# RENTAL MANAGEMENT SYSTEM

DATABASE MANAGEMENT SYSTEMS PROJECT


**PREPARED BY**

R. OM VENKAT SAI

23CSB0A05

**B. TECH CSE-A**

2023-27

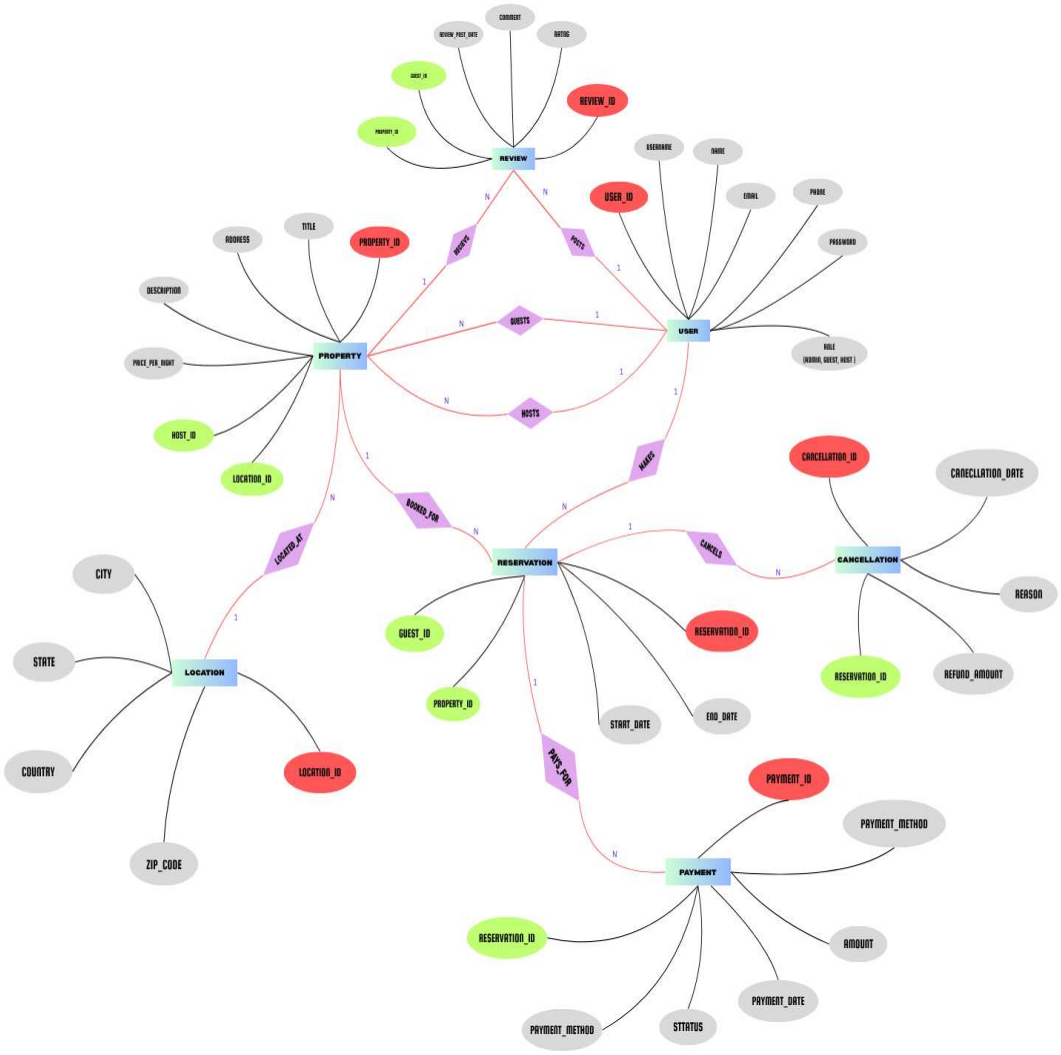
# Contents



## PROJECT OVERVIEW

- The **Rental Management System** is a comprehensive database designed to handle all aspects of property rental and booking.
- It consists of relations for users, properties, locations, reservations, payments, cancellations, reviews, and property restrictions.
- This system enables endless management of property listings by hosts and easy reservation and payment processes for guests.
- With its extensive features, it provides users with efficient tools to list, search, book, pay for, and review rental properties, making it very impactful for hosts, guests, and administrators alike.

# ER DIGRAM (ENTITY RELATIONSHIP DIGRAM)




# ER-DIGRAM EXPLANATION

- **USER:**

  **USER** relation Stores essential details of all users, including **UserID** (primary key), **username**, **name**, **email** (unique), **phone**, and **password**. Each user is assigned a role (Guest, Host, or Admin) enforced via an ENUM. This ensures access control, with admins managing the system, hosts listing properties, and guests making reservations.

- ## LOCATION:

  **LOCATION** relation stores the details about the place where properties listed in the database are located. This relation includes **LocationID** (primary key), **city**, **state**, **country**, and optional **zip code**. This table normalizes location data, avoiding redundancy and allowing properties in the same city to share a single location record.

- ## PROPERTY:

  Represents individual rental properties listed by hosts. Each Property record has a **PropertyID** (primary key), foreign key **HostID** referencing **USER** (who is the host), and a foreign key **LocationID** referencing **LOCATION**. Additional attributes include **title**, **address**, **description**, and **price per night**. The relationship between PROPERTY and USER is many-to-one: each property is owned by exactly one host, while a host can own multiple properties.

- ## RESERVATION:

  **RESERVATION** relation stores bookings made by guests for properties. Each reservation has a **ReservationID** (primary key), **StartDate**, and **EndDate** (with a constraint making sure that StartDate < EndDate). It links to PROPERTY via **PropertyID** and to **USER** (as Guest) via **GuestID**. This models a many-to-one relationship from **RESERVATION** to **PROPERTY** (a property can have many reservations) and to **USER** (a guest can make many reservations). Participation is total for RESERVATION with respect to PROPERTY and USER, as every reservation must be tied to both.

- ## PAYMENT:

  **PAYMENT** relation stores details regarding payments for reservations. Each Payment has a **PaymentID** (primary key), foreign key **ReservationID**, **amount**, **payment date**, **method**, and **status** (with ENUM values SUCCESS, PROCESSING, FAILED). The relationship between

PAYMENT and RESERVATION is many-to-one, allowing each reservation to have multiple associated payments if needed (e.g., installments).

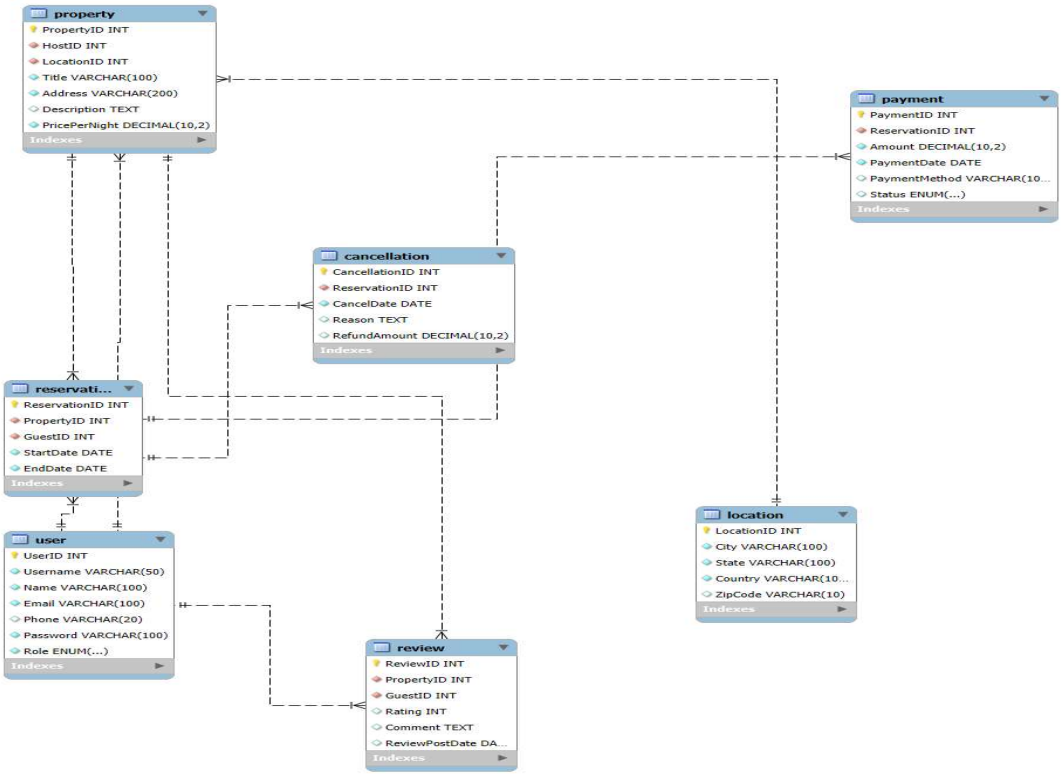
- **CANCELLATION:**
  **CANCELLATION** relation stores information about reservation cancellations. Each Cancellation has a **CancellationID** (primary key), foreign key **ReservationID**, **cancel date**, **reason**, and **refund amount**. This models a one-to-one relationship from RESERVATION to CANCELLATION, as a reservation can have at most one cancellation record.

- **REVIEW:**
  **REVIEW** relation store information regarding guest reviews for properties. Each Review has a **ReviewID** (primary key), foreign keys **PropertyID** and **GuestID** referencing **PROPERTY** and **USER**, respectively, **rating**, **comment**, and **post date**. The relationship between REVIEW and PROPERTY is many-to-one, indicating that a property can have many reviews. Similarly, the relationship to **USER** (as Guest) is many-to-one, allowing each guest to leave multiple reviews.

- **RELATIONSHIPS OVERVIEW:**
  - **USER** hosts **PROPERTY** (1:N), which means one host can list multiple properties but one property can have only one host assuming no sharing properties.
  - **USER** makes **RESERVATION** (1:N), it is allowed for one guest to book many reservations as he/she can, but once reserved that same **reservationID** can be assigned to any other user.
  - **PROPERTY** receives **RESERVATION** (1:N), as a single property can be rented to any number of users at different time stamps.
  - **RESERVATION** is paid via **PAYMENT** (1:N), for a single payment there will be only one reservation, but each reservation may have multiple transaction as there can be failed payments or never successful processing.. payments

- **RESERVATION** may have a **CANCELLATION** (1:1), allowing optional cancellation records.
- **PROPERTY** receives **REVIEW** (1:N), enabling users to give an many review as they want to on the same property.
- **USER** posts **REVIEW** (1:N), supporting many reviews from each guest.
- **PROPERTY** is located at **LOCATION** (M:1), in the same city,country and state there may any number of properties, but a single property can be at a single location.

## RELATIONAL MODEL



**User**:

*Attributes*: userID, username, name, email, phone, password, role
*Primary Key*: userID

Functional Dependencies:

- `userID → (username, name, email, phone, password, role)`

**Location**:

*Attributes*: locationID, city, state, country, zipCode
*Primary Key*: locationID

Functional Dependencies:

- `locationID → (city, state, country, zipCode)`

**Property**:

*Attributes:* propertyID, hostID, locationID, title, address, description, pricePerNight
*Primary Key:* propertyID
*Foreign Keys:* hostID (references User), locationID (references Location)

Functional Dependencies:

- `propertyID → (hostID, locationID, title, address, description, pricePerNight)`

**Reservation**:

*Attributes:* reservationID, propertyID, guestID, startDate, endDate
*Primary Key:* reservationID
*Foreign Keys:* propertyID (references Property), guestID (references User)

Functional Dependencies:

- **reservationID → (propertyID, guestID, startDate, endDate)**

**Payment**:

*Attributes:* paymentID, reservationID, amount, paymentDate, paymentMethod, status
*Primary Key:* paymentID
*Foreign Key:* reservationID (references Reservation)

Functional Dependencies:

- **paymentID → (reservationID, amount, paymentDate, paymentMethod, status)**

**Cancellation**:

*Attributes:* cancellationID, reservationID, cancelDate, reason, refundAmount
*Primary Key:* cancellationID
*Foreign Key:* reservationID (references Reservation)

Functional Dependencies:

- **cancellationID → (reservationID, cancelDate, reason, refundAmount)**

**Review**:

*Attributes:* reviewID, propertyID, guestID, rating, comment, reviewPostDate
*Primary Key:* reviewID
*Foreign Keys:* propertyID (references Property), guestID (references User)

Functional Dependencies:

- **reviewID → (propertyID, guestID, rating, comment, reviewPostDate)**

# NORMALIZATION ANALYSIS (Up to BCNF)

## 1NF (First Normal Form)

- All relations must have atomic values (no repeating groups or arrays).
- Each attribute accommodates holds only one value.
- **All relations satisfy 1NF.**

## 2NF (Second Normal Form)

- All relations must have a **single-attributed primary key** (except Reservation with guestID, propertyID foreign keys—but PK is reservationID).
- No partial dependency on part of a composite key.
- All non-key attributes depend on the full primary key.
- **All tables are in 2NF.**

## 3NF (Third Normal Form)

- No transitive dependencies.
- All non-key attributes depend *only* on the primary key.

- User: all the attributes depend on **userID**.
- Location: all the attributes depend on **locationID**.
- Property: all the attributes depend on **propertyID**.
- Reservation: all the attributes depend on **reservationID**.
- Payment: all the attributes depend on **paymentID**.
- Cancellation: all the attributes depend on **cancellationID**.
- Review: all the attributes depend on **reviewID**.

CONCLUSION : **All tables satisfy 3NF**!

### BCNF (Boyce-Codd Normal Form)

- For every functional dependency **X** → **Y**, X is a **superkey**.
- All tables have primary keys that determine all other attributes.
- No partial or transitive dependencies.
- No anomalies from functional dependencies violating BCNF.

CONCLUSION : **All tables satisfy BCNF**!

## DATABASE SCHEMA WITH DATA AND QUERIES:

### Database SetUp:

```
CREATE DATABASE IF NOT EXISTS RentalManagementDB;
USE RentalManagementDB;
```

### Creating Tables: (SCHEMAS of each Relation)

```
CREATE TABLE USER
(
  UserID INT AUTO_INCREMENT PRIMARY KEY,
  Username VARCHAR(50) UNIQUE NOT NULL,
  Name VARCHAR(100) NOT NULL,
  Email VARCHAR(100) UNIQUE NOT NULL,
  Phone VARCHAR(20),
  Password VARCHAR(100) NOT NULL,
  Role ENUM('HOST', 'GUEST', 'ADMIN') NOT NULL
);
```

```sql
CREATE TABLE LOCATION
(
  LocationID INT AUTO_INCREMENT PRIMARY KEY,
  City VARCHAR(100) NOT NULL,
  State VARCHAR(100) NOT NULL,
  Country VARCHAR(100) NOT NULL,
  ZipCode VARCHAR(10)
);

CREATE TABLE PROPERTY
(
  PropertyID INT AUTO_INCREMENT PRIMARY KEY,
  HostID INT NOT NULL,
  LocationID INT NOT NULL,
  Title VARCHAR(100) NOT NULL,
  Address VARCHAR(200) NOT NULL,
  Description TEXT,
  PricePerNight DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (HostID) REFERENCES USER(UserID) ON DELETE CASCADE,
  FOREIGN KEY (LocationID) REFERENCES LOCATION(LocationID) ON DELETE CASCADE
);

CREATE TABLE RESERVATION
(
  ReservationID INT AUTO_INCREMENT PRIMARY KEY,
  PropertyID INT NOT NULL,
  GuestID INT NOT NULL,
  StartDate DATE NOT NULL,
  EndDate DATE NOT NULL,
  FOREIGN KEY (PropertyID) REFERENCES PROPERTY(PropertyID) ON DELETE CASCADE,
  FOREIGN KEY (GuestID) REFERENCES USER(UserID) ON DELETE CASCADE,
  CHECK(StartDate < EndDate)
);

CREATE TABLE PAYMENT
(
  PaymentID INT AUTO_INCREMENT PRIMARY KEY,
  ReservationID INT NOT NULL,
  Amount DECIMAL(10, 2) NOT NULL,
  PaymentDate DATE NOT NULL,
  PaymentMethod VARCHAR(100),
  Status ENUM('SUCCESS', 'PROCESSING', 'FAILED') DEFAULT 'PROCESSING',
  FOREIGN KEY (ReservationID) REFERENCES RESERVATION(ReservationID) ON DELETE
CASCADE
);



CREATE TABLE CANCELLATION
(
```

```
  CancellationID INT AUTO_INCREMENT PRIMARY KEY,
  ReservationID INT NOT NULL,
  CancelDate DATE NOT NULL,
  Reason TEXT,
  RefundAmount DECIMAL(10, 2) DEFAULT 0.00,
  FOREIGN KEY (ReservationID) REFERENCES RESERVATION(ReservationID) ON DELETE
CASCADE
);


CREATE TABLE REVIEW
(
  ReviewID INT AUTO_INCREMENT PRIMARY KEY,
  PropertyID INT NOT NULL,
  GuestID INT NOT NULL,
  Rating INT CHECK (Rating BETWEEN 1 AND 5),
  Comment TEXT,
  ReviewPostDate DATE,
  FOREIGN KEY (PropertyID) REFERENCES PROPERTY(PropertyID) ON DELETE CASCADE,
  FOREIGN KEY (GuestID) REFERENCES USER(UserID) ON DELETE CASCADE
);
```

## Inserting data into Tables:

- USER Table data:

```
INSERT INTO USER (Username, Name, Email, Phone, Password, Role) VALUES
('rahulhost', 'Rahul Sharma', 'rahul.sharma@gmail.com', '9876543210', 'rahulpass',
'HOST'),
('sneha_guest', 'Sneha Iyer', 'sneha.iyer@gmail.com', '9123456780', 'snehapass',
'GUEST'),
('arvindhost', 'Arvind Kumar', 'arvind.kumar@gmail.com', '9988776655', 'arvindpass',
'HOST'),
('priya_guest', 'Priya Singh', 'priya.singh@gmail.com', '9876501234', 'priyapass',
'GUEST'),
('neerajhost', 'Neeraj Patel', 'neeraj.patel@gmail.com', '9876512345', 'neerajpass',
'HOST'),
('akash_guest', 'Akash Verma', 'akash.verma@gmail.com', '8765432109', 'akashpass',
'GUEST'),
('divya_host', 'Divya Menon', 'divya.menon@gmail.com', '9988123456', 'divyapass',
'HOST'),
('vijay_guest', 'Vijay Reddy', 'vijay.reddy@gmail.com', '9123498765', 'vijaypass',
'GUEST'),
('pallavi_host', 'Pallavi Desai', 'pallavi.desai@gmail.com', '9876540987',
'pallavipass', 'HOST'),
('adminuser', 'Admin User', 'admin@gmail.com', '9000000000', 'adminpass', 'ADMIN');
```

- LOCATION Table data:

```
INSERT INTO LOCATION (City, State, Country, ZipCode) VALUES
('Bengaluru', 'Karnataka', 'India', '560001'),
('Hyderabad', 'Telangana', 'India', '500001'),
('Mumbai', 'Maharashtra', 'India', '400001'),
('Chennai', 'Tamil Nadu', 'India', '600001'),
('New Delhi', 'Delhi', 'India', '110001'),
('Pune', 'Maharashtra', 'India', '411001'),
('Kolkata', 'West Bengal', 'India', '700001'),
('Ahmedabad', 'Gujarat', 'India', '380001'),
('Jaipur', 'Rajasthan', 'India', '302001'),
('Lucknow', 'Uttar Pradesh', 'India', '226001');
```

- PROPERTY Table data:

```
INSERT INTO PROPERTY (HostID, LocationID, Title, Address, Description,
PricePerNight) VALUES
(1, 1, '2BHK Apartment', 'MG Road, Bengaluru', 'Spacious 2BHK near city center',
3200.00),
(3, 2, 'Modern Studio', 'Madhapur, Hyderabad', 'Well-designed studio with AC',
2500.00),
(5, 3, 'Sea View Apartment', 'Marine Drive, Mumbai', 'Beautiful sea-facing flat',
4500.00),
(7, 4, 'Chettinad House', 'Mylapore, Chennai', 'Heritage house with courtyard',
2800.00),
(1, 5, 'Luxorius Flat', 'Connaught Place, New Delhi', 'Premium apartment in heart of
the city', 5200.00),
(3, 6, 'Budget Friendly Room', 'FC Road, Pune', 'Affordable stay with all basic
facilities', 1500.00),
(5, 7, 'Riverside Bunglow', 'Howrah, Kolkata', 'Spacious Bunglow by the river',
6000.00),
(7, 8, 'City Center Property', 'CG Road, Ahmedabad', 'Walkable to shops and
restaurants', 2700.00),
(9, 9, 'Jaipur heritage Bunglow', 'Amer Road, Jaipur', 'Royal experience in heritage
Bunglow', 4000.00),
(9, 10, 'Guesthouse in Lucknow', 'Hazratganj, Lucknow', 'Comfortable and centrally
located', 2300.00),
(3, 3, '3BHK Flat', 'Goregaon East, Mumbai', 'Spacious 3BHK near filmc city',
4500.00),
(1, 1, 'No-Booking Loft', 'Church Street, Bengaluru', 'Modern loft no bookings yet',
4000.00),
(3, 3, 'Empty Studio', 'Bandra, Mumbai', 'Studio apartment waiting for first guest',
2800.00);
```

- RESERVATION Table data:

```
INSERT INTO RESERVATION (PropertyID, GuestID, StartDate, EndDate) VALUES
(1, 2, '2024-09-22', '2024-09-25'),
(2, 4, '2024-10-15', '2024-10-17'),
(3, 6, '2024-11-02', '2024-11-04'),
(4, 8, '2024-12-08', '2024-12-10'),
(5, 2, '2025-01-01', '2025-01-03'),
(6, 4, '2025-01-05', '2025-01-07'),
(7, 6, '2025-02-10', '2025-02-12'),
(8, 8, '2025-03-15', '2025-03-20'),
(9, 2, '2025-04-25', '2025-04-28'),
(10,4, '2025-05-01', '2025-05-05'),
(3, 2, '2025-06-10', '2025-06-13'),
(2, 4, '2025-06-15', '2025-06-17'),
(10, 4, '2025-07-01', '2025-07-03'),
(2, 2, '2025-08-01', '2025-08-05'),
(3, 2, '2025-09-10', '2025-09-12'),
(4, 4, '2025-08-15', '2025-08-18'),
(1, 2, '2023-05-10', '2023-05-15'),
(2, 4, '2025-10-20', '2025-10-23'),
(3, 6, '2022-01-10', '2022-01-15');
```

- PAYMENT Table data:

```
INSERT INTO PAYMENT (ReservationID, Amount, PaymentDate, PaymentMethod, Status)
VALUES
(1, 9600.00, '2024-09-19', 'UPI', 'SUCCESS'),
(2, 5100.00, '2024-10-11', 'Credit Card', 'SUCCESS'),
(3, 13750.00, '2024-10-30', 'Net Banking', 'SUCCESS'),
(4, 14500.00, '2024-12-04', 'UPI', 'SUCCESS'),
(5, 10450.00, '2024-12-29', 'Credit Card', 'SUCCESS'),
(6, 3100.00, '2025-01-03', 'UPI', 'PROCESSING'),
(7, 12200.00, '2025-02-04', 'Net Banking', 'SUCCESS'),
(8, 16500.00, '2025-03-10', 'Credit Card', 'SUCCESS'),
(9, 9400.00, '2025-04-21', 'UPI', 'FAILED'),
(9, 9400.00, '2025-04-21', 'Credit Card', 'SUCCESS'),
(10, 14050.00, '2025-04-27', 'UPI', 'SUCCESS'),
(14, 10000.00, '2025-07-25', 'Credit Card', 'SUCCESS'),
(15, 9500.00, '2025-09-01', 'UPI', 'PROCESSING'),
(16, 12000.00, '2025-08-10', 'Credit Card', 'FAILED'),
(6, 3100.00, DATE_SUB(CURDATE(), INTERVAL 400 DAY), 'UPI', 'PROCESSING'), -- fake
data to test the last query of the queries list
(17, 8000.00, '2023-05-05', 'UPI', 'PROCESSING');
```

- CANCELLATION Table data:

```
INSERT INTO CANCELLATION (ReservationID, CancelDate, Reason, RefundAmount) VALUES
(1, '2024-09-18', 'Change in travel plans', 9600.00),
(2, '2024-10-14', 'Found better option', 0.00),
(3, '2024-10-31', 'Unexpected emergency', 13750.00),
(5, '2024-12-27', 'Personal reasons', 5225.00),
(6, '2025-01-01', 'Schedule conflict', 3100.00),
(14, '2025-07-28', 'Change in plan', 10000.00),
(15, '2025-09-02', 'Work conflict', 4750.00),
(16, '2025-08-11', 'Emergency', 3000.00);
```

- REVIEW Table data:

```
INSERT INTO REVIEW (PropertyID, GuestID, Rating, Comment, ReviewPostDate) VALUES
(1, 2, 5, 'Amazing stay in Bengaluru, super clean!', '2024-09-25'),
(2, 4, 4, 'Nice location in Hyderabad.', '2024-10-18'),
(3, 6, 5, 'Seaview is worth every penny.', '2024-11-04'),
(4, 8, 3, 'Chennai home is nice but needs upkeep.', '2024-12-11'),
(5, 2, 5, 'Luxury experience in Delhi!', '2025-01-03'),
(6, 4, 4, 'The room was neat and clean.', '2025-01-08'),
(7, 6, 5, 'Riverside view made my day.', '2025-02-12'),
(8, 8, 4, 'Very convenient and sweet place to visit.', '2025-03-21'),
(9, 2, 5, 'Loved the royal bungalow in Jaipur.', '2025-04-28'),
(10, 4, 4, 'Guesthouse was very comfy.', '2025-05-06'),
(2, 2, 2, 'Not as clean as expected.', '2025-08-06'),
(3, 2, 3, 'Decent stay.', '2025-09-12'),
(4, 4, 5, 'Loved the heritage vibes!', '2025-08-18');
```

## Reading Data From Relations:

```
SELECT * FROM USER;
SELECT * FROM LOCATION;
SELECT * FROM PROPERTY;
SELECT * FROM  RESERVATION;
SELECT * FROM  PAYMENT;
SELECT * FROM CANCELLATION;
SELECT * FROM REVIEW;
```

## Initial data display:

- USER data:

| UserID | Username | Name | Email | Phone | Password | Role |
|---|---|---|---|---|---|---|
| 1 | rahulhost | Rahul Sharma | rahul.sharma@gmail.com | 9876543210 | rahulpass | HOST |
| 2 | sneha_guest | Sneha Iyer | sneha.iyer@gmail.com | 9123456780 | snehapass | GUEST |
| 3 | arvindhost | Arvind Kumar | arvind.kumar@gmail.com | 9988776655 | arvindpass | HOST |
| 4 | priya_guest | Priya Singh | priya.singh@gmail.com | 9876501234 | priyapass | GUEST |
| 5 | neerajhost | Neeraj Patel | neeraj.patel@gmail.com | 9876512345 | neerajpass | HOST |
| 6 | akash_guest | Akash Verma | akash.verma@gmail.com | 8765432109 | akashpass | GUEST |
| 7 | divya_host | Divya Menon | divya.menon@gmail.com | 9988123456 | divyapass | HOST |
| 8 | vijay_guest | Vijay Reddy | vijay.reddy@gmail.com | 9123498765 | vijaypass | GUEST |
| 9 | pallavi_host | Pallavi Desai | pallavi.desai@gmail.com | 9876540987 | pallavipass | HOST |
| 10 | adminuser | Admin User | admin@gmail.com | 9000000000 | adminpass | ADMIN |

- LOCATION data:

| LocationID | City | State | Country | ZipCode |
|---|---|---|---|---|
| 1 | Bengaluru | Karnataka | India | 560001 |
| 2 | Hyderabad | Telangana | India | 500001 |
| 3 | Mumbai | Maharashtra | India | 400001 |
| 4 | Chennai | Tamil Nadu | India | 600001 |
| 5 | New Delhi | Delhi | India | 110001 |
| 6 | Pune | Maharashtra | India | 411001 |
| 7 | Kolkata | West Bengal | India | 700001 |
| 8 | Ahmedabad | Gujarat | India | 380001 |
| 9 | Jaipur | Rajasthan | India | 302001 |
| 10 | Lucknow | Uttar Pradesh | India | 226001 |

- PROPERTY data:

| PropertyID | HostID | LocationID | Title | Address | Description | PricePerNight |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2BHK Apartment | MG Road, Bengaluru | Spacious 2BHK near city center | 3200.00 |
| 2 | 3 | 2 | Modern Studio | Madhapur, Hyderabad | Well-designed studio with AC | 2500.00 |
| 3 | 5 | 3 | Sea View Apartment | Marine Drive, Mumbai | Beautiful sea-facing flat | 4500.00 |
| 4 | 7 | 4 | Chettinad House | Mylapore, Chennai | Heritage house with courtyard | 2800.00 |
| 5 | 1 | 5 | Luxorius Flat | Connaught Place, New Delhi | Premium apartment in heart of the city | 5200.00 |
| 6 | 3 | 6 | Budget Friendly Room | FC Road, Pune | Affordable stay with all basic facilities | 1500.00 |
| 7 | 5 | 7 | Riverside Bunglow | Howrah, Kolkata | Spacious Bunglow by the river | 6000.00 |
| 8 | 7 | 8 | City Center Property | CG Road, Ahmedabad | Walkable to shops and restaurants | 2700.00 |
| 9 | 9 | 9 | Jaipur heritage Bunglow | Amer Road, Jaipur | Royal experience in heritage Bunglow | 4000.00 |
| 10 | 9 | 10 | Guesthouse in Lucknow | Hazra | Comfortable and centrally located | 2300.00 |
| 11 | 3 | 3 | 3BHK Flat | Goregaon East, Mumbai | Spacious 3BHK near filmc city | 4500.00 |
| 12 | 1 | 1 | No-Booking Loft | Church Street, Bengaluru | Modern loft no bookings yet | 4000.00 |
| 13 | 3 | 3 | Empty Studio | Bandra, Mumbai | Studio apartment waiting for first guest | 2800.00 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Hazratganj, Lucknow

- RESERVATION data:

| ReservationID | PropertyID | GuestID | StartDate | EndDate |
|---|---|---|---|---|
| 1 | 1 | 2 | 2024-09-22 | 2024-09-25 |
| 2 | 2 | 4 | 2024-10-15 | 2024-10-17 |
| 3 | 3 | 6 | 2024-11-02 | 2024-11-04 |
| 4 | 4 | 8 | 2024-12-08 | 2024-12-10 |
| 5 | 5 | 2 | 2025-01-01 | 2025-01-03 |
| 6 | 6 | 4 | 2025-01-05 | 2025-01-07 |
| 7 | 7 | 6 | 2025-02-10 | 2025-02-12 |
| 8 | 8 | 8 | 2025-03-15 | 2025-03-20 |
| 9 | 9 | 2 | 2025-04-25 | 2025-04-28 |
| 10 | 10 | 4 | 2025-05-01 | 2025-05-05 |
| 11 | 3 | 2 | 2025-06-10 | 2025-06-13 |
| 12 | 2 | 4 | 2025-06-15 | 2025-06-17 |
| 13 | 10 | 4 | 2025-07-01 | 2025-07-03 |
| 14 | 2 | 2 | 2025-08-01 | 2025-08-05 |
| 15 | 3 | 2 | 2025-09-10 | 2025-09-12 |
| 16 | 4 | 4 | 2025-08-15 | 2025-08-18 |
| 17 | 1 | 2 | 2023-05-10 | 2023-05-15 |
| 18 | 2 | 4 | 2025-10-20 | 2025-10-23 |
| 19 | 3 | 6 | 2022-01-10 | 2022-01-15 |
| NULL | NULL | NULL | NULL | NULL |

- PAYMENT data:

| PaymentID | ReservationID | Amount | PaymentDate | PaymentMethod | Status |
|---|---|---|---|---|---|
| 1 | 1 | 9600.00 | 2024-09-19 | UPI | SUCCESS |
| 2 | 2 | 5100.00 | 2024-10-11 | Credit Card | SUCCESS |
| 3 | 3 | 13750.00 | 2024-10-30 | Net Banking | SUCCESS |
| 4 | 4 | 14500.00 | 2024-12-04 | UPI | SUCCESS |
| 5 | 5 | 10450.00 | 2024-12-29 | Credit Card | SUCCESS |
| 6 | 6 | 3100.00 | 2025-01-03 | UPI | PROCESSING |
| 7 | 7 | 12200.00 | 2025-02-04 | Net Banking | SUCCESS |
| 8 | 8 | 16500.00 | 2025-03-10 | Credit Card | SUCCESS |
| 9 | 9 | 9400.00 | 2025-04-21 | UPI | FAILED |
| 10 | 9 | 9400.00 | 2025-04-21 | Credit Card | SUCCESS |
| 11 | 10 | 14050.00 | 2025-04-27 | UPI | SUCCESS |
| 12 | 14 | 10000.00 | 2025-07-25 | Credit Card | SUCCESS |
| 13 | 15 | 9500.00 | 2025-09-01 | UPI | PROCESSING |
| 14 | 16 | 12000.00 | 2025-08-10 | Credit Card | FAILED |
| 15 | 6 | 3100.00 | 2024-05-24 | UPI | PROCESSING |
| 16 | 17 | 8000.00 | 2023-05-05 | UPI | PROCESSING |

- CANCELLATION data:

| CancellationID | ReservationID | CancelDate | Reason | RefundAmount |
|---|---|---|---|---|
| 1 | 1 | 2024-09-18 | Change in travel plans | 9600.00 |
| 2 | 2 | 2024-10-14 | Found better option | 0.00 |
| 3 | 3 | 2024-10-31 | Unexpected emergency | 13750.00 |
| 4 | 5 | 2024-12-27 | Personal reasons | 5225.00 |
| 5 | 6 | 2025-01-01 | Schedule conflict | 3100.00 |
| 6 | 14 | 2025-07-28 | Change in plan | 10000.00 |
| 7 | 15 | 2025-09-02 | Work conflict | 4750.00 |
| 8 | 16 | 2025-08-11 | Emergency | 3000.00 |

- REVIEW data:

| ReviewID | PropertyID | GuestID | Rating | Comment | ReviewPostDate |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 5 | Amazing stay in Bengaluru, super clean! | 2024-09-25 |
| 2 | 2 | 4 | 4 | Nice location in Hyderabad. | 2024-10-18 |
| 3 | 3 | 6 | 5 | Seaview is worth every penny. | 2024-11-04 |
| 4 | 4 | 8 | 3 | Chennai home is nice but needs upkeep. | 2024-12-11 |
| 5 | 5 | 2 | 5 | Luxury experience in Delhi! | 2025-01-03 |
| 6 | 6 | 4 | 4 | The room was neat and clean. | 2025-01-08 |
| 7 | 7 | 6 | 5 | Riverside view made my day. | 2025-02-12 |
| 8 | 8 | 8 | 4 | Very convenient and sweet place to visit. | 2025-03-21 |
| 9 | 9 | 2 | 5 | Loved the royal bungalow in Jaipur. | 2025-04-28 |
| 10 | 10 | 4 | 4 | Guesthouse was very comfy. | 2025-05-06 |
| 11 | 2 | 2 | 2 | Not as clean as expected. | 2025-08-06 |
| 12 | 3 | 2 | 3 | Decent stay. | 2025-09-12 |
| 13 | 4 | 4 | 5 | Loved the heritage vibes! | 2025-08-18 |

## Some Demo Queries:

- List all properties with their host names and locations

```
SELECT P.PropertyID as Property_Id, U.Name AS Name, P.Title, P.PricePerNight,
L.City, L.State, L.Country
FROM
    PROPERTY P
    JOIN USER U ON P.HostID = U.UserID
    JOIN LOCATION L ON P.LocationID = L.LocationID
ORDER BY
    L.City,P.Title;
```

**OUTPUT:**

| Property_Id | Name | Title | PricePerNight | City | State | Country |
|---|---|---|---|---|---|---|
| 8 | Divya Menon | City Center Property | 2700.00 | Ahmedabad | Gujarat | India |
| 1 | Rahul Sharma | 2BHK Apartment | 3200.00 | Bengaluru | Karnataka | India |
| 12 | Rahul Sharma | No-Booking Loft | 4000.00 | Bengaluru | Karnataka | India |
| 4 | Divya Menon | Chettinad House | 2800.00 | Chennai | Tamil Nadu | India |
| 2 | Arvind Kumar | Modern Studio | 2500.00 | Hyderabad | Telangana | India |
| 9 | Pallavi Desai | Jaipur heritage Bunglow | 4000.00 | Jaipur | Rajasthan | India |
| 7 | Neeraj Patel | Riverside Bunglow | 6000.00 | Kolkata | West Bengal | India |
| 10 | Pallavi Desai | Guesthouse in Lucknow | 2300.00 | Lucknow | Uttar Pradesh | India |
| 11 | Arvind Kumar | 3BHK Flat | 4500.00 | Mumbai | Maharashtra | India |
| 13 | Arvind Kumar | Empty Studio | 2800.00 | Mumbai | Maharashtra | India |
| 3 | Neeraj Patel | Sea View Apartment | 4500.00 | Mumbai | Maharashtra | India |
| 5 | Rahul Sharma | Luxorius Flat | 5200.00 | New Delhi | Delhi | India |
| 6 | Arvind Kumar | Budget Friendly Room | 1500.00 | Pune | Maharashtra | India |

- Show reservations with guest names and property titles.

```
SELECT R.ReservationId, U.Name as Name, P.Title
FROM
    RESERVATION R
    JOIN USER U ON R.GuestID = U.UserID
    JOIN PROPERTY P ON R.PropertyID = P.PropertyID;
```

**OUTPUT:**

| ReservationId | Name | Title |
|---|---|---|
| 1 | Sneha Iyer | 2BHK Apartment |
| 5 | Sneha Iyer | Luxorius Flat |
| 9 | Sneha Iyer | Jaipur heritage Bunglow |
| 11 | Sneha Iyer | Sea View Apartment |
| 14 | Sneha Iyer | Modern Studio |
| 15 | Sneha Iyer | Sea View Apartment |
| 17 | Sneha Iyer | 2BHK Apartment |
| 2 | Priya Singh | Modern Studio |
| 6 | Priya Singh | Budget Friendly Room |
| 10 | Priya Singh | Guesthouse in Lucknow |
| 12 | Priya Singh | Modern Studio |
| 13 | Priya Singh | Guesthouse in Lucknow |
| 16 | Priya Singh | Chettinad House |
| 18 | Priya Singh | Modern Studio |
| 3 | Akash Verma | Sea View Apartment |
| 7 | Akash Verma | Riverside Bunglow |
| 19 | Akash Verma | Sea View Apartment |
| 4 | Vijay Reddy | Chettinad House |
| 8 | Vijay Reddy | City Center Property |

- List all properties along with their reservation count [including properties with zero reservations]

```
SELECT P.PropertyID, P.Title, COUNT(R.ReservationID) AS COUNT
FROM
    PROPERTY P
    LEFT JOIN RESERVATION R ON P.PropertyID = R.PropertyID
GROUP BY P.propertyID;
```

**OUTPUT:**

| PropertyID | Title | COUNT |
|---|---|---|
| 1 | 2BHK Apartment | 2 |
| 2 | Modern Studio | 4 |
| 3 | Sea View Apartment | 4 |
| 4 | Chettinad House | 2 |
| 5 | Luxorius Flat | 1 |
| 6 | Budget Friendly Room | 1 |
| 7 | Riverside Bunglow | 1 |
| 8 | City Center Property | 1 |
| 9 | Jaipur heritage Bunglow | 1 |
| 10 | Guesthouse in Lucknow | 2 |
| 11 | 3BHK Flat | 0 |
| 12 | No-Booking Loft | 0 |
| 13 | Empty Studio | 0 |

- List average rating for each property.

```
SELECT P.PropertyID, P.Title, AVG(R.Rating) AS AVERGAE
FROM
    PROPERTY P
    JOIN REVIEW R ON R.PropertyID = P.PropertyID
GROUP BY P.PropertyID;
```

**OUTPUT:**

| | PropertyID | Title | AVERGAE |
|---|---|---|---|
| | 1 | 2BHK Apartment | 5.0000 |
| | 2 | Modern Studio | 3.0000 |
| | 3 | Sea View Apartment | 4.0000 |
| | 4 | Chettinad House | 4.0000 |
| | 5 | Luxorius Flat | 5.0000 |
| | 6 | Budget Friendly Room | 4.0000 |
| | 7 | Riverside Bunglow | 5.0000 |
| | 8 | City Center Property | 4.0000 |
| | 9 | Jaipur heritage Bunglow | 5.0000 |
| | 10 | Guesthouse in Lucknow | 4.0000 |

- Show total payment amount received per property. [After removing the Refund Amount if the transaction got cancelled in between ]

```
WITH
T AS
(
    SELECT R.ReservationID, SUM(Amount) AS Amount
    FROM
        RESERVATION R
        JOIN PAYMENT P ON R.ReservationID = P.ReservationID
    GROUP BY R.ReservationID
),
T1 AS
(
    SELECT P.PropertyID, T.ReservationID,
            CASE
                WHEN Amount IS NULL THEN 0
                ELSE Amount
            END AS Amount
    FROM PROPERTY P
    LEFT JOIN T ON P.PropertyID = (SELECT PropertyID FROM RESERVATION WHERE
ReservationID = T.ReservationID)
),
T2 AS
(
    SELECT T1.PropertyID, T1.Amount,
        CASE
            WHEN C.RefundAmount IS NULL THEN 0
            ELSE C.RefundAmount
        END AS RefundAmount
    FROM T1
    LEFT JOIN CANCELLATION C ON C.ReservationID = T1.ReservationID
)
SELECT PropertyID, Amount-RefundAmount AS Net_Amount
FROM T2
ORDER BY PropertyID ASC;
```

**OUTPUT:**

| PropertyID | Net_Amount |
|---|---|
| 1 | 8000.00 |
| 1 | 0.00 |
| 2 | 0.00 |
| 2 | 5100.00 |
| 3 | 4750.00 |
| 3 | 0.00 |
| 4 | 9000.00 |
| 4 | 14500.00 |
| 5 | 5225.00 |
| 6 | 3100.00 |
| 7 | 12200.00 |
| 8 | 16500.00 |
| 9 | 18800.00 |
| 10 | 14050.00 |
| 11 | 0.00 |
| 12 | 0.00 |
| 13 | 0.00 |

- Identify loyal or repeat customers. [in future if possible we can give a special discount to these].

```
SELECT U.Name, COUNT(ReservationID) AS VISIT_COUNT
FROM USER U
    JOIN RESERVATION R ON U.UserID = R.GuestID
WHERE U.Role = 'GUEST'
GROUP BY U.UserID
HAVING VISIT_COUNT > 2;
```

**OUTPUT:**

| Name | VISIT_COUNT |
|---|---|
| Sneha Iyer | 7 |
| Priya Singh | 7 |
| Akash Verma | 3 |

- Top 5 Properties by Review Rating.

```
SELECT DENSE_RANK() OVER (ORDER BY AVG(Rating) DESC) AS RNK,
P.PropertyID, P.Title, AVG(Rating) AS AVG_RATING
FROM REVIEW R
    JOIN PROPERTY P ON R.PropertyID = P.PropertyID
GROUP BY P.PropertyID
ORDER BY RNK
LIMIT 5;
```

**OUTPUT:**

| RNK | PropertyID | Title | AVG_RATING |
|---|---|---|---|
| 1 | 1 | 2BHK Apartment | 5.0000 |
| 1 | 5 | Luxorius Flat | 5.0000 |
| 1 | 7 | Riverside Bunglow | 5.0000 |
| 1 | 9 | Jaipur heritage Bunglow | 5.0000 |
| 2 | 3 | Sea View Apartment | 4.0000 |

- Top 5 most expensive properties.

```
SELECT DENSE_RANK() OVER (ORDER BY PricePerNight DESC) AS RNK,
P.PropertyID, P.Title, PricePerNight
FROM PROPERTY P
ORDER BY RNK
LIMIT 5;
```

**OUTPUT:**

| RNK | PropertyID | Title | PricePerNight |
|---|---|---|---|
| 1 | 7 | Riverside Bunglow | 6000.00 |
| 2 | 5 | Luxorius Flat | 5200.00 |
| 3 | 3 | Sea View Apartment | 4500.00 |
| 3 | 11 | 3BHK Flat | 4500.00 |
| 4 | 9 | Jaipur heritage Bunglow | 4000.00 |

- procedure to update the property only by host.

```
DELIMITER $$
CREATE PROCEDURE UpdateProperty(
    IN inHostID INT,
    IN inPropertyID INT,
    IN inTitle VARCHAR(100),
    IN inAddress VARCHAR(200),
    IN inDescription TEXT,
    IN inPricePerNight DECIMAL(10, 2)
)
BEGIN
    IF (SELECT COUNT(*) FROM USER WHERE UserID = inHostID and Role = 'HOST') = 0
THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'UNAUTHORIZED USER'; -- CUSTOM
ERROR
    ELSE
        UPDATE PROPERTY
        SET
            Title = inTitle,
            Address = inAddress,
            Description = inDescription,
            PricePerNight = inPricePerNight
        WHERE PropertyID = inPropertyID AND HostID = inHostID;
    END IF;
END$$
DELIMITER ;
```

- Delete old failed payments about 1 year before [made a procedure as it requires authentication].

```
DELIMITER $$
CREATE PROCEDURE DeleteOldFailedPayments(IN inAdminID INT)
BEGIN
    IF (SELECT COUNT(*) FROM USER WHERE UserID = inAdminID and Role = 'ADMIN') = 0
THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'UNAUTHORIZED USER'; -- CUSTOM
ERROR
    ELSE
        DELETE FROM PAYMENT
        WHERE PaymentID IN (
            SELECT PaymentID
            FROM (
                SELECT PaymentID
```

```
                FROM PAYMENT
                WHERE (Status = 'FAILED' OR Status = 'PROCESSING')
                    AND (PaymentDate < DATE_SUB(CURDATE(), INTERVAL 1 YEAR))
            ) AS sub
        );
    END IF;
END$$
DELIMITER ;
```

- procedure for cancelling with automatic refund.

```
DELIMITER $$
CREATE PROCEDURE cancelReservation(IN resID INT, IN reason TEXT)
BEGIN

    -- our local variables
    DECLARE start_date DATE;
    DECLARE curr_amount DECIMAL(10,2);
    DECLARE Refund DECIMAL(10,2);

    IF (SELECT COUNT(*) FROM RESERVATION WHERE resID = ReservationID) > 0 THEN
        SELECT StartDate into start_date FROM RESERVATION WHERE ReservationID =
resID;
        SELECT P.Amount into curr_amount FROM PAYMENT P WHERE P.ReservationID =
resID;
        IF DATEDIFF(start_date, CURDATE()) >= 7 THEN
            SET Refund = curr_amount;
        ELSEIF DATEDIFF(start_date, CURDATE()) >= 5 THEN
            SET Refund = curr_amount/2;
        ELSEIF DATEDIFF(start_date, CURDATE()) >= 2 THEN
            SET Refund = curr_amount/4;
        ELSE
            SET Refund = 0;
        END IF;
    END IF;

    INSERT INTO CANCELLATION (ReservationID, CancelDate, Reason, RefundAmount)
    VALUES (resID, CURDATE(), reason, Refund);

END$$
DELIMITER ;
```

## @ DESTROYING STALE DATA WITH IN A PARTICULAR TIME

- delete all the payments whose status is processing and the deadline crossed

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM PAYMENT
WHERE Status = 'PROCESSING'
    AND ReservationID IN (
        SELECT ReservationID
        FROM RESERVATION
        WHERE StartDate <= CURDATE()
    );
SET SQL_SAFE_UPDATES = 1;
SELECT * FROM PAYMENT;
```

- delete user who reserved a property but did not go for payment with in 1 day

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM RESERVATION
WHERE reservationID NOT IN (
        SELECT reservationID
        FROM PAYMENT
    );
SET SQL_SAFE_UPDATES = 1;
```

- delete user whose endDate has gone past one year ago

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM RESERVATION
WHERE CURDATE() >= DATE_ADD(endDate, INTERVAL 1 YEAR) ;
SET SQL_SAFE_UPDATES = 1;
```

# Conclusion:

This project successfully demonstrates most of the design and implementation of a Rental Management System using a relational database. The system incorporates essential entities such as Users, Properties, Reservations, Payments, Cancellations, Locations, and Reviews, ensuring data integrity through the use of foreign keys and normalization up to BCNF.

The developed ER diagram and normalized schema help to eliminate redundancy and maintain consistency across all operations, such as booking, payment processing, and cancellations.

Overall, this database solution offers a scalable and efficient backbone for any property rental application, supporting both hosts and guests in managing their bookings with ease.