**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;

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

USE HotelManagementSystem;

-- Hotels Table

CREATE TABLE Hotels (

HotelID INT PRIMARY KEY IDENTITY(1,1),

Name VARCHAR(100) NOT NULL,

Location VARCHAR(100) NOT NULL,

Rating DECIMAL(2,1) DEFAULT 0.0 CHECK (Rating BETWEEN 1.0 AND 5.0)

);

-- Rooms Table

CREATE TABLE Rooms (

RoomID INT PRIMARY KEY IDENTITY(1,1),

HotelID INT NOT NULL,

RoomNumber VARCHAR(20) NOT NULL,

RoomType VARCHAR(50),

PricePerNight DECIMAL(10,2) NOT NULL,

Available BIT DEFAULT 1 NOT NULL,

FOREIGN KEY (HotelID) REFERENCES Hotels(HotelID) ON DELETE CASCADE

);

-- Guests Table

CREATE TABLE Guests (

GuestID INT PRIMARY KEY IDENTITY(1,1),

FullName VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

PhoneNumber VARCHAR(15) UNIQUE NOT NULL,

CheckInDate DATETIME NOT NULL,

CheckOutDate DATETIME NOT NULL

);

-- Bookings Table

CREATE TABLE Bookings (

BookingID INT PRIMARY KEY IDENTITY(1,1),

GuestID INT NOT NULL,

RoomID INT NOT NULL,

BookingDate DATETIME NOT NULL DEFAULT GETDATE(),

TotalAmount DECIMAL(10,2) NOT NULL CHECK (TotalAmount >= 0),

Status VARCHAR(20) NOT NULL CHECK (Status IN ('Confirmed', 'Cancelled', 'Checked Out')),

FOREIGN KEY (GuestID) REFERENCES Guests(GuestID) ON DELETE CASCADE,

FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID) ON DELETE CASCADE

);

-- Payments Table

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY IDENTITY(1,1),

BookingID INT NOT NULL,

AmountPaid DECIMAL(10,2) NOT NULL CHECK (AmountPaid >= 0),

PaymentDate DATETIME NOT NULL DEFAULT GETDATE(),

PaymentMethod VARCHAR(20) NOT NULL CHECK (PaymentMethod IN ('Credit Card', 'Cash', 'Debit Card', 'UPI')),

FOREIGN KEY (BookingID) REFERENCES Bookings(BookingID) ON DELETE CASCADE

);

-- Events Table

CREATE TABLE Events (

EventID INT PRIMARY KEY IDENTITY(1,1),

HotelID INT NOT NULL,

EventName VARCHAR(100) NOT NULL,

EventDate DATETIME NOT NULL,

Venue VARCHAR(100) NOT NULL,

FOREIGN KEY (HotelID) REFERENCES Hotels(HotelID) ON DELETE CASCADE

);

-- Event Participants Table

CREATE TABLE EventParticipants (

ParticipantID INT PRIMARY KEY IDENTITY(1,1),

ParticipantName VARCHAR(100) NOT NULL,

ParticipantType VARCHAR(20) NOT NULL CHECK (ParticipantType IN ('Guest', 'Organization')),

EventID INT NOT NULL,

FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE

);

1. **Ensure the script handles potential errors**, such as checking if the database or tables already exist before creating them.

-- Check if the database exists before creating it

IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = 'HotelManagementSystem')

BEGIN

CREATE DATABASE HotelManagementSystem;

PRINT 'Database HotelManagementSystem created successfully.';

END

ELSE

PRINT 'Database HotelManagementSystem already exists.';

USE HotelManagementSystem;

-- Check if a table exists before creating it

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Hotels')

BEGIN

CREATE TABLE Hotels (

HotelID INT PRIMARY KEY IDENTITY(1,1),

Name VARCHAR(100) NOT NULL,

Location VARCHAR(100) NOT NULL,

Rating DECIMAL(2,1) DEFAULT 0.0 CHECK (Rating BETWEEN 1.0 AND 5.0)

);

PRINT 'Table Hotels created successfully.';

END

ELSE

PRINT 'Table Hotels already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Rooms')

BEGIN

CREATE TABLE Rooms (

RoomID INT PRIMARY KEY IDENTITY(1,1),

HotelID INT NOT NULL,

RoomNumber VARCHAR(20) NOT NULL,

RoomType VARCHAR(50),

PricePerNight DECIMAL(10,2) NOT NULL,

Available BIT DEFAULT 1 NOT NULL,

FOREIGN KEY (HotelID) REFERENCES Hotels(HotelID) ON DELETE CASCADE

);

PRINT 'Table Rooms created successfully.';

END

ELSE

PRINT 'Table Rooms already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Guests')

BEGIN

CREATE TABLE Guests (

GuestID INT PRIMARY KEY IDENTITY(1,1),

FullName VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

PhoneNumber VARCHAR(15) UNIQUE NOT NULL,

CheckInDate DATETIME NOT NULL,

CheckOutDate DATETIME NOT NULL

);

PRINT 'Table Guests created successfully.';

END

ELSE

PRINT 'Table Guests already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Bookings')

BEGIN

CREATE TABLE Bookings (

BookingID INT PRIMARY KEY IDENTITY(1,1),

GuestID INT NOT NULL,

RoomID INT NOT NULL,

BookingDate DATETIME NOT NULL DEFAULT GETDATE(),

TotalAmount DECIMAL(10,2) NOT NULL CHECK (TotalAmount >= 0),

Status VARCHAR(20) NOT NULL CHECK (Status IN ('Confirmed', 'Cancelled', 'Checked Out')),

FOREIGN KEY (GuestID) REFERENCES Guests(GuestID) ON DELETE CASCADE,

FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID) ON DELETE CASCADE

);

PRINT 'Table Bookings created successfully.';

END

ELSE

PRINT 'Table Bookings already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Payments')

BEGIN

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY IDENTITY(1,1),

BookingID INT NOT NULL,

AmountPaid DECIMAL(10,2) NOT NULL CHECK (AmountPaid >= 0),

PaymentDate DATETIME NOT NULL DEFAULT GETDATE(),

PaymentMethod VARCHAR(20) NOT NULL CHECK (PaymentMethod IN ('Credit Card', 'Cash', 'Debit Card', 'UPI')),

FOREIGN KEY (BookingID) REFERENCES Bookings(BookingID) ON DELETE CASCADE

);

PRINT 'Table Payments created successfully.';

END

ELSE

PRINT 'Table Payments already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'Events')

BEGIN

CREATE TABLE Events (

EventID INT PRIMARY KEY IDENTITY(1,1),

HotelID INT NOT NULL,

EventName VARCHAR(100) NOT NULL,

EventDate DATETIME NOT NULL,

Venue VARCHAR(100) NOT NULL,

FOREIGN KEY (HotelID) REFERENCES Hotels(HotelID) ON DELETE CASCADE

);

PRINT 'Table Events created successfully.';

END

ELSE

PRINT 'Table Events already exists.';

IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'EventParticipants')

BEGIN

CREATE TABLE EventParticipants (

ParticipantID INT PRIMARY KEY IDENTITY(1,1),

ParticipantName VARCHAR(100) NOT NULL,

ParticipantType VARCHAR(20) NOT NULL CHECK (ParticipantType IN ('Guest', 'Organization')),

EventID INT NOT NULL,

FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE

);

PRINT 'Table EventParticipants created successfully.';

END

ELSE

PRINT 'Table EventParticipants already exists.';

1. **Write an SQL query** to retrieve a list of available rooms for booking (Available = 1).

select RoomNumber from Rooms where Available=1;

1. **Retrieve names of participants** registered for a specific hotel event using an @EventID parameter.

declare @EventID int =1;

select ee.ParticipantName from EventParticipants ee join Events e on ee.EventID=e.EventID where ee.EventID=@EventID;

1. **Create a stored procedure** that allows a hotel to update its information (name and location) in the "Hotels" table.

go

CREATE PROCEDURE UpdateHotelsInfo

@HotelID INT,

@NewName VARCHAR(100),

@NewLocation VARCHAR(100)

AS

BEGIN

-- Check if the hotel exists

IF EXISTS (SELECT 1 FROM Hotels WHERE HotelID = @HotelID)

BEGIN

-- Update hotel information

UPDATE Hotels

SET Name = @NewName,

Location = @NewLocation

WHERE HotelID = @HotelID;

-- Provide feedback

PRINT 'Hotel Information Updated Successfully.';

END

ELSE

BEGIN

-- Provide error feedback

RAISERROR ('Error: HotelID does not exist.', 16, 1);

END

END;

EXEC UpdateHotelsInfo @HotelID = 1, @NewName = 'Luxury Grand Hotel', @NewLocation = 'San Francisco';

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

SELECT H.HotelID, H.Name AS HotelName, 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

ORDER BY TotalRevenue DESC;

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

select RoomNumber from Rooms where Available=0;

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

select DATENAME(MONTH,PaymentDate),Sum(AmountPaid) from Payments group by DATENAME(MONTH,PaymentDate);

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

select RoomType from Rooms where PricePerNight between 50 and 150 or PricePerNight>300;

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

SELECT R.RoomID, R.RoomNumber, R.RoomType, G.GuestID, G.FullName, G.Email, G.PhoneNumber

FROM Rooms R

JOIN Bookings B ON R.RoomID = B.RoomID

JOIN Guests G ON B.GuestID = G.GuestID

WHERE B.Status = 'Confirmed'

AND G.CheckInDate <= GETDATE()

AND G.CheckOutDate > GETDATE();

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

declare @city varchar(50)='Banquet Hall';

select count(ep.ParticipantID) from EventParticipants ep join Events e on e.EventID=ep.EventID and e.Venue=@city;

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

DECLARE @HotelID INT;

SET @HotelID = 1; -- Change this to the desired hotel ID

SELECT DISTINCT RoomType

FROM Rooms

WHERE HotelID = @HotelID;

1. **Find guests who have never made a booking** from the hotel management system.
2. **Retrieve names of all booked rooms** along with the guests who booked them.

select r.RoomNumber,r.RoomType,g.FullName from Rooms r join Bookings b on r.RoomID=b.RoomID join Guests g on b.GuestID=g.GuestID order by g.FullName;

1. **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

LEFT JOIN Rooms r ON h.HotelID = r.HotelID AND r.Available = 1

GROUP BY h.HotelID, h.Name;

1. **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.HotelID, r1.RoomType

FROM Rooms r1

JOIN Rooms r2 ON r1.HotelID = r2.HotelID

AND r1.RoomType = r2.RoomType

AND r1.RoomID < r2.RoomID

ORDER BY r1.HotelID, r1.RoomType;

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

SELECT h.HotelID, h.Name AS HotelName, e.EventID, e.EventName

FROM Hotels h

CROSS JOIN Events e

ORDER BY h.HotelID, e.EventID;

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

SELECT TOP 1 h.HotelID, h.Name, COUNT(b.BookingID) AS TotalBookings

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

ORDER BY TotalBookings DESC;