ROOM AND SERVICE BOOKING APP

NAME : SHASHANK BELLAD SRN : PES1UG22AM150
NAME : VIGNESH MADIVALA SRN : PES1UG22AM190

Description of the Statement :

The "Room and Services Booking App" is an integrated hotel management system designed to handle guest and room management, reservations, staff details, services, payments, and guest feedback. It features a MySQL database with triggers and stored procedures to automate room availability updates and enforce privilege-based user role management. The frontend, built using Streamlit, provides a userfriendly interface for data management and privilege-based access control.

User Requirements:

The Hotel Management System allows the users to book hotel rooms and other services and helps the Hotel to manage their rooms, services, bookings, etc.

1. The system should be Realtime
2. The system must provide fast time response to the user actions.
2. The system should support large number of guests, reservations and bookings.
3. Should support validation like value constraints.
4. The Staff members need to be given permissions based on their positions.
5. The system should support multivalued fields.
6. The system should be indexed for fast access.
7. Payment and guest data should be securely stored.

<u>Guest Management:</u>

Guests can create accounts with their personal information, such as name, age, phone number.
Guest can manage their reservations.

Room Management:

The Hotel Management manages the room information such as room type, availability, prices, etc. which helps in the reservation of rooms.

Staf f Management:

The Hotel can manage staff information such as name, age, salary, position.

Roles and permissions can be given to the staff based on their positions.

Reservation Management:

Guest can make reservations for the rooms.

Staff allows for guest reservations by checking the availability and can modify or cancel reservations.

Services Management:

The staff manages the various services provided by the Hotel like room service, dining service, gym, sports, etc.

Bookings Management:

The Staff can manage the guest booking for various services.

Payment Management:

Guest will make payments for the reservations and services and hotel can manage all the transactions made by the guest such as the

paymentID, payment method, date.

List of Software/Tools/Programming Languages Used

Database

- MySQL:
  Used for the backend database to manage structured data.

Programming Languages

- Python:
  Used for developing the Streamlit-based frontend.
- SQL:
  Used for creating the database schema, triggers, stored procedures, and executing queries.

Frontend Framework

- Streamlit:
  Lightweight web framework for building the user interface.

Libraries and Tools

- pymysql:
  Python library for connecting and executing SQL queries in MySQL.
- pandas:
  For processing and displaying database results in tabular form.

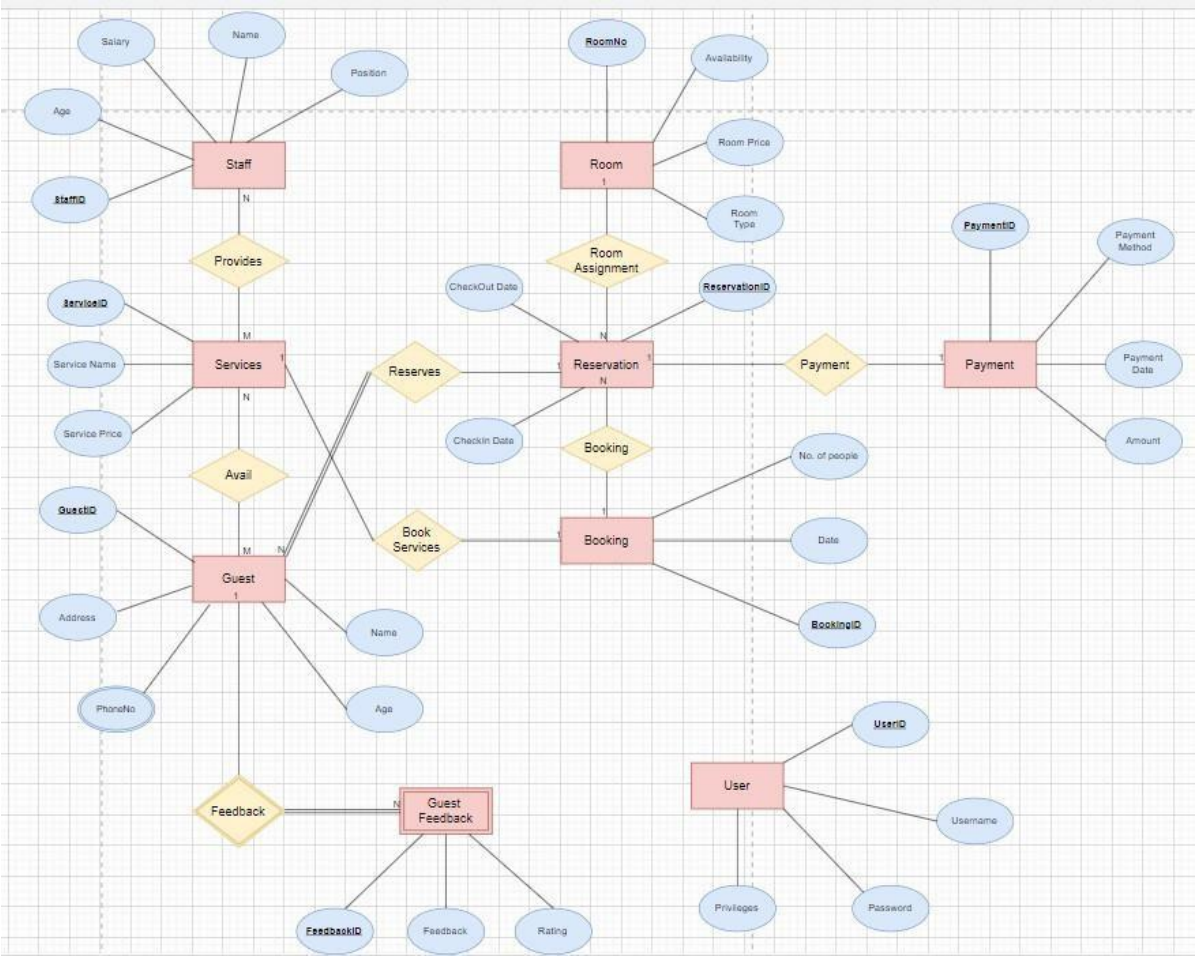
Development Environment

- Any Python IDE (e.g., PyCharm, VS Code)
- MySQL Workbench (or equivalent) for database management

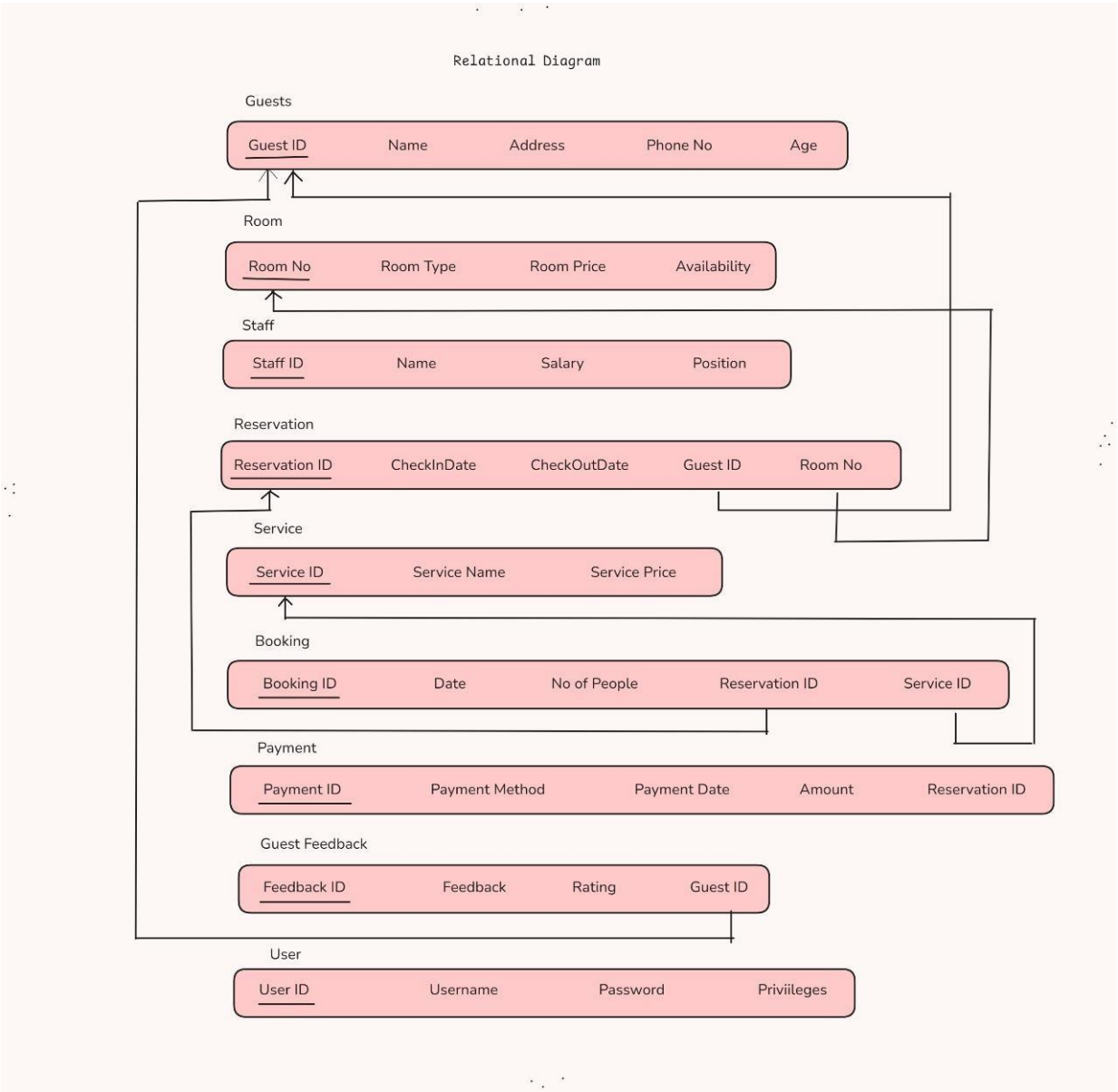
Other Tools

- Browser for running Streamlit apps.
- Command-line tools for Streamlit and Python setup.

ER DIAGRAM:


Salary
Name
Position
Age
Staff
StaffID
N
Provides
M
ServiceID
Service Name
Services
N
Service Price
Avail
GuestID
M
N
Guest
Address
PhoneNo
Name
Age
Feedback
N
Guest Feedback
FeedbackID
Feedback
Rating
RoomNo
Availability
Room Price
Room
Room Type
Room Assignment
CheckOut Date
ReservationID
N
Reserves
Reservation
N
Payment
Payment
PaymentID
Payment Method
Payment Date
Amount
Checkin Date
Booking
No. of people
Book Services
Booking
Date
BookingID
User
UserID
Username
Privileges
Password

Relational Schema:



DDL COMANDS:

CREATE DATABASE

```
CREATE DATABASE Hotel_Management;
USE Hotel_Management;
```

CREATE TABLE

```
CREATE TABLE Guest (
    GuestID VARCHAR(20) PRIMARY KEY,
    Name VARCHAR(50),
    Address VARCHAR(100),
    Phone VARCHAR(15),
    Age INT
);
```

Similar commands are used to create the following tables:

Guest
Room
Reservation
Staff
Service
Booking
Payments
GuestFeedback
User

```sql
CREATE TABLE Room(
    RoomNumber INT PRIMARY KEY,
    RoomType VARCHAR(50),
    Price DECIMAL(10, 2),
    Availability ENUM('Available', 'Occupied', 'Under Maintenance') NOT NULL
);

CREATE TABLE Reservation(
    ReservationID VARCHAR(20) PRIMARY KEY,
    CheckInDate DATE,
    CheckOutDate DATE,
    GuestID VARCHAR(20),
    RoomNumber INT,
    FOREIGN KEY (GuestID) REFERENCES Guest(GuestID) ON DELETE CASCADE,
    FOREIGN KEY (RoomNumber) REFERENCES Room(RoomNumber)
);
```

ALTER TABLE

```sql
ALTER TABLE Guest
DROP INDEX phone;
```

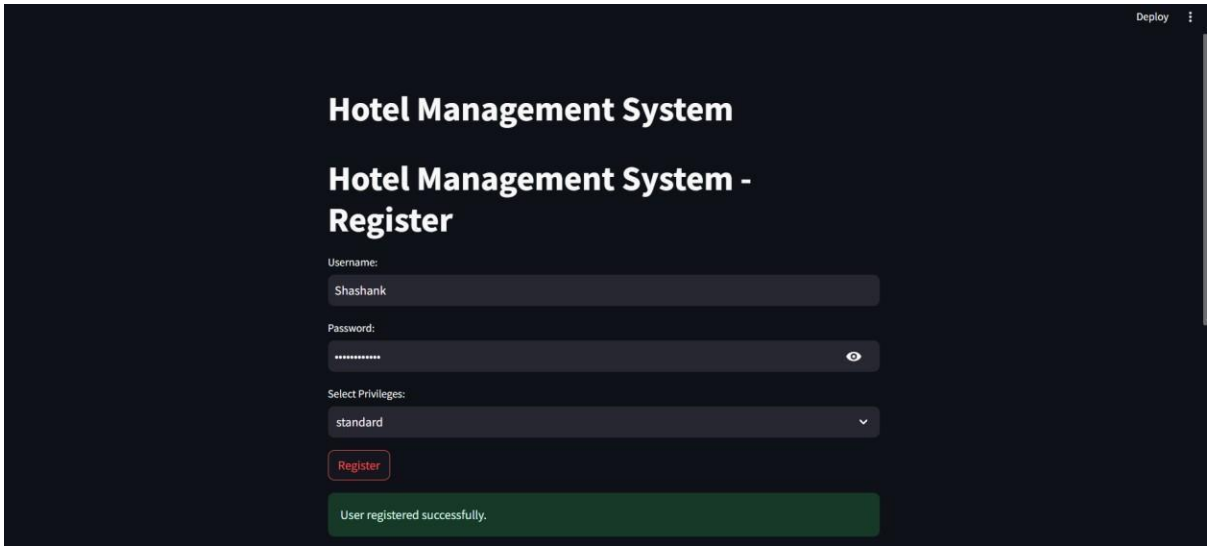
FOREIGN KEY Constraints

```sql
FOREIGN KEY (GuestID) REFERENCES Guest(GuestID) ON DELETE CASCADE,
FOREIGN KEY (RoomNumber) REFERENCES Room(RoomNumber)
```

CRUD Operations:

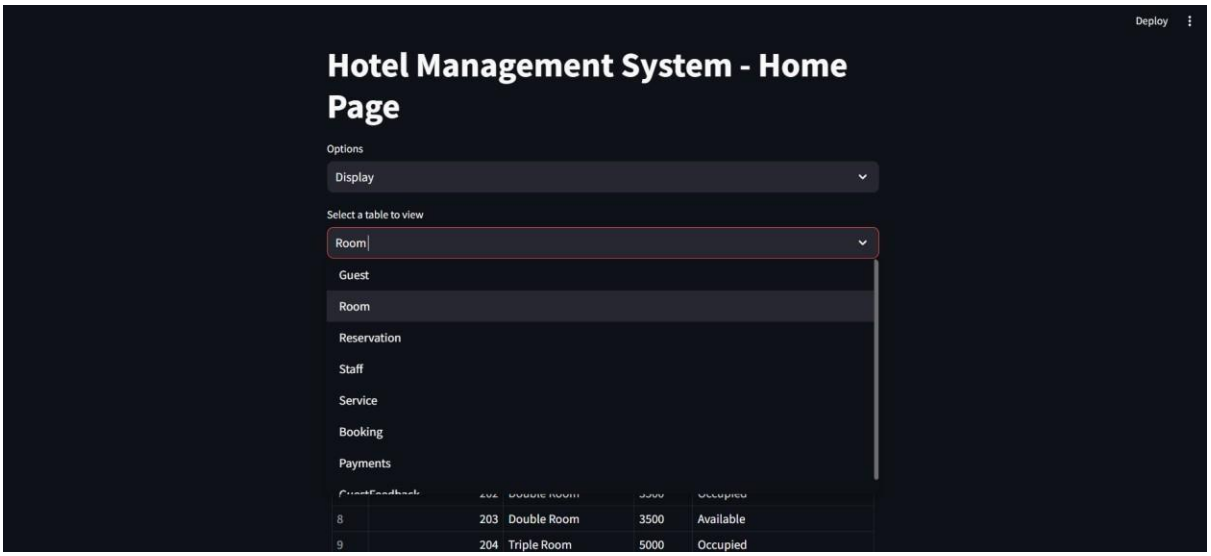
User creation:

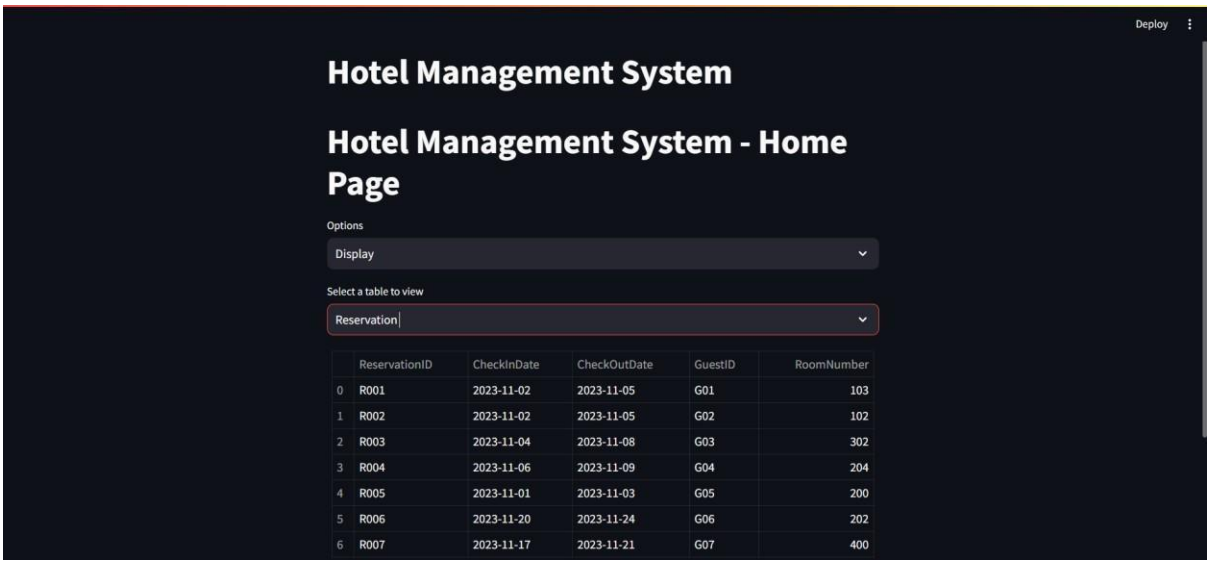
```python
def register_user(username, password, privileges):
    try:
        cursor = connection.cursor()
        if(privileges=="admin"):
            cursor.callproc('insert_admin_user', (username, password))
        else :
            query = "INSERT INTO User (Username, Password, Privileges) VALUES (%s, %s, %s)"
            cursor.execute(query, (username, password, privileges))
        connection.commit()
        cursor.close()
        st.success("User registered successfully.")
    except pymysql.Error as err:
        st.error(f"Error: {err}")
        st.error("User registration failed.")
```



Read Tables:

```python
def display_entity(table_name):
    data = execute_query(f"SELECT * FROM {table_name}")
    st.table(data)
```



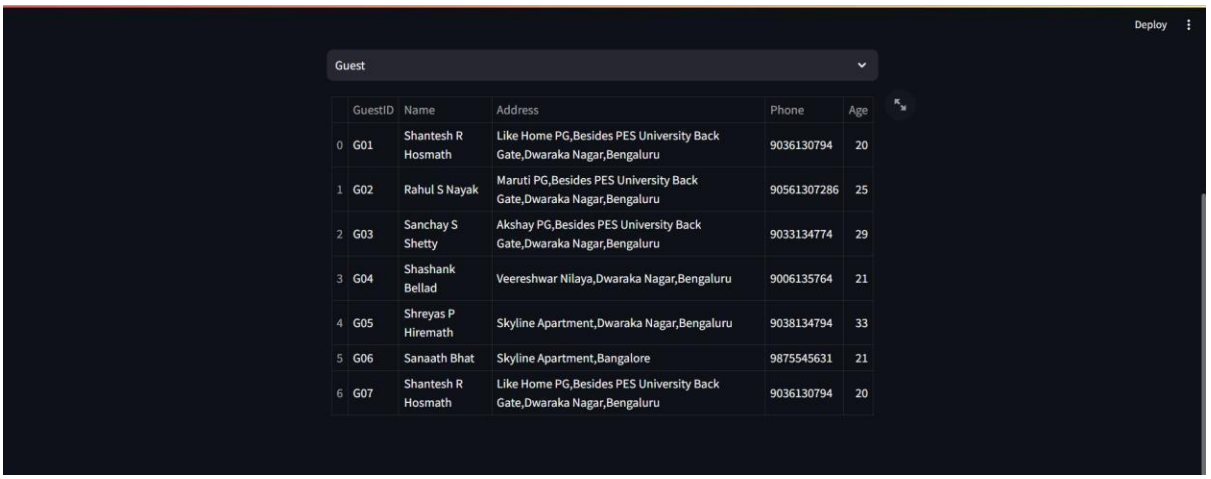


Update Operation:

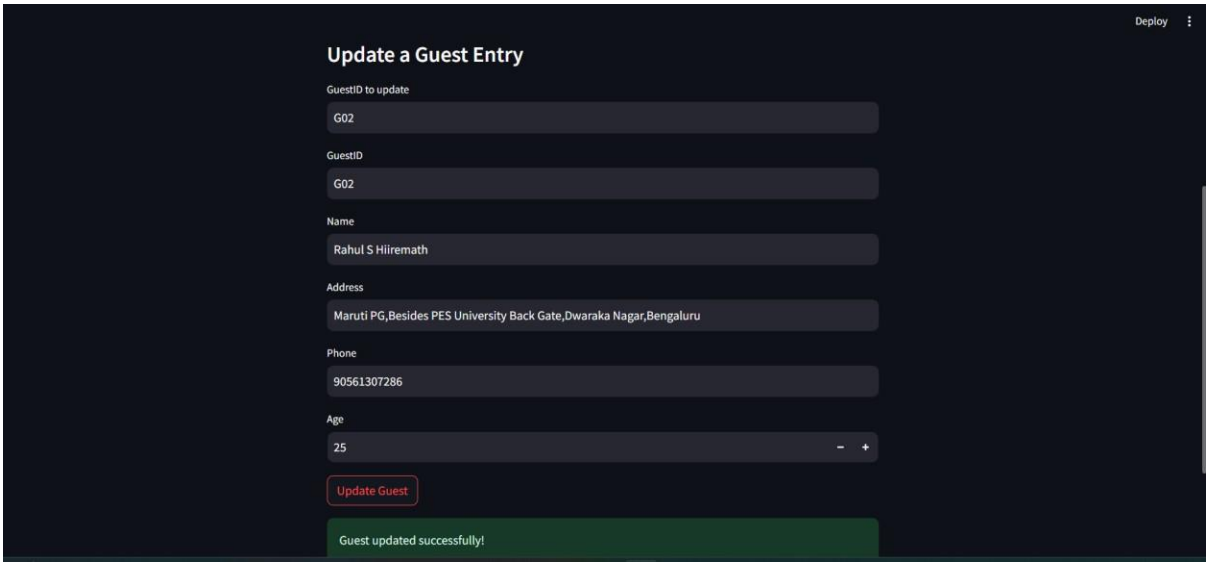
```python
def update_entity(entity_name, table_name, columns,primary_key):
    if(privileges=="admin"):
        st.subheader(f"Update a {entity_name} Entry")
        id_to_update = st.text_input(f"{primary_key} to update")
        if id_to_update:
            entry_to_update = execute_query(f"SELECT * FROM {table_name} WHERE {primary_key} = %s", (id_to_update,))
            if not entry_to_update.empty:
                update_values = []
                for col_name, col_type in columns:
                    if col_type == "int":
                        update_values.append(st.number_input(f"{col_name}", value=entry_to_update.iloc[0][col_name]))
                    else:
                        update_values.append(st.text_input(f"{col_name}", value=entry_to_update.iloc[0][col_name]))
                if st.button(f"Update {entity_name}"):
                    set_clause=','.join([f'{col_name} = %s' for col_name, _ in columns])
                    query = f"UPDATE {table_name} SET {set_clause} WHERE {primary_key} = %s"
                    data = tuple([update_values[i] for i in range(len(columns))]+ [id_to_update])
                    execute_query(query, data)
                    st.success(f"{entity_name} updated successfully!")

            else:
                st.error(f"{entity_name} ID not found.")
    else:
        st.error("User with standard privileges cannot update data.")
```

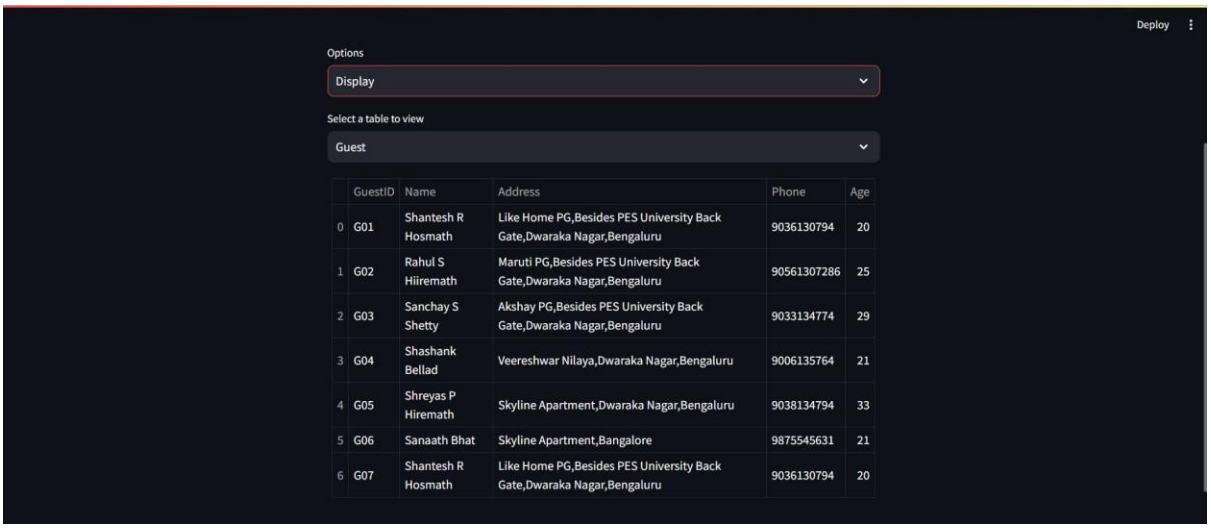
Before updating:



	GuestID	Name	Address	Phone	Age
0	G01	Shantesh R Hosmath	Like Home PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9036130794	20
1	G02	Rahul S Nayak	Maruti PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	90561307286	25
2	G03	Sanchay S Shetty	Akshay PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9033134774	29
3	G04	Shashank Bellad	Veereshwar Nilaya,Dwaraka Nagar,Bengaluru	9006135764	21
4	G05	Shreyas P Hiremath	Skyline Apartment,Dwaraka Nagar,Bengaluru	9038134794	33
5	G06	Sanaath Bhat	Skyline Apartment,Bangalore	9875545631	21
6	G07	Shantesh R Hosmath	Like Home PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9036130794	20



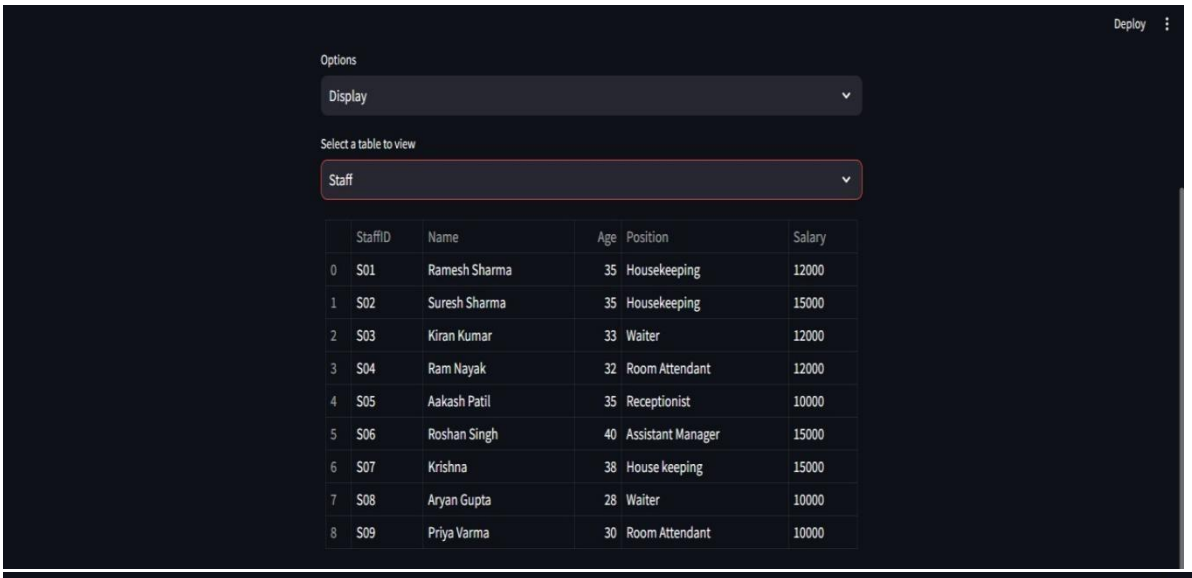
After updating guest with guestID=2:



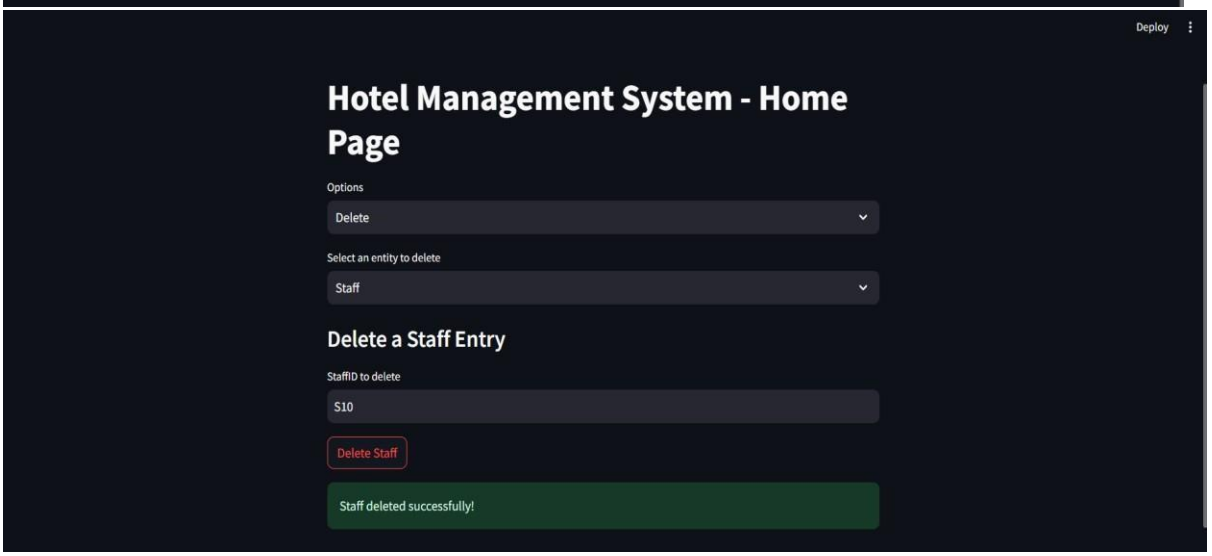
	GuestID	Name	Address	Phone	Age
0	G01	Shantesh R Hosmath	Like Home PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9036130794	20
1	G02	Rahul S Hiiremath	Maruti PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	90561307286	25
2	G03	Sanchay S Shetty	Akshay PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9033134774	29
3	G04	Shashank Bellad	Veereshwar Nilaya,Dwaraka Nagar,Bengaluru	9006135764	21
4	G05	Shreyas P Hiremath	Skyline Apartment,Dwaraka Nagar,Bengaluru	9038134794	33
5	G06	Sanaath Bhat	Skyline Apartment,Bangalore	9875545631	21
6	G07	Shantesh R Hosmath	Like Home PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9036130794	20

Delete Operation:

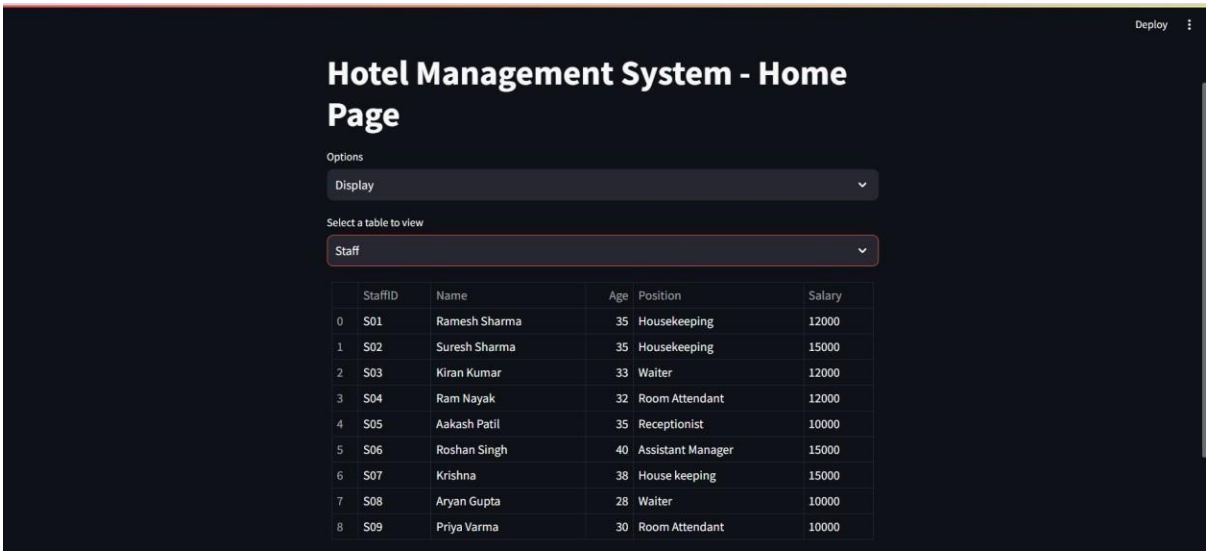
Before deleting:



	StaffID	Name	Age	Position	Salary
0	S01	Ramesh Sharma	35	Housekeeping	12000
1	S02	Suresh Sharma	35	Housekeeping	15000
2	S03	Kiran Kumar	33	Waiter	12000
3	S04	Ram Nayak	32	Room Attendant	12000
4	S05	Aakash Patil	35	Receptionist	10000
5	S06	Roshan Singh	40	Assistant Manager	15000
6	S07	Krishna	38	House keeping	15000
7	S08	Aryan Gupta	28	Waiter	10000
8	S09	Priya Varma	30	Room Attendant	10000



After deleting Staff details with StaffID=S10:



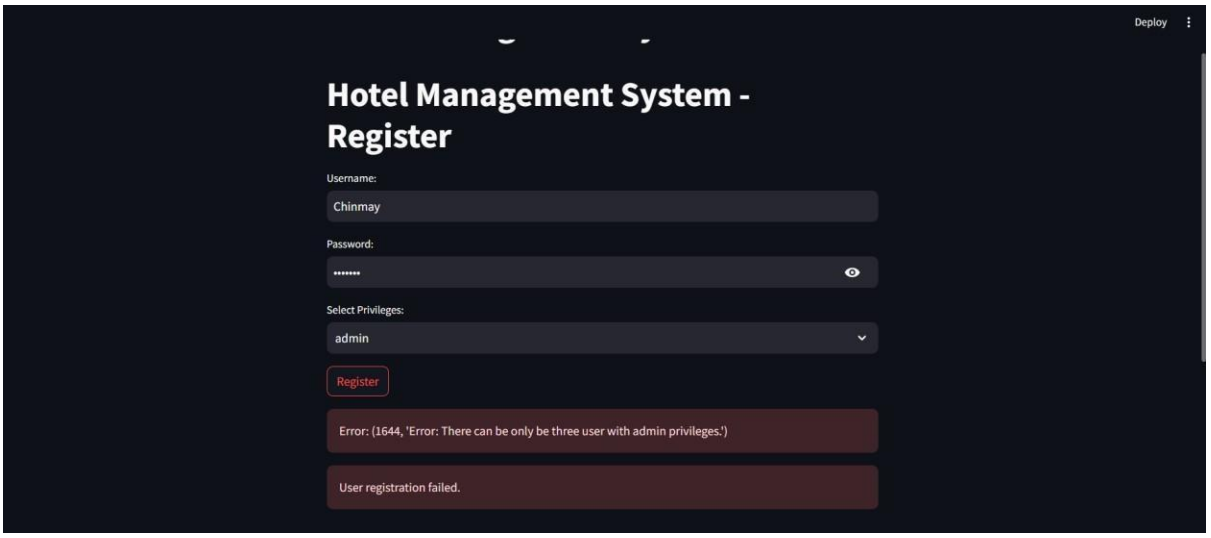
	StaffID	Name	Age	Position	Salary
0	S01	Ramesh Sharma	35	Housekeeping	12000
1	S02	Suresh Sharma	35	Housekeeping	15000
2	S03	Kiran Kumar	33	Waiter	12000
3	S04	Ram Nayak	32	Room Attendant	12000
4	S05	Aakash Patil	35	Receptionist	10000
5	S06	Roshan Singh	40	Assistant Manager	15000
6	S07	Krishna	38	House keeping	15000
7	S08	Aryan Gupta	28	Waiter	10000
8	S09	Priya Varma	30	Room Attendant	10000

Procedure used to limit the registration of admin users to 3:

```
CREATE PROCEDURE insert_admin_user(IN p_username VARCHAR(50),IN p_password VARCHAR(50))
BEGIN
    DECLARE admin_count INT;

    -- Check if the new user is being assigned admin privileges
    IF (SELECT COUNT(*) FROM User WHERE Privileges = 'admin') > 2 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: There can be only be three user with admin privileges.';
    ELSE
        -- Insert the new user with admin privileges
        INSERT INTO User (Username, Password, Privileges) VALUES (p_username, p_password, 'admin');
        SELECT 'User with admin privileges inserted successfully.' AS Message;
    END IF;
END //
DELIMITER ;
```

```
def register_user(username, password, privileges):
    try:
        cursor = connection.cursor()
        if(privileges=="admin"):
            cursor.callproc('insert_admin_user', (username, password))
        else :
            query = "INSERT INTO User (Username, Password, Privileges) VALUES (%s, %s, %s)"
            cursor.execute(query, (username, password, privileges))
        connection.commit()
        cursor.close()
        st.success("User registered successfully.")
    except pymysql.Error as err:
        st.error(f"Error: {err}")
        st.error("User registration failed.")
```



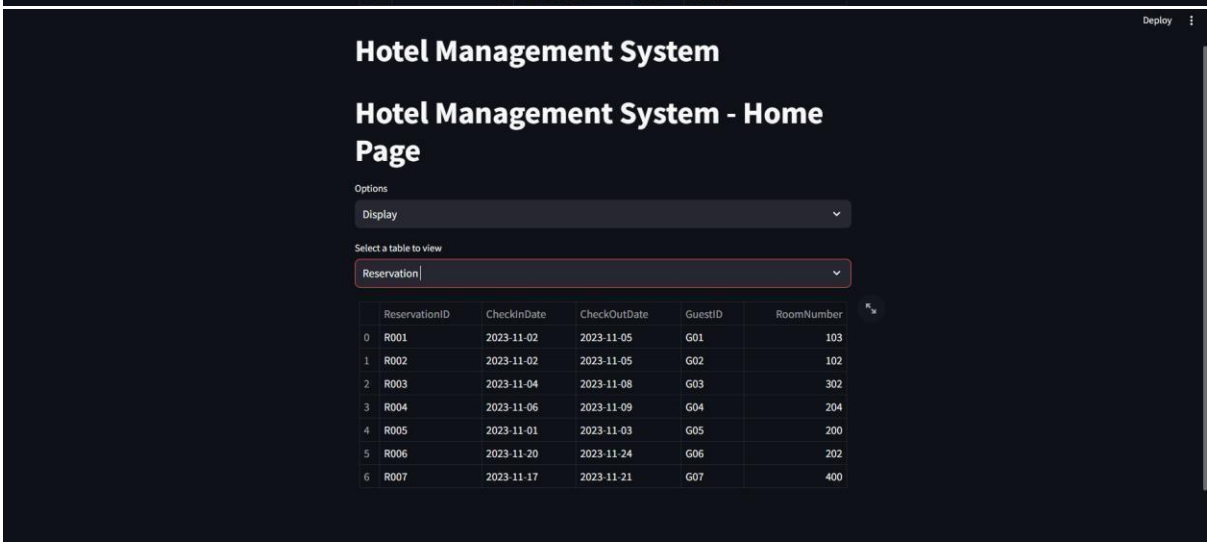
Trigger to automatically update the Room status on deleting or adding a reservation:

```
CREATE TRIGGER UpdateRoomAvailabilityOnUpdate
AFTER UPDATE ON Reservation
FOR EACH ROW
BEGIN
    IF OLD.RoomNumber != NEW.RoomNumber THEN
        -- Room number changed, update old and new room availability
        UPDATE Room
        SET Availability = 'Available'
        WHERE RoomNumber = OLD.RoomNumber;

        UPDATE Room
        SET Availability = 'Occupied'
        WHERE RoomNumber = NEW.RoomNumber;
    END IF;
END;
//
DELIMITER ;
```

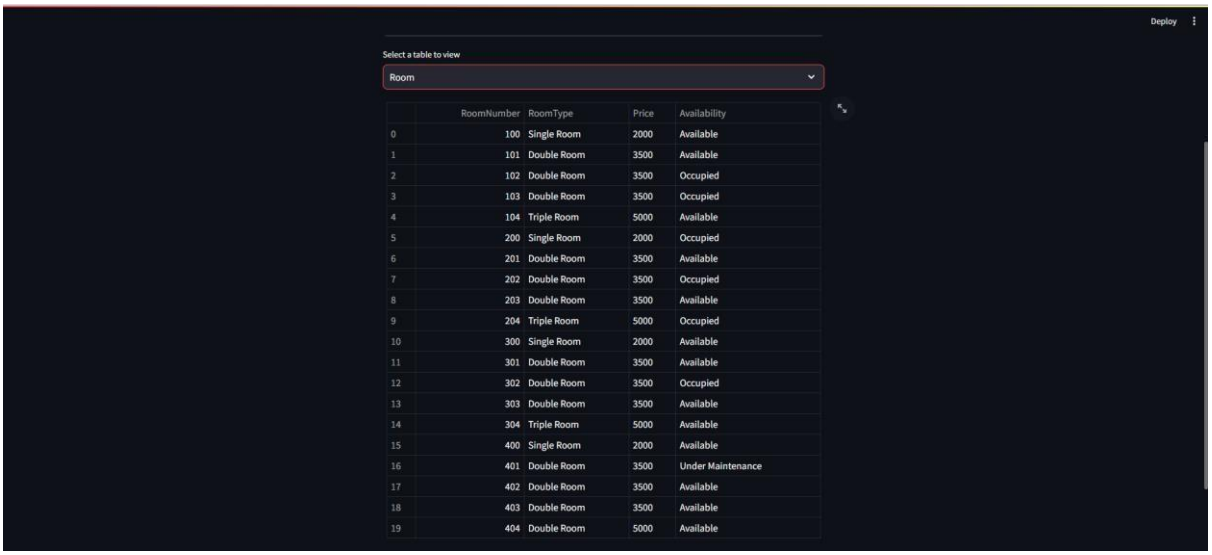
Before deleting a reservation:

	RoomNumber	RoomType	Price	Availability
0	100	Single Room	2000	Available
1	101	Double Room	3500	Available
2	102	Double Room	3500	Occupied
3	103	Double Room	3500	Occupied
4	104	Triple Room	5000	Available
5	200	Single Room	2000	Occupied
6	201	Double Room	3500	Available
7	202	Double Room	3500	Occupied
8	203	Double Room	3500	Available
9	204	Triple Room	5000	Occupied
10	300	Single Room	2000	Available
11	301	Double Room	3500	Available
12	302	Double Room	3500	Occupied
13	303	Double Room	3500	Available
14	304	Triple Room	5000	Available
15	400	Single Room	2000	Occupied
16	401	Double Room	3500	Under Maintenance
17	402	Double Room	3500	Available
18	403	Double Room	3500	Available
19	404	Double Room	5000	Available



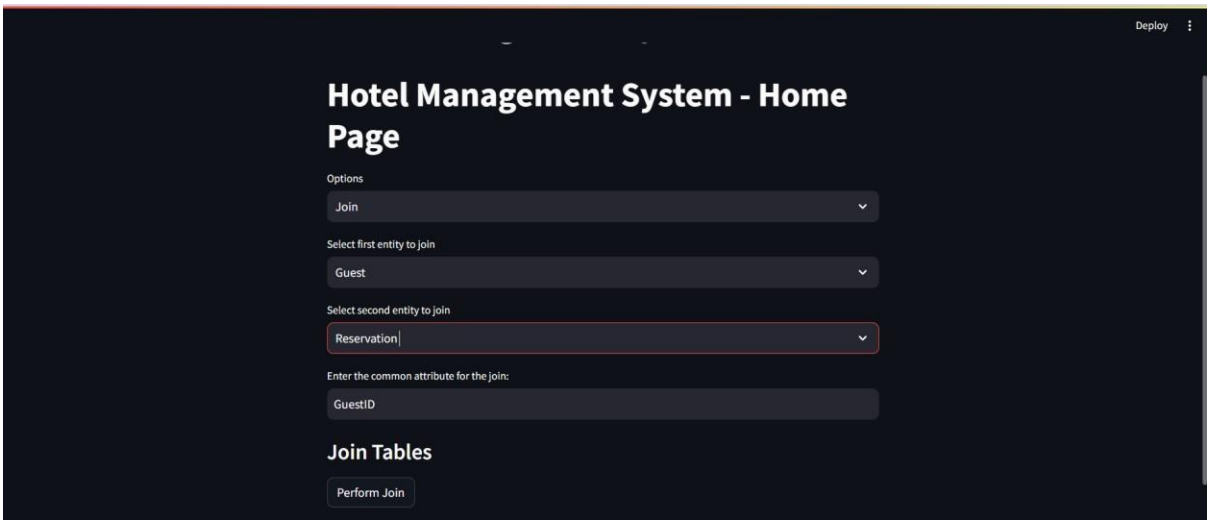
	ReservationID	CheckInDate	CheckOutDate	GuestID	RoomNumber
0	R001	2023-11-02	2023-11-05	G01	103
1	R002	2023-11-02	2023-11-05	G02	102
2	R003	2023-11-04	2023-11-08	G03	302
3	R004	2023-11-06	2023-11-09	G04	204
4	R005	2023-11-01	2023-11-03	G05	200
5	R006	2023-11-20	2023-11-24	G06	202
6	R007	2023-11-17	2023-11-21	G07	400

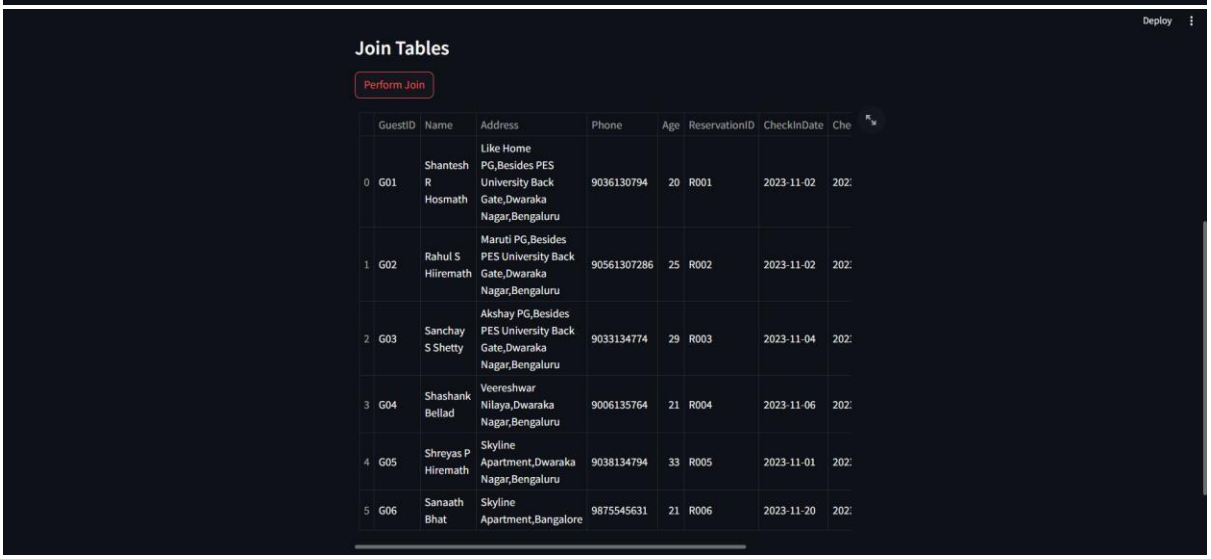
After deleting reservation corresponding to room number 400:



	RoomNumber	RoomType	Price	Availability
0	100	Single Room	2000	Available
1	101	Double Room	3500	Available
2	102	Double Room	3500	Occupied
3	103	Double Room	3500	Occupied
4	104	Triple Room	5000	Available
5	200	Single Room	2000	Occupied
6	201	Double Room	3500	Available
7	202	Double Room	3500	Occupied
8	203	Double Room	3500	Available
9	204	Triple Room	5000	Occupied
10	300	Single Room	2000	Available
11	301	Double Room	3500	Available
12	302	Double Room	3500	Occupied
13	303	Double Room	3500	Available
14	304	Triple Room	5000	Available
15	400	Single Room	2000	Available
16	401	Double Room	3500	Under Maintenance
17	402	Double Room	3500	Available
18	403	Double Room	3500	Available
19	404	Double Room	5000	Available

Join operation:





	GuestID	Name	Address	Phone	Age	ReservationID	CheckInDate	Che
0	G01	Shantesh R Hosmath	Like Home PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9036130794	20	R001	2023-11-02	202
1	G02	Rahul S Hiiremath	Maruti PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	90561307286	25	R002	2023-11-02	202
2	G03	Sanchay S Shetty	Akshay PG,Besides PES University Back Gate,Dwaraka Nagar,Bengaluru	9033134774	29	R003	2023-11-04	202
3	G04	Shashank Bellad	Veereshwar Nilaya,Dwaraka Nagar,Bengaluru	9006135764	21	R004	2023-11-06	202
4	G05	Shreyas P Hiremath	Skyline Apartment,Dwaraka Nagar,Bengaluru	9038134794	33	R005	2023-11-01	202
5	G06	Sanaath Bhat	Skyline Apartment,Bangalore	9875545631	21	R006	2023-11-20	202