

# Hotel Reservation Management System

## Brief Description of the Designed Database

The designed database is a system for managing a hotel reservation platform. It covers various aspects of hotel operations, including guest management, room reservations, payment processing, inventory control and other related operations. The database facilitates efficient data retrieval and updates, ensuring that all hotel operations are synchronized and accessible in real-time.

## Motivation for Selecting This Project Topic

The motivation for selecting a hotel reservation system comes from the critical role that data management plays in the hospitality industry. Hotels rely heavily on accurate and efficient data handling to enhance guest experiences, optimize resource utilization, and remain competitive. Designing a database for such a system presents an opportunity to address real-world challenges in data organization, integrity, and scalability. It allows for the application of database design principles to create a solution that can significantly improve operational efficiency and customer satisfaction in a dynamic industry.

## How This Database Can Be Used

This database is designed assuming the primary users of the system are internal staff of the hotel who would be handling reservations and related things. It is assumed that it makes reservations either in-person or via phone calls through its representatives. The various internal users who could use this database and the use case include:

- **Front Desk Staff:** Manage reservations, check-ins, and check-outs efficiently with real-time access to room availability and guest information.
- **Internal Employees:** Look for maintenance issues, check on inventory and supplier
- **Managers:** can look at various reports generated from by the system to make an informed decision.

## User Requirements

The system stores information about **guests** who make reservations for rooms. The details of a guest to be stored are first name, last name, email, phone number, and address. Address consists of street, city, state and zip code. A guest is uniquely identified by their email.

**Reservations** are managed by the system, capturing the check-in and check-out dates, the room booked, and the reservation status (e.g., Booked, Checked-in, Canceled).

The system maintains data on each **room**, including its unique number, floor, and status (e.g., Available, Occupied, Under Maintenance), room images.

Different **room types** are defined within the system, describing each type (e.g., Single, Double, Suite) along with the maximum occupancy, base price, set of images for each category and description. The room type serves as a unique identifier.

**Payment transactions** are tracked by the system, including the payment method, amount, date, and a reference number to identify the transaction. Each transaction has a unique identifier.

Accepted **payment methods** are listed in the system, providing details about the method name (e.g., Credit Card, Cash, Mobile Money).

Information about **employees** is stored, including their ssn, first name, last name, hourly rate, hire date, phone number, email, and address that is broken down to like guest address.

**Job roles** within the hotel are defined in the system, specifying the different positions (e.g., Manager, Housekeeper) and a description of each role's responsibilities.

Guest **feedback** is collected in the system, recording ratings, comments and date of submission and has a unique feedback id.

The system handles **maintenance requests** for rooms, storing details about the issue, which employee is assigned to resolve it, and the request's current status (e.g., Pending, In Progress, Completed), request date and a unique identifier.

**Amenities** available in the hotel are listed, providing descriptions of each amenity (e.g., Wi-Fi, Air Conditioning) and name of the amenity that guests can access in their rooms or throughout the hotel. Each amenity has a unique id.

**Suppliers** providing inventory for the hotel are recorded in the system, including the supplier's name, contact person full name, phone number, email and address. Each supplier is identified uniquely in the system.

The system manages **inventory items**, tracking the item name, description, quantity in stock, and the supplier that provides each item. Each inventory item is identified uniquely by its id.

### Specific Constraints:

#### 1. Uniqueness Constraints:

- **Guest Email:** Each guest must have a unique email address, ensuring that no duplicate emails exist in the system.
- **Room Number:** Each room in a hotel must have a unique room number within that hotel.

#### 2. Date Constraints:

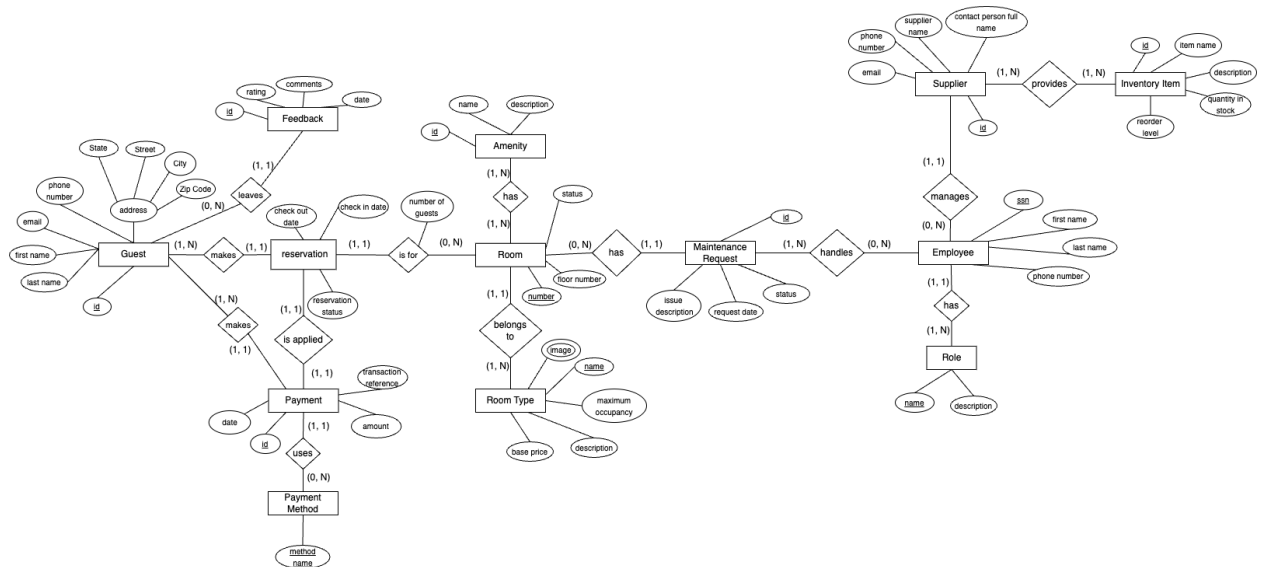
- **Check-in and Check-out Dates:** The check-out date for a reservation must always be after the check-in date.
- **Reservation Date Validation:** Reservations cannot be made in the past. The system must only allow future dates for new reservations.

### Functional Requirements:

1. **Guest Listing with Filters:** Users should be able to view all Guest in a table, with filters for first name, last name, email, phone, street, city and state
2. **Make a New Reservation:** The system should allow users to create new reservations by selecting a guest, room, check-in, and check-out dates.
3. **Reservation Listing with Filters:** Users should be able to view all reservations with the ability to filter by guest name, room number, and reservation status.
4. **Dynamic Pricing for Reservations:** The system should calculate the total reservation cost dynamically based on room type, stay duration, and occupancy rates.
5. **Payment Integration:** Users should be able to process payments for reservations by selecting a payment method and generating a transaction reference.
6. **Maintenance Request Creation:** Users should be able to create new maintenance requests, specifying the room number, issue description, and assigning an employee to handle the task.
7. **Inventory Listing with Filters:** Users should be able to view all inventory items with filters for supplier name and item name.
8. **Supplier Listing with Filters:** Users should be able to view all suppliers and filter them by name, contact person, and managing employee.
9. **Edit Inventory Items:** Users should be able to edit inventory details like quantity in stock and reorder levels through a popup dialog.

10. **Low Stock Alerts:** The system should display items that are below their reorder level in a dedicated low stock alert page.

## ER Model



## Relational Model

Guest (id PK, first\_name, last\_name, email, phone\_number, street, city, state, zip\_code)

Reservation (guest\_id FK, check\_in\_date, check\_out\_date, reservation\_status, room\_inumberFK, payment\_id)

Feedback (id PK, guest\_id FK, rating, comments, date)

Payment (id PK, guest\_id FK, payment\_method\_id FK, amount, date, transaction\_reference)  
PaymentMethod (method\_name PK)  
Room (number PK, room\_name FK, floor\_number, status)  
RoomType (name PK, description, maximum\_occupancy, base\_price, image)  
Amenity (id PK, name, description)  
RoomAmenity (room\_number FK, amenity\_id FK)  
MaintenanceRequest (id PK, room\_number FK, issue\_description, request\_date, status)  
MaintenanceRequestEmployee (employee\_ssn FK, maintenance\_request\_id FK)  
Supplier (id PK, supplier\_name, contact\_person\_full\_name, phone\_number, email, employee\_id FK)  
InventoryItem (id PK, supplier\_id FK, item\_name, description, quantity\_in\_stock, reorder\_level)  
SupplierProvidedItem (supplier\_id FK, Inventory\_id FK)  
Employee (ssn PK, first\_name, last\_name, phone\_number, role\_name FK)  
Role (name PK, description)

## 1. Guest Relation

### **Relation:**

Guest (**id** PK, first\_name, last\_name, email, phone\_number, street, city, state, zip\_code)

### Transformation:

- The Guest entity was straightforward to map into a relation.
- Attributes of Guest include personal details, such as first\_name, last\_name, and address information.

### Cardinalities:

- No specific relationship cardinality affected this table design.

### 3NF Validation:

1. All attributes are functionally dependent on the primary key id.
2. There are no transitive dependencies since address fields (street, city, state, zip\_code) are atomic and directly tied to id.
3. No partial dependencies exist because id is the sole primary key.

## 2. Reservation Relation

### **Relation:**

Reservation (guest\_id FK, check\_in\_date, check\_out\_date, reservation\_status, room\_number FK, payment\_id FK)

Transformation:

- The Reservation entity links Guest, Room, and Payment through foreign keys (guest\_id, room\_number, and payment\_id).
- Dates (check\_in\_date, check\_out\_date) and reservation status are included as attributes.

Cardinalities:

- One-to-many between Guest and Reservation is enforced via guest\_id FK.
- One-to-one/many-to-one between Room and Reservation depends on availability but is enforced via room\_number FK.

3NF Validation:

1. All attributes depend on the composite key (guest\_id, room\_number), which is unique for each reservation.
2. Attributes like reservation\_status are directly tied to the reservation and have no transitive dependencies.
3. The design avoids partial dependencies by not splitting the composite key unnecessarily.

## 3. Feedback Relation

### **Relation:**

Feedback (id PK, guest\_id FK, rating, comments, date)

Transformation:

- Feedback relates directly to Guest (guest\_id FK) with additional attributes such as rating and comments.

Cardinalities:

- One-to-many between Guest and Feedback allows multiple feedback entries per guest.

3NF Validation:

1. The primary key id determines all other attributes (rating, comments, date).
  2. No transitive dependencies exist since all feedback attributes directly depend on id.
  3. Partial dependencies are absent due to the single primary key.
- 

#### 4. Payment Relation

**Relation:**

Payment (**id** PK, guest\_id FK, payment\_method\_name FK, amount, date, transaction\_reference)

Transformation:

- Links Guest and PaymentMethod using foreign keys.

Cardinalities:

- One-to-many between Guest and Payment ensures multiple payments per guest.
- Many-to-one between Payment and PaymentMethod.

3NF Validation:

1. The primary key id determines all attributes (amount, date, etc.).
2. No transitive dependencies exist since attributes depend directly on id.
3. No partial dependencies exist due to the single primary key.

#### 5. Room Relation

**Relation:**

Room (**number** PK, room\_name, floor\_number, status)

Transformation:

- Room attributes include room identification, status, and location.

Cardinalities:

- One-to-many between RoomType and Room ensures a type can include many rooms.

3NF Validation:

1. Primary key number determines attributes like room\_name, floor\_number, and status.
2. No transitive dependencies exist because each room directly relates to its primary key.
3. No partial dependencies due to the atomic nature of the attributes.

## 6. RoomType Relation

**Relation:**

RoomType (**name** PK, description, maximum\_occupancy, base\_price, image)

Transformation:

- Attributes describe different room types and pricing.

Cardinalities:

- Many-to-one between Room and RoomType.

3NF Validation:

1. Primary key name determines all other attributes.
2. No transitive dependencies exist since each room type is atomic.
3. No partial dependencies exist due to the single primary key.

## 7. Amenity Relation

**Relation:**

Amenity (**id** PK, name, description)

Transformation:

- Amenity includes attributes for identification and description.



Cardinalities:

- Many-to-many between Room and Amenity, resolved through RoomAmenity.

3NF Validation:

1. Primary key id determines all attributes.
2. Attributes like name and description are atomic.
3. No partial or transitive dependencies.

#### 8. RoomAmenity Relation

**Relation:**

RoomAmenity (room\_number FK, amenity\_id FK)

Transformation:

- Resolves the many-to-many relationship between Room and Amenity.

3NF Validation:

1. Composite key (room\_number, amenity\_id) uniquely identifies each pair.
2. Attributes depend only on the composite key.
3. No transitive or partial dependencies exist.

#### 9. MaintenanceRequest Relation

**Relation:**

MaintenanceRequest (**id** PK, room\_number FK, issue\_description, request\_date, status)

Transformation:

- MaintenanceRequest links directly to Room (room\_number FK) and includes attributes for the issue and its status.

3NF Validation:

1. Primary key id determines all attributes.
2. No transitive dependencies exist.
3. Attributes are atomic, and no partial dependencies exist.

#### 10. MaintenanceRequestEmployee Relation

**Relation:**

MaintenanceRequestEmployee (employee\_ssn FK, maintenance\_request\_id FK)

Transformation:

- Resolves the many-to-many relationship between Employee and MaintenanceRequest.

3NF Validation:

1. Composite key (employee\_ssn, maintenance\_request\_id) uniquely identifies each pair.
2. No transitive dependencies exist.
3. No partial dependencies exist.

## 11. Supplier Relation

**Relation:**

Supplier (**id** PK, supplier\_name, contact\_person\_full\_name, phone\_number, email, employee\_id FK)

Transformation:

- Attributes include supplier details, linked to an Employee.

Cardinalities:

- Many-to-one between Supplier and Employee.

3NF Validation:

1. Primary key id determines all attributes.
2. Attributes are atomic, with no transitive or partial dependencies.

## 12. InventoryItem Relation

**Relation:**

InventoryItem (**id** PK, supplier\_id FK, item\_name, description, quantity\_in\_stock, reorder\_level)

Transformation:

- Links Supplier to inventory items.

3NF Validation:

1. Primary key id determines all attributes.
2. No transitive dependencies exist.
3. Attributes like item\_name and description are atomic.

### 13. SupplierProvidedItem Relation

**Relation:**

SupplierProvidedItem (supplier\_id FK, inventory\_id FK)

Transformation:

- Resolves the many-to-many relationship between Supplier and InventoryItem.

3NF Validation:

1. Composite key (supplier\_id, inventory\_id) uniquely identifies each pair.
2. Attributes depend only on the composite key.
3. No transitive or partial dependencies.

### 14. Employee Relation

**Relation:**

Employee (ssn PK, first\_name, last\_name, phone\_number, role\_name FK)

Transformation:

- Links Role through role\_name FK.

Cardinalities:

- Many-to-one between Employee and Role.

3NF Validation:

1. Primary key ssn determines all attributes.
2. Attributes like first\_name are atomic.
3. No transitive or partial dependencies.

### 15. Role Relation

**Relation:**

Role (name PK, description)

Transformation:

- Attributes describe roles in the system.

3NF Validation:

1. Primary key name determines all attributes.
2. Attributes are atomic, with no transitive or partial dependencies.

## Data Dictionary

Data Dictionary for Table: **Amenity**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER (Primary Key)	Primary Key	Auto-increment, Non-null
name	VARCHAR(50)	None	Unique, Non-null
description	TEXT	None	Optional

Data Dictionary for Table: **Employee**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
ssn	CHAR(9)	Primary Key	Unique, Non-null
first_name	VARCHAR(50)	None	Non-null
last_name	VARCHAR(50)	None	Non-null
phone_number	VARCHAR(20)	None	Optional
role_name	VARCHAR(50)	Foreign Key	References Role(name), Non-null

Data Dictionary for Table: **Guest**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
first_name	VARCHAR(50)	None	Non-null
last_name	VARCHAR(50)	None	Non-null
email	VARCHAR(100)	None	Unique, Non-null
phone_number	VARCHAR(20)	None	Non-null
street	VARCHAR(100)	None	Optional
city	VARCHAR(50)	None	Optional
state	VARCHAR(50)	None	Optional
zip_code	VARCHAR(10)	None	Optional

**Data Dictionary for Table: InventoryItem**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
supplier_id	INTEGER	Foreign Key	References Supplier(id), Non-null
item_name	VARCHAR(100)	None	Non-null
description	TEXT	None	Optional
quantity_in_stock	INTEGER	None	Non-null
reorder_level	INTEGER	None	Non-null

**Data Dictionary for Table: LowStockAlert**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
item_id	INTEGER	Primary Key, Foreign Key	References InventoryItem(id), Non-null
item_name	VARCHAR(255)	None	Non-null
current_stock	INTEGER	None	Non-null, Must be $\geq 0$
reorder_level	INTEGER	None	Non-null, Must be $\geq 0$
alert_date	TIMESTAMP	None	Non-null, Auto-updated on row modification

**Data Dictionary for Table: MaintenanceRequest**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
room_number	INTEGER	Foreign Key	References Room(number), Non-null
issue_description	TEXT	None	Non-null
request_date	DATE	None	Non-null
status	VARCHAR(20)	None	Non-null, Values: 'Pending', 'In Progress', 'Completed'

**Data Dictionary for Table: MaintenanceRequestEmployee**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
----------------	----------------------	---------------------------	------------------------

employee_ssn	CHAR(9)	Primary Key, Foreign Key	References Employee(ssn), Non-null
maintenance_request_id	INTEGER	Primary Key, Foreign Key	References MaintenanceRequest(id), Non-null

**Data Dictionary for Table: Payment**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
guest_id	INTEGER	Foreign Key	References Guest(id), Non-null
payment_method_name	VARCHAR(50)	Foreign Key	References PaymentMethod(method_name), Non-null
amount	DECIMAL(10,2)	None	Non-null, Positive value
date	DATE	None	Non-null
transaction_reference	VARCHAR(100)	None	Unique, Non-null

**Data Dictionary for Table: PaymentMethod**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
method_name	VARCHAR(50)	Primary Key	Unique, Non-null

**Data Dictionary for Table: Reservation**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
reservation_id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
guest_id	INTEGER	Foreign Key	References Guest(id), Non-null
check_in_date	DATE	None	Non-null
check_out_date	DATE	None	Non-null, Greater than check_in_date
reservation_status	VARCHAR(20)	None	Non-null, Enum('Pending', 'Confirmed', 'Checked-Out')
amount_due	DECIMAL(10,2)	None	Non-null, Positive value
room_number	INTEGER	Foreign Key	References Room(number), Non-null
payment_id	INTEGER	Foreign Key	References Payment(id), Nullable

**Data Dictionary for Table: Role**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
name	VARCHAR(50)	Primary Key	Unique, Non-null
description	TEXT	None	Nullable

**Data Dictionary for Table: Room**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
number	INTEGER	Primary Key	Unique, Non-null
room_name	VARCHAR(50)	None	Non-null
floor_number	INTEGER	None	Non-null, Positive value
status	VARCHAR(20)	None	Non-null, Enum('Available', 'Occupied', 'Under Maintenance')
room_type_name	VARCHAR(50)	Foreign Key	References RoomType(name), Non-null

**Data Dictionary for Table: RoomAmenity**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
room_number	INTEGER	Primary Key, Foreign Key	References Room(number), Non-null
amenity_id	INTEGER	Primary Key, Foreign Key	References Amenity(id), Non-null

**Data Dictionary for Table: Supplier**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
id	INTEGER	Primary Key	Unique, Non-null, Auto-increment
supplier_name	VARCHAR(100)	None	Non-null, Unique
contact_person_full_name	VARCHAR(100)	None	Non-null
phone_number	VARCHAR(20)	None	Non-null, Valid phone number format
email	VARCHAR(100)	None	Non-null, Valid email format
employee_id	CHAR(9)	Foreign Key	References Employee(ssn), Nullable

**Data Dictionary for Table: SupplierProvidedItem**

Attribute Name	Data Type and Domain	Primary Key / Foreign Key	Additional Constraints
supplier_id	INTEGER	Primary Key, Foreign Key	References Supplier(id), Non-null
inventory_id	INTEGER	Primary Key, Foreign Key	References InventoryItem(id), Non-null

### Workload


Natnael (Did majority of the work for this project as peer member had to deal with other personal matters)

- developed the website
- worked on the final report
- worked on the ER design

Ninad

- worked on ER design and relational model

### Access Link

 database systems final video.mp4

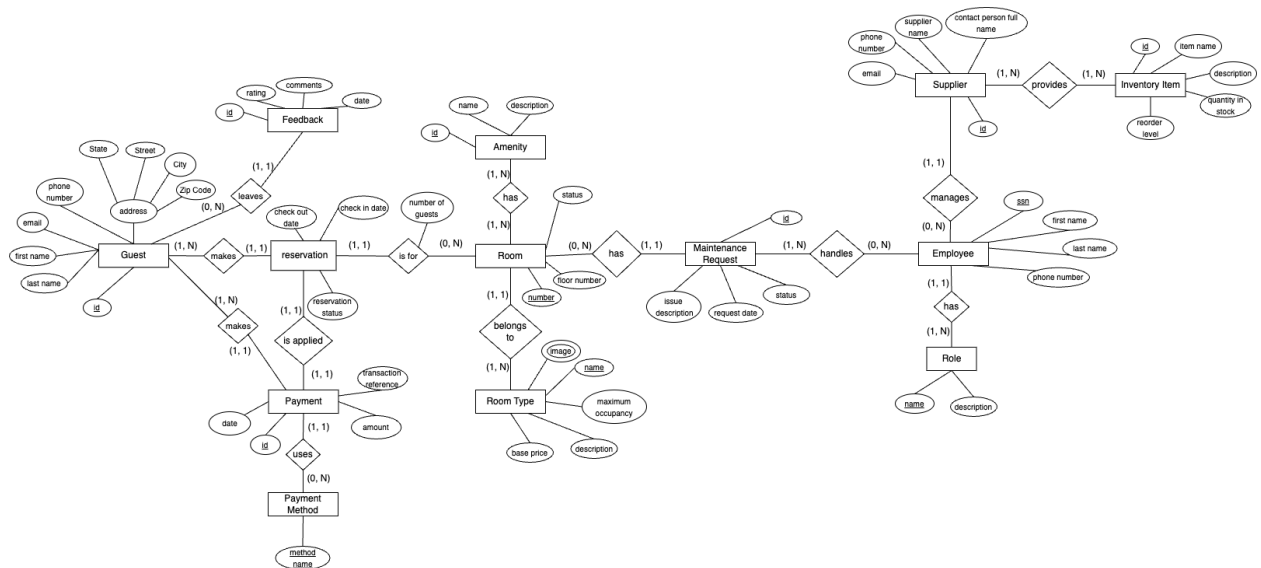


## Appendix

### ER model:

<https://drive.google.com/file/d/1WKfsFP8Z9FGSsTqFQ3wE1w6lbBv3L73n/view?usp=sharing>

g



### Relational Model:

Guest (id PK, first\_name, last\_name, email, phone\_number, street, city, state, zip\_code)

Reservation (guest\_id FK, check\_in\_date, check\_out\_date, reservation\_status, room\_inumberFK, payment\_id)

Feedback (id PK, guest\_id FK, rating, comments, date)

Payment (id PK, guest\_id FK, payment\_method\_id FK, amount, date, transaction\_reference)

PaymentMethod (method\_name PK)

Room (number PK, room\_name FK, floor\_number, status)

RoomType (name PK, description, maximum\_occupancy, base\_price, image)

Amenity (id PK, name, description)

RoomAmenity (room\_number FK, amenity\_id FK)

MaintenanceRequest (id PK, room\_number FK, issue\_description, request\_date, status)

MaintenanceRequestEmployee (employee\_ssn FK, maintenance\_request\_id FK)

Supplier (id PK, supplier\_name, contact\_person\_full\_name, phone\_number, email, employee\_id FK)

InventoryItem (id PK, supplier\_id FK, item\_name, description, quantity\_in\_stock, reorder\_level)

SupplierProvidedItem (supplier\_id FK, Inventory\_id FK)

Employee (ssn PK, first\_name, last\_name, phone\_number, role\_name FK)

Role (name PK, description)