date': lst3, 'time': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch == 'g':
        # Extracting dates from the 'date' column
        dates = [row[3] for row in result]

        # Counting the occurrences of each date
        date_counts = {}
        for date in dates:
            date_counts[date] = date_counts.get(date, 0) + 1

        # Sorting the dates for better visualization
        sorted_dates = sorted(date_counts.keys())

        # Creating a bar chart
        plt.figure(figsize=(10, 6))
        plt.bar(sorted_dates, [date_counts[date] for date in sorted_dates])
        plt.xlabel('Date')
        plt.ylabel('Number of Appointments')
        plt.title('Number of Appointments on Each Date')
        plt.xticks(rotation=45)
        plt.show()

# Define the function to update appointment details
def update_appointment():
    appointment_id = int(input("Enter appointment ID to update: "))
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "UPDATE Appointments SET patient_id = %s, doctor_id = %s, date = %s, time = %s WHERE
Id = %s"
    val = (patient_id, doctor_id, date, time, appointment_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete appointment details
def delete_appointment():
    appointment_id = int(input("Enter appointment ID to delete: "))
    cursor = mydb.cursor()
```

```
    sql = "DELETE FROM Appointments WHERE Id = %s"
    val = (appointment_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Get the user's choice
def getchoice():
    while True:
        create_database()
        create_patients_table()
        create_doctors_table()
        create_appointments_table()

        print("1. Add Patient                2. View Patients        3. Update Patient        4. Delete
Patient")
        print("5. Add Doctor                 6. View Doctors         7. Update Doctor         8. Delete
Doctor")
        print("9. Add Appointment            10. View Appointments   11. Update Appointment
12. Delete Appointment")
        print("13. Exit")
        opp = input("Enter your choice: ")
        if opp == '1':
            add_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '2':
            view_patients()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '3':
            update_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '4':
            delete_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '5':
            add_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '6':
            view_doctors()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '7':
            update_doctor()
            input("Press ENTER KEY to continue.....")
            print()
```

```
        elif opp == '8':
            delete_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '9':
            add_appointment()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '10':
            view_appointments()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '11':
            update_appointment()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '12':
            delete_appointment()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '13':
            print('Exited !')
            break

# Recall Choice function
getchoice()
# Disconnecting from the server
mydb.close()'specialization': lst2, 'contact': lst3})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
    elif ch == 'g':
        # Extracting data for the pie chart
        specializations = [row[2] for row in result]
        unique_specializations = list(set(specializations))
        specialization_counts = [specializations.count(spec) for spec in unique_specializations]
        # Plotting the pie chart
        plt.figure(figsize=(8, 8))
        plt.pie(specialization_counts, labels=unique_specializations, autopct='%1.1f%%', startangle=140)
        plt.title('Specialization Distribution of Doctors')
        plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
        plt.show()

# Define the function to update doctor details
def update_doctor():
    doctor_id = int(input("Enter doctor ID to update: "))
    name = input("Enter doctor name: ")
    specialization = input("Enter doctor specialization: ")
    contact = input("Enter doctor contact number: ")
```

```
    cursor = mydb.cursor()
    sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"


    val = (name, specialization, contact, doctor_id)
  cursor.execute(sql, val)  mydb.commit()    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete doctor details
def delete_doctor():
    doctor_id = int(input("Enter doctor ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Doctors WHERE Id = %s"
    val = (doctor_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Define the function to add a new appointment
def add_appointment():
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "INSERT INTO Appointments (patient_id, doctor_id, date, time) VALUES (%s, %s, %s, %s)"
    val = (patient_id, doctor_id, date, time)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) inserted.")

# Define the function to view appointment details
def view_appointments():
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM Appointments")
    result = cursor.fetchall()
    print("Press (l) to see in the form of list")
    print("Press (f) to see in the form of DataFrame")
    print("Press (g) to see in the form of graph")
    ch = input("Enter your choice: ")#patient_id, doctor_id, date, time
    if 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]
        df = pd.DataFrame({'patient_id': lst1, 'doctor_id': lst2, 'date': lst3, 'time': lst4})
        print(df.to_markdown())
    elif ch=='l':
        result_list = [list(row) for row in result]
        print(result_list)
```

```
    elif ch == 'g':
        # Extracting dates from the 'date' column
        dates = [row[3] for row in result]

        # Counting the occurrences of each date
        date_counts = {}
        for date in dates:
            date_counts[date] = date_counts.get(date, 0) + 1

        # Sorting the dates for better visualization
        sorted_dates = sorted(date_counts.keys())

        # Creating a bar chart
        plt.figure(figsize=(10, 6))
        plt.bar(sorted_dates, [date_counts[date] for date in sorted_dates])
        plt.xlabel('Date')
        plt.ylabel('Number of Appointments')
        plt.title('Number of Appointments on Each Date')
        plt.xticks(rotation=45)
        plt.show()

# Define the function to update appointment details
def update_appointment():
    appointment_id = int(input("Enter appointment ID to update: "))
    patient_id = int(input("Enter patient ID: "))
    doctor_id = int(input("Enter doctor ID: "))
    date = input("Enter appointment date (YYYY-MM-DD): ")
    time = input("Enter appointment time (HH:MM AM/PM): ")
    cursor = mydb.cursor()
    sql = "UPDATE Appointments SET patient_id = %s, doctor_id = %s, date = %s, time = %s WHERE
Id = %s"
    val = (patient_id, doctor_id, date, time, appointment_id)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) updated.")

# Define the function to delete appointment details
def delete_appointment():
    appointment_id = int(input("Enter appointment ID to delete: "))
    cursor = mydb.cursor()
    sql = "DELETE FROM Appointments WHERE Id = %s"
    val = (appointment_id,)
    cursor.execute(sql, val)
    mydb.commit()
    print(cursor.rowcount, "record(s) deleted.")

# Get the user's choice
def getchoice():
    while True:
        create_database()
```

```
        create_patients_table()
        create_doctors_table()
        create_appointments_table()
 print("1. Add Patient        2. View Patients
         3. Update Patient      4. Delete Patient")
print("5. Add Doctor          6. View Doctor
         7. Update Doctor        8. Delete Doctor")

print("9. Add Appointment   10. View Appointments          11. Update Appointment          12. Delete
Appointment")
        print("13. Exit")
        opp = input("Enter your choice: ")
        if opp == '1':
            add_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '2':
            view_patients()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '3':
            update_patient()
            input("Press ENTER KEY to continue.....")
            print()

        elif opp == '4':
            delete_patient()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '5':
            add_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '6':
            view_doctors()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '7':
            update_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '8':
            delete_doctor()
            input("Press ENTER KEY to continue.....")
            print()
        elif opp == '9':
            add_appointment()
            input("Press ENTER KEY to continue.....")
            print()
```

```
    elif opp == '10':
        view_appointments()
        input("Press ENTER KEY to continue.....")
        print()
    elif opp == '11':
        update_appointment()
        input("Press ENTER KEY to continue.....")
        print()
    elif opp == '12':
        delete_appointment()
        input("Press ENTER KEY to continue.....")
        print()
    elif opp == '13':
        print('Exited !')
        break

# Recall Choice function
getchoice()

# Disconnecting from the server
mydb.close()
```

CONCLUSION

In conclusion, the provided Python code implements a Hospital Management System utilizing a MySQL database for storing information about patients, doctors, and appointments. The system allows users to perform various operations such as adding, viewing, updating, and deleting records related to patients, doctors, and appointments.

The code establishes a connection to a MySQL server and creates a database named "Hosp" if it doesn't exist. Tables for Patients, Doctors, and Appointments are created to organize and store relevant information.Users can add, view, update, and delete patient records.

Additionally, the code provides options to display patient information in different formats such as a DataFrame, line chart, or histogram.Similar to patient operations, users can add, view, update, and delete doctor records.The code offers visualization options, including DataFrame, bar chart, and pie chart representations of doctor details.

Users can add, view, update, and delete appointment records.Visualization choices include a DataFrame view and graphs like line charts and bar charts to represent appointment-related data.The code provides a user-friendly interface where users can choose from different operations through a menu. Matplotlib and Pandas are employed for data visualization.

For instance, patient ages can be displayed using a line chart or a histogram, and doctor details can be visualized using bar charts or pie charts. The program runs in a loop, allowing users to perform multiple operations without restarting the application. The code is modular, with separate functions for database creation, table creation, data manipulation, and user interaction. This modular structure enhances readability and maintainability.

Users can exit the program at any point, and the connection to the MySQL server is properly closed. Overall, the code provides a functional and interactive platform for managing hospital-related information with added features for visualizing data in different formats.

Future Scope of Project

Some suggestions for the “Future Scope of Project”:

1.Integration with Online Platforms:Explore the possibility of integration the "Hospital Management" 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(Hospital Management System).

BIBLOGRAPHY

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