Hospital Database Operations Report

Aman Bhattarai

2024-09-17

Table of contents

1 Introduction

This report demonstrates various operations performed on a hospital database using Python and SQLite. The operations include connecting to the database, fetching information, updating records, and closing the connection.

2 Database Operations Overview

1. Connecting to the database
2. Fetching hospital and doctor information

3. Retrieving doctors by specialty and salary
4. Listing doctors from a specific hospital
5. Updating doctor experience
6. Visualizing doctor data
7. Closing and verifying the database connection

Warning

Important: Ensure that the `HospitalInfo.db` file is in the same directory as this script before running.

2.1 Setup and Database Connection

First, we import the necessary libraries and set up our database connection.

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt

# Set pandas display options
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', 20)
```

Now, let's define our function to connect to the database:

```
def connect_to_database(db_file):
    """Connects to the SQLite database and prints version and table information."""
    try:
        sqlconnection = sqlite3.connect(db_file)
        print("Database connected successfully")
        version_query = "SELECT sqlite_version();"
        version = pd.read_sql_query(version_query, sqlconnection)
        print("SQLite version:", version.iloc[0, 0])

        cursor = sqlconnection.cursor()
        cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
        tables = cursor.fetchall()

        print("\nTables in the database:")
        for table in tables:
            table_name = table[0]
            print(f"\n--- Table: {table_name} ---")
            cursor.execute(f"PRAGMA table_info({table_name})")
            columns = cursor.fetchall()
            df = pd.DataFrame(columns, columns=['cid', 'name', 'type', 'notnull', 'dflt_va
            print(df[['name', 'type']])

            print(f"\nContents of {table_name}:")
            df = pd.read_sql_query(f"SELECT * FROM {table_name}", sqlconnection)
            print(df)
            print("\n")
        return sqlconnection

    except sqlite3.Error as error:
        print(f"Error occurred: {error}")
        return None
```

2.2 Fetching Hospital and Doctor Information

This function retrieves information about a specific hospital and doctor:

```
def fetch_hospital_doctor_info(sqlconnection):
    """Fetches and prints hospital and doctor information based on user input."""
    try:
        hospital_id = 104  # For demonstration, using fixed values
        query_hospital = "SELECT * FROM Hospital WHERE ID = ?"
        doctor_id = 4    # For demonstration, using fixed values
        query_doctor = "SELECT * FROM Doctor WHERE DocID = ?"

        df_hospital = pd.read_sql_query(query_hospital, sqlconnection, params=(hospital_id
        df_doctor = pd.read_sql_query(query_doctor, sqlconnection, params=(doctor_id,))

        if not df_hospital.empty:
            print(f"\nFetching Hospital information for ID: {hospital_id}")
            print(df_hospital)
        else:
            print(f"No matching records found for Hospital with ID {hospital_id}.")

        if not df_doctor.empty:
            print(f"\nFetching Doctor information for ID: {doctor_id}")
            print(df_doctor)
        else:
            print(f"No matching records found for Doctor with ID {doctor_id}.")

    except pd.io.sql.DatabaseError as error:
        print(f"Error occurred: {error}")
```

2.3 Retrieving Doctors by Specialty and Salary

This function demonstrates how to filter doctors based on their specialty and salary:

```
def get_doctors_by_speciality_salary(sqlconnection):
    """Gets and prints doctors based on the given speciality and salary."""
    try:
        speciality = "Cardialogist"  # For demonstration, using fixed values
        salary = 1000000.0          # For demonstration, using fixed values
        query_doctor = "SELECT * FROM Doctor WHERE LOWER(Speciality) = ? and salary = ?"

        df_doctor = pd.read_sql_query(query_doctor, sqlconnection, params=(speciality, sal

        if not df_doctor.empty:
            print(f"\nFetching Doctor information for speciality `{speciality}` and salary
            print(df_doctor)
        else:
            print(f"\nNo matching records found for Doctor with speciality `{speciality}`

    except pd.io.sql.DatabaseError as error:
        print(f"Error occurred: {error}")
```

2.4 Listing Doctors from a Specific Hospital

This function retrieves all doctors working in a given hospital:

```
def get_doctors_by_hospital(sqlconnection):
    """Gets and prints a list of doctors from a given hospital ID."""
    try:
        hospital_id = 101  # For demonstration, using fixed values
        query = """
        SELECT D.*
        FROM Doctor D
        JOIN Hospital H ON D.HospitalID = H.ID
        WHERE H.ID = ?
        """

        df = pd.read_sql_query(query, sqlconnection, params=(hospital_id,))

        if not df.empty:
            print(f"\nFetching Doctor information for hospital with id `{hospital_id}`\n")
            print(df)
        else:
            print(f"\nNo matching records found for Doctor.\n")

    except pd.io.sql.DatabaseError as error:
        print(f"Error occurred: {error}")
```

2.5 Updating Doctor Experience

This function demonstrates how to update a doctor's experience in the database:

```
def update_doctor_experience(sqlconnection):
    """Updates the experience of a doctor based on the given Doctor ID."""
    try:
        cursor = sqlconnection.cursor()
        doctor_id = 1  # For demonstration, using fixed values
        new_experience = "10 years"  # For demonstration, using fixed values

        find_doctor = "SELECT * FROM Doctor WHERE DocID = ?"
        cursor.execute(find_doctor, (doctor_id,))
        result = cursor.fetchone()

        if result:
            print("Before update:", result)
            update_query = "UPDATE Doctor SET Exp = ? WHERE DocID = ?"
            cursor.execute(update_query, (new_experience, doctor_id))
            sqlconnection.commit()
            print(f"\nExperience updated for Doctor ID {doctor_id}")
            cursor.execute(find_doctor, (doctor_id,))
            print("After update:", cursor.fetchone())
        else:
            print(f"No doctor found with ID {doctor_id}")

    except sqlite3.Error as error:
        print(f"Error occurred: {error}")
```

2.6 Visualizing Doctor Data

This function creates visualizations of the doctor data:

```
def visualize_doctor_data(sqlconnection):
    """Creates visualizations of doctor data."""
    try:
        # Fetch all doctor data
        df_doctors = pd.read_sql_query("SELECT * FROM Doctor", sqlconnection)

        # Visualization 1: Bar chart of doctors per specialty
        specialty_counts = df_doctors['Speciality'].value_counts()
        plt.figure(figsize=(10, 6))
        specialty_counts.plot(kind='bar')
        plt.title('Number of Doctors per Specialty')
        plt.xlabel('Specialty')
        plt.ylabel('Number of Doctors')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # Visualization 2: Scatter plot of salary vs experience
        plt.figure(figsize=(10, 6))
        plt.scatter(df_doctors['Exp'], df_doctors['salary'])
        plt.title('Salary vs Experience')
        plt.xlabel('Experience (years)')
        plt.ylabel('Salary')
        plt.tight_layout()
        plt.show()

    except pd.io.sql.DatabaseError as error:
        print(f"Error occurred: {error}")
```

2.7 Closing and Verifying the Connection

Finally, this function closes the database connection and verifies that it’s closed:

```python
def close_and_verify_connection(sqlconnection):
    """Closes the database connection and verifies if it's closed."""
    if sqlconnection:
        sqlconnection.close()
        print("\nDatabase connection closed.")

    try:
        sqlconnection.execute("SELECT 1")
        print("The connection is still open.")
    except:
        print("The connection is closed.")
```

2.8 Main Program

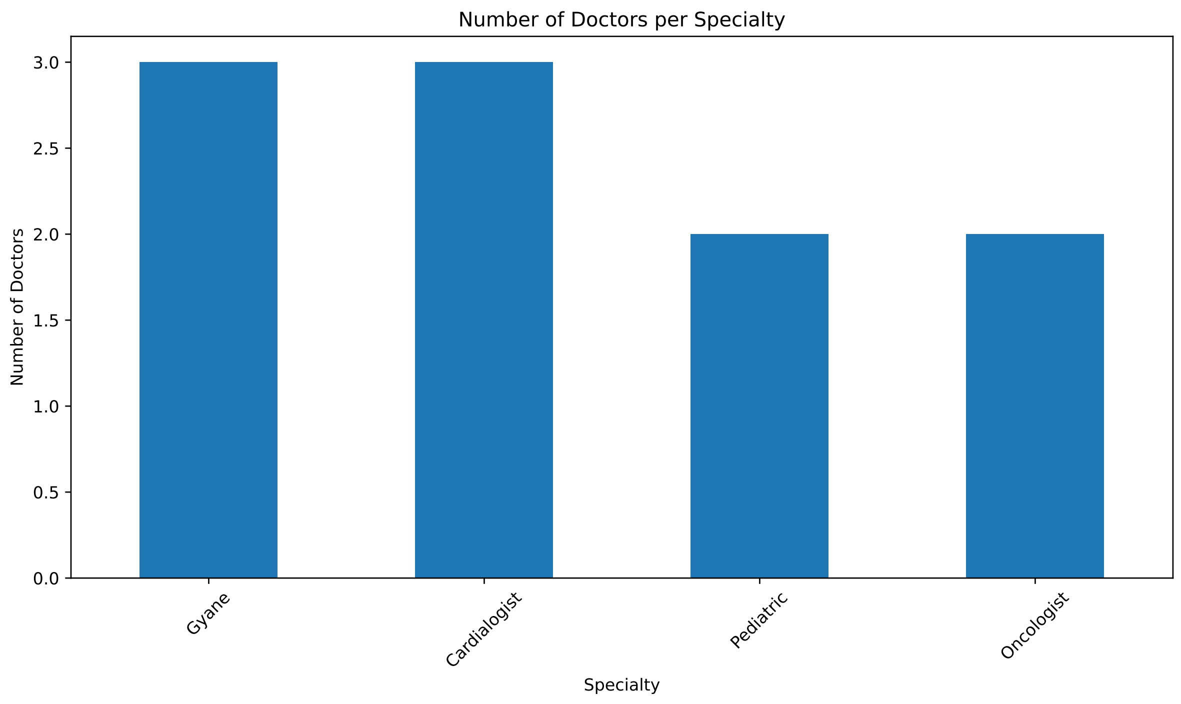
The main program that executes all the above functions:

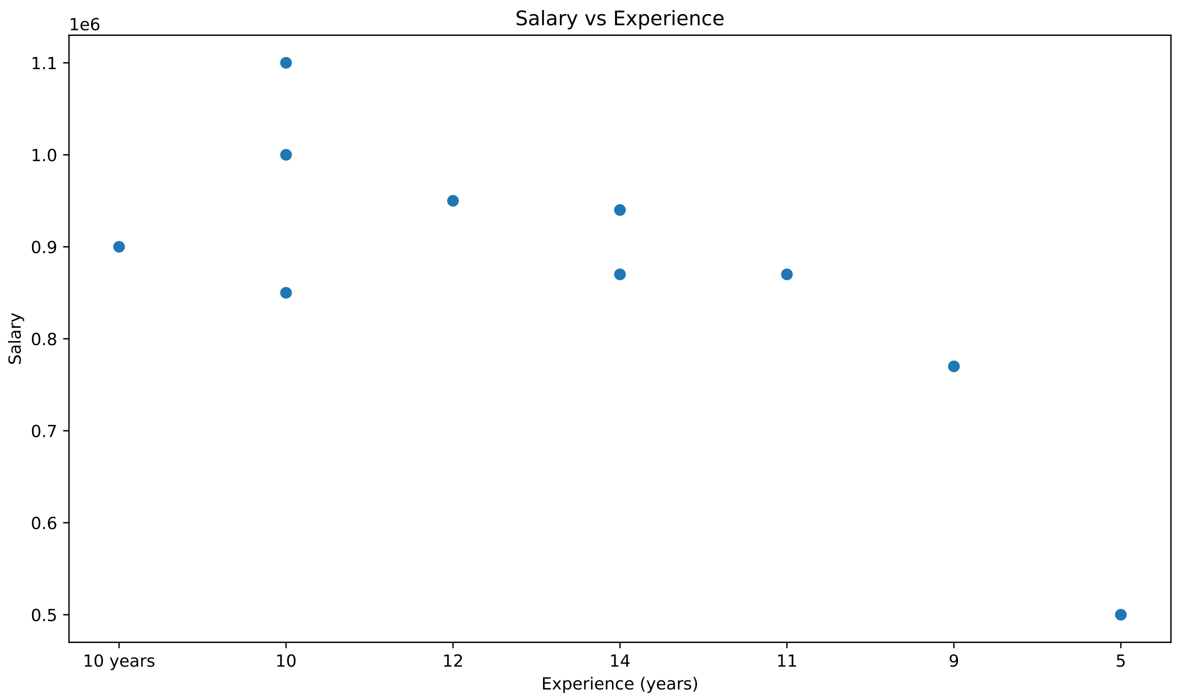
```python
def main():
    connection = connect_to_database("HospitalInfo.db")
    if connection:
        fetch_hospital_doctor_info(connection)
        get_doctors_by_speciality_salary(connection)
        get_doctors_by_hospital(connection)
        update_doctor_experience(connection)
        visualize_doctor_data(connection)
        close_and_verify_connection(connection)

if __name__ == "__main__":
    main()
```

```
Database connected successfully
SQLite version: 3.43.2

Tables in the database:

--- Table: Hospital ---
       name     type
0        ID  INTEGER
```

```
1      Name     TEXT
2  BedCount  INTEGER

Contents of Hospital:
    ID                 Name  BedCount
0  101          Mayo Clinic       230
1  102          JP Hoplkins       130
2  103        New Amsterdam       200
3  104      Clevland Clinic        30
4  105      Toronto Hopital       180
5  106               Natura       150
6  107  Johnson and Johnson       600



--- Table: Doctor ---
          name     type
0        DocID  INTEGER
1      DocName     TEXT
2   HospitalID  INTEGER
3  JoiningDate     BLOB
4   Speciality     TEXT
5          Exp     TEXT
6       salary     REAL

Contents of Doctor:
   DocID  DocName  HospitalID JoiningDate    Speciality       Exp     salary
0      1  Michael         101  2005-02-10     Pediatric  10 years   900000.0
1      2    Linda         101  2007-08-08         Gyane        10   850000.0
2      3  William         102  2004-09-11  Cardialogist        10  1000000.0
3      4  Richard         101  2011-09-05     Pediatric        12   950000.0
4      5    Karen         103  2020-09-05    Oncologist        10  1100000.0
5      6   Robert         104  1998-09-04         Gyane        14   940000.0
6      7    Susan         105  1994-06-05    Oncologist        11   870000.0
7      8    Nancy         106  1994-06-05  Cardialogist        14   870000.0
8      9     Nick         107  2019-06-05  Cardialogist         9   770000.0
9     10     Adam         110        None         Gyane         5   500000.0



Fetching Hospital information for ID: 104
    ID             Name  BedCount
0  104  Clevland Clinic        30
```

```
Fetching Doctor information for ID: 4
   DocID  DocName  HospitalID JoiningDate Speciality Exp    salary
0      4  Richard         101  2011-09-05  Pediatric  12  950000.0

No matching records found for Doctor with speciality `Cardialogist` and salary `1000000.0`.


Fetching Doctor information for hospital with id `101`

   DocID  DocName  HospitalID JoiningDate Speciality       Exp    salary
0      1  Michael         101  2005-02-10  Pediatric  10 years  900000.0
1      2    Linda         101  2007-08-08      Gyane        10  850000.0
2      4  Richard         101  2011-09-05  Pediatric        12  950000.0
Before update: (1, 'Michael', 101, '2005-02-10', 'Pediatric', '10 years', 900000.0)

Experience updated for Doctor ID 1
After update: (1, 'Michael', 101, '2005-02-10', 'Pediatric', '10 years', 900000.0)
```





```
Database connection closed.
The connection is closed.
```

3 Conclusion

This report demonstrates the power and flexibility of using Python with SQLite for managing healthcare data. The operations covered range from basic queries to data updates and visualizations, showcasing how such a system can support various aspects of hospital management and decision-making processes.

4 References

SQLite Documentation. https://www.sqlite.org/docs.html
Python SQLite3 Module. https://docs.python.org/3/library/sqlite3.html
Pandas Documentation. https://pandas.pydata.org/docs/
Matplotlib Documentation. https://matplotlib.org/stable/contents.html