Coding Challenges - Hotel Management System

Instructions:

- Submit your SQL solutions via a GitHub repository and share the link with trainers.
- Implement the schema, queries, and procedures as outlined below.

Problem Statement:

Create an SQL schema from the **Hotel** and **Guest** classes, using their attributes for table column names.

SQL Schema:

Table: Hotels

- HotelID (**Primary Key, int**): Unique identifier for each hotel.
- Name (**string**): The name of the hotel.
- Location (**string**): The location of the hotel.
- Rating (**decimal**): The average rating of the hotel (1-5).

Table: Rooms

- RoomID (**Primary Key, int**): Unique identifier for each room.
- HotelID (**Foreign Key, int**): References **HotelID** in Hotels table.
- RoomNumber (**string**): Room number or identifier.
- RoomType (**string**): Type of room (e.g., "Single," "Double," "Suite").
- PricePerNight (**decimal**): Cost per night.
- Available (**bit**): Indicates if the room is available for booking (1 for yes, 0 for no).

Table: Guests

- GuestID (**Primary Key, int**): Unique identifier for each guest.
- FullName (**string**): Name of the guest.
- Email (**string**): Guest email (unique).
- PhoneNumber (**string**): Guest phone number (unique).
- CheckInDate (**datetime**): The date the guest checked in.
- CheckOutDate (**datetime**): The date the guest checked out.

Table: Bookings

- BookingID (**Primary Key, int**): Unique identifier for each booking.
- GuestID (**Foreign Key, int**): References **GuestID** in Guests table.
- RoomID (**Foreign Key, int**): References **RoomID** in Rooms table.
- BookingDate (**datetime**): The date of booking.
- TotalAmount (**decimal**): The total price for the stay.

- Status (**string**): Booking status (e.g., "Confirmed," "Cancelled," "Checked Out").

Table: Payments

- PaymentID (**Primary Key, int**): Unique identifier for each payment.
- BookingID (**Foreign Key, int**): References **BookingID** in Bookings table.
- AmountPaid (**decimal**): The amount paid.
- PaymentDate (**datetime**): Date and time of payment.
- PaymentMethod (**string**): Payment method (e.g., "Credit Card," "Cash").

Table: Events

- EventID (**Primary Key, int**): Unique identifier for each event hosted at the hotel.
- HotelID (**Foreign Key, int**): References **HotelID** in Hotels table.
- EventName (**string**): The name or title of the event.
- EventDate (**datetime**): Date and time of the event.
- Venue (**string**): Venue of the event.

Table: EventParticipants

- ParticipantID (**Primary Key, int**): Unique identifier for each participant.
- ParticipantName (**string**): Name of the participant (guest or organization).
- ParticipantType (**string**): Type of participant ("Guest" or "Organization").
- EventID (**Foreign Key, int**): References **EventID** of the associated event.

Tasks:

1. **Provide a SQL script** to initialize the **Hotel Management System** database.

   Create database Hotelmanagementsystem

2. **Create tables** for **hotels, rooms, guests, bookings, payments, events, and event participants**, defining appropriate primary and foreign keys.

   ```
   CREATE TABLE Hotels (
     HotelID INT PRIMARY KEY IDENTITY(1,1),
     Name VARCHAR(255) NOT NULL,
     Location VARCHAR(255) NOT NULL,
     Rating DECIMAL(2,1) CHECK (Rating BETWEEN 1 AND 5)
   );
   CREATE TABLE Rooms (
     RoomID INT PRIMARY KEY IDENTITY(1,1),
     HotelID INT,
     RoomNumber VARCHAR(50) NOT NULL,
   ```

```
    RoomType VARCHAR(50) NOT NULL,
    PricePerNight DECIMAL(10,2) NOT NULL,
    Available BIT NOT NULL,
    FOREIGN KEY (HotelID) REFERENCES Hotels(HotelID) ON DELETE CASCADE
);
CREATE TABLE Guests (
    GuestID INT PRIMARY KEY IDENTITY(1,1),
    FullName VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    PhoneNumber VARCHAR(20) UNIQUE NOT NULL,
    CheckInDate DATETIME,
    CheckOutDate DATETIME
);
CREATE TABLE Bookings (
    BookingID INT PRIMARY KEY IDENTITY(1,1),
    GuestID INT,
    RoomID INT,
    BookingDate DATETIME NOT NULL,
    TotalAmount DECIMAL(10,2) NOT NULL,
    Status VARCHAR(50) NOT NULL,
    FOREIGN KEY (GuestID) REFERENCES Guests(GuestID),
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)
);
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY IDENTITY(1,1),
    BookingID INT,
    AmountPaid DECIMAL(10,2) NOT NULL,
    PaymentDate DATETIME NOT NULL,
    PaymentMethod VARCHAR(50) NOT NULL,
    FOREIGN KEY (BookingID) REFERENCES Bookings(BookingID) );
CREATE TABLE EventParticipants (
    ParticipantID INT PRIMARY KEY IDENTITY(1,1),
    ParticipantName VARCHAR(255) NOT NULL,
    ParticipantType VARCHAR(50) NOT NULL CHECK (ParticipantType IN ('Guest',
'Organization')),
    EventID INT,
    FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE
);
```

Records insertion

```
INSERT INTO Hotels (Name, Location, Rating) VALUES
('Grand Plaza', 'New York', 4.5),
('Ocean View', 'Miami', 4.0),
('Mountain Retreat', 'Denver', 4.2),
('City Lights', 'Los Angeles', 4.7),
('Heritage Inn', 'Boston', 4.3);

INSERT INTO Rooms (HotelID, RoomNumber, RoomType, PricePerNight, Available) VALUES
(1, '101', 'Single', 120.00, 1),
(1, '102', 'Double', 150.00, 1),
(2, '201', 'Suite', 200.00, 0),
(3, '301', 'Single', 110.00, 1),
(4, '401', 'Double', 160.00, 0);

INSERT INTO Guests (FullName, Email, PhoneNumber, CheckInDate, CheckOutDate) VALUES
('John Doe', 'john.doe@example.com', '1234567890', '2025-03-20', '2025-03-22'),
('Alice Smith', 'alice.smith@example.com', '0987654321', '2025-03-21', '2025-03-23'),
('Michael Brown', 'michael.brown@example.com', '1122334455', '2025-03-19', '2025-03-21'),
('Emily Davis', 'emily.davis@example.com', '2233445566', '2025-03-22', '2025-03-24'),
('Chris Wilson', 'chris.wilson@example.com', '3344556677', '2025-03-23', '2025-03-25');

INSERT INTO Bookings (GuestID, RoomID, BookingDate, TotalAmount, Status) VALUES
(1, 1, '2025-03-18', 240.00, 'Confirmed'),
(2, 2, '2025-03-19', 300.00, 'Checked Out'),
(3, 3, '2025-03-17', 400.00, 'Cancelled'),
(4, 4, '2025-03-20', 220.00, 'Confirmed'),
(5, 5, '2025-03-21', 320.00, 'Checked Out');
```

```
INSERT INTO Payments (BookingID, AmountPaid, PaymentDate, PaymentMethod) VALUES
(1, 240.00, '2025-03-18', 'Credit Card'),
(2, 300.00, '2025-03-19', 'Cash'),
(3, 0.00, '2025-03-17', 'N/A'),
(4, 220.00, '2025-03-20', 'Debit Card'),
(5, 320.00, '2025-03-21', 'Credit Card');

INSERT INTO Events (HotelID, EventName, EventDate, Venue) VALUES
(1, 'Business Summit', '2025-04-10', 'Grand Ballroom'),
(2, 'Wedding Expo', '2025-04-15', 'Conference Hall'),
(3, 'Tech Meetup', '2025-04-20', 'Event Center'),
(4, 'Music Fest', '2025-04-25', 'Outdoor Stage'),
(5, 'Art Exhibition', '2025-04-30', 'Gallery Hall');

INSERT INTO EventParticipants (ParticipantName, ParticipantType, EventID) VALUES
('Divi', 'Guest', 1),
('swetha', 'Guest', 2),
('Greengold ltd', 'Organization', 3),
('Mouni', 'Guest', 4),
('Creative Arts Ltd.', 'Organization', 5);
```

3. **Ensure the script handles potential errors**, such as checking if the database or tables already exist before creating them.
4. **Write an SQL query** to retrieve a list of available rooms for booking (Available = 1).

   ```
   Select *from rooms where available=’1’;
   ```
5. **Retrieve names of participants** registered for a specific hotel event using an @EventID parameter.

   ```
   CREATE PROCEDURE GetEventParticipants @EventID INT
   AS
   BEGIN
      SELECT ParticipantName FROM EventParticipants WHERE EventID = @EventID;
   END;
   ```
6. **Create a stored procedure** that allows a hotel to update its information (name and location) in the "Hotels" table.

   ```
   CREATE PROCEDURE UpdateHotelInfo @HotelID INT, @Name VARCHAR(255), @Location VARCHAR(255)
   ```

```
AS

BEGIN

   UPDATE Hotels SET Name = @Name, Location = @Location WHERE HotelID = @HotelID;

END;
```

7. **Write an SQL query** to calculate the **total revenue generated** by each hotel from confirmed bookings.

   ```
   SELECT h.HotelID, h.Name, SUM(b.TotalAmount) AS TotalRevenue

   FROM Hotels h

   JOIN Rooms r ON h.HotelID = r.HotelID

   JOIN Bookings b ON r.RoomID = b.RoomID

   WHERE b.Status = 'Confirmed'

   GROUP BY h.HotelID, h.Name;
   ```

8. **Find rooms that have never been booked** by selecting their details from the Rooms table.

   ```
   SELECT * FROM Rooms WHERE RoomID NOT IN (SELECT RoomID FROM Bookings);
   ```

9. **Retrieve total payments per month and year**, ensuring missing months are handled properly.

   ```
   SELECT YEAR(PaymentDate) AS Year, MONTH(PaymentDate) AS Month(SUM(AmountPaid), AS TotalPayments

   FROM Payments

   GROUP BY YEAR(PaymentDate), MONTH(PaymentDate)

   ORDER BY Year, Month;
   ```

10. **Retrieve a list of room types** that are either **priced between $50 and $150 per night or above $300 per night**.

    ```
    SELECT DISTINCT RoomType FROM Rooms WHERE PricePerNight BETWEEN 50 AND 150 OR PricePerNight > 300;
    ```

11. **Retrieve rooms along with their guests**, including only rooms that are currently occupied.

    ```
    SELECT r.RoomID, r.RoomNumber, g.FullName

    FROM Rooms r

    JOIN Bookings b ON r.RoomID = b.RoomID

    JOIN Guests g ON b.GuestID = g.GuestID

    WHERE b.Status = 'Checked In';
    ```

12. **Find the total number of participants** in events held in a specific city (@CityName).

    ```
    CREATE PROCEDURE GetTotalParticipantsInCity @CityName VARCHAR(255)

    AS

    BEGIN

       SELECT COUNT(ep.ParticipantID) AS TotalParticipants
    ```

```
    FROM EventParticipants ep
    JOIN Events e ON ep.EventID = e.EventID
    JOIN Hotels h ON e.HotelID = h.HotelID
    WHERE h.Location = @CityName;
END;
```

13. **Retrieve a list of unique room types** available in a specific hotel.

```
DECLARE @HotelID INT = 1;
SELECT DISTINCT RoomType FROM Rooms WHERE HotelID = @HotelID;
```

14. **Find guests who have never made a booking** from the hotel management system.

```
SELECT * FROM Guests WHERE GuestID NOT IN (SELECT GuestID FROM Bookings);
```

15. **Retrieve names of all booked rooms** along with the guests who booked them.

```
SELECT r.RoomNumber, g.FullName
FROM Rooms r
JOIN Bookings b ON r.RoomID = b.RoomID
JOIN Guests g ON b.GuestID = g.GuestID;
```

16. **Retrieve all hotels along with the count of available rooms** in each hotel.

```
SELECT h.HotelID, h.Name, COUNT(r.RoomID) AS AvailableRooms
FROM Hotels h
JOIN Rooms r ON h.HotelID = r.HotelID
WHERE r.Available = 1
GROUP BY h.HotelID, h.Name;
```

17. **Find pairs of rooms from the same hotel** that belong to the same room type.

```
SELECT r1.RoomID AS Room1, r2.RoomID AS Room2, r1.RoomType
FROM Rooms r1
JOIN Rooms r2 ON r1.HotelID = r2.HotelID AND r1.RoomType = r2.RoomType AND r1.RoomID <
r2.RoomID;
```

18. **List all possible combinations** of hotels and events.

```
SELECT h.Name AS HotelName, e.EventName
FROM Hotels h
CROSS JOIN Events e;
```

19. **Determine the hotel with the highest number of bookings.**

```
SELECT TOP 1 h.HotelID, h.Name, COUNT(b.BookingID) AS BookingCount
FROM Hotels h
JOIN Rooms r ON h.HotelID = r.HotelID
JOIN Bookings b ON r.RoomID = b.RoomID

GROUP BY h.HotelID, h.Name

ORDER BY BookingCount DESC;
```

BY,
DIVYALAKSHMI M