Case Study 6: Hotel Reservation System

1. Scenario Overview

The Hotel Reservation System helps hotels manage their rooms, customers, bookings, and payments efficiently. It ensures accurate record keeping, easy retrieval of data, and prevents double bookings through proper transaction handling and normalization.

2. Entities and Relationships

The main entities involved in the Hotel Reservation System are:

1. Room – Details about each room including type, rate, and availability.

2. Customer – Stores guest information such as name, contact, and address.

3. Booking – Information about reservations made by customers.

4. Payment – Payment details related to each booking.

Relationships:

• A Customer can make multiple Bookings.

• A Booking is linked to one Room.

• Each Booking generates one Payment record.

• A Room can be booked multiple times by different customers at different periods.

3. Relational Schema Design with SQL Code

Below is the relational schema and corresponding SQL code for creating the required tables:

```
-- Table: Room

CREATE TABLE Room (

   Room_ID INT PRIMARY KEY,

   Room_No VARCHAR(10) UNIQUE NOT NULL,
    Room_Type VARCHAR(20) NOT NULL,
    Rate DECIMAL(10,2) NOT NULL,
    Status VARCHAR(15) CHECK(Status IN ('Available', 'Booked'))
);

-- Table: Customer
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    Customer_Name VARCHAR(50) NOT NULL,
    Contact_No VARCHAR(15) UNIQUE,
    Address VARCHAR(100)
);

-- Table: Booking
CREATE TABLE Booking (
    Booking_ID INT PRIMARY KEY,
    Customer_ID INT,
    Room_ID INT,
    CheckIn_Date DATE NOT NULL,
    CheckOut_Date DATE NOT NULL,
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (Room_ID) REFERENCES Room(Room_ID)
);


-- Table: Payment
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Booking_ID INT,
    Amount DECIMAL(10,2) NOT NULL,
    Payment_Date DATE DEFAULT CURRENT_DATE,
    Payment_Mode VARCHAR(20) CHECK(Payment_Mode IN
('Cash','Card','Online')),
    FOREIGN KEY (Booking_ID) REFERENCES Booking(Booking_ID)
);
```

4. Sample Data (INSERT Statements)

```
INSERT INTO Room VALUES
(1, '101', 'Deluxe', 4000.00, 'Available'),
(2, '102', 'Standard', 2500.00, 'Booked'),
(3, '103', 'Deluxe', 4000.00, 'Available'),
(4, '104', 'Suite', 6000.00, 'Booked');


INSERT INTO Customer VALUES
```

```
(1, 'Ravi Kumar', '9876543210', 'Delhi'),

(2, 'Sneha Patel', '8765432109', 'Mumbai');


INSERT INTO Booking VALUES

(101, 1, 2, '2025-10-01', '2025-10-03'),

(102, 2, 4, '2025-10-10', '2025-10-12');


INSERT INTO Payment VALUES

(201, 101, 5000.00, '2025-10-03', 'Card'),

(202, 102, 12000.00, '2025-10-12', 'Online');
```

5. SQL Queries and Outputs

a) List all available rooms of type 'Deluxe':

```
SELECT Room_No, Room_Type, Rate

FROM Room

WHERE Room_Type = 'Deluxe' AND Status = 'Available';
```

b) Display all bookings made in the last 30 days:

```
SELECT *

FROM Booking

WHERE CheckIn_Date >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY);
```

c) Show total revenue collected by each room type:

```
SELECT R.Room_Type, SUM(P.Amount) AS Total_Revenue

FROM Payment P

JOIN Booking B ON P.Booking_ID = B.Booking_ID

JOIN Room R ON B.Room_ID = R.Room_ID

GROUP BY R.Room_Type;
```

6. Normalization up to 3NF

• 1NF: All tables have atomic values and no repeating groups.

• 2NF: All non-key attributes are fully dependent on the primary key.

• 3NF: There are no transitive dependencies — non-key attributes depend only on the primary key.

The database design is normalized up to Third Normal Form (3NF) ensuring data integrity and minimal redundancy.

7. Transaction Management and Consistency

Simultaneous bookings may occur when multiple customers try to reserve the same room at the same time. To handle this, the system uses **transactions** that follow ACID properties:

• **Atomicity** – A booking either completes fully or not at all.

• **Consistency** – Ensures the room status changes correctly upon successful booking.

• **Isolation** – Locks prevent two users from booking the same room simultaneously.

• **Durability** – Once committed, booking data remains safe even after system failures.

Example SQL Transaction:

```
BEGIN TRANSACTION;

UPDATE Room SET Status = 'Booked' WHERE Room_ID = 1 AND Status =
'Available';

INSERT INTO Booking (Booking_ID, Customer_ID, Room_ID, CheckIn_Date,
CheckOut_Date)

VALUES (103, 1, 1, '2025-10-20', '2025-10-22');

COMMIT;
```

8. Conclusion

The Hotel Reservation System effectively manages room availability, bookings, payments, and customer records. By using normalized database structures, proper foreign keys, and transaction control, the system ensures data integrity, avoids redundancy, and handles concurrent operations reliably.