Here’s a **PostgreSQL schema** with **FOREIGN KEY relationships** for your **Hotel Management System**.

**SQL Schema with Foreign Keys**

sql

CopyEdit

-- Table: Guests

CREATE TABLE guests (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20) UNIQUE NOT NULL,

loyalty\_tier VARCHAR(50) CHECK (loyalty\_tier IN ('Bronze', 'Silver', 'Gold', 'Platinum'))

);

-- Table: Rooms

CREATE TABLE rooms (

id SERIAL PRIMARY KEY,

room\_number VARCHAR(20) UNIQUE NOT NULL,

type VARCHAR(100) NOT NULL,

floor INT NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Available', 'Occupied', 'Reserved', 'Maintenance', 'Cleaning')),

price DECIMAL(10,2) NOT NULL

);

-- Table: Bookings

CREATE TABLE bookings (

id SERIAL PRIMARY KEY,

guest\_id INT NOT NULL REFERENCES guests(id) ON DELETE CASCADE,

room\_id INT NOT NULL REFERENCES rooms(id) ON DELETE CASCADE,

check\_in DATE NOT NULL,

check\_out DATE NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Confirmed', 'Checked In', 'Checked Out', 'Cancelled', 'Pending')),

amount DECIMAL(10,2) NOT NULL,

payment\_status VARCHAR(50) NOT NULL CHECK (payment\_status IN ('Paid', 'Pending'))

);

-- Table: Billing

CREATE TABLE billing (

id SERIAL PRIMARY KEY,

guest\_id INT NOT NULL REFERENCES guests(id) ON DELETE CASCADE,

booking\_id INT NOT NULL REFERENCES bookings(id) ON DELETE CASCADE,

date DATE NOT NULL DEFAULT CURRENT\_DATE,

due\_date DATE NOT NULL,

amount DECIMAL(10,2) NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Paid', 'Pending', 'Overdue', 'Cancelled', 'Refunded')),

payment\_method VARCHAR(50) NOT NULL CHECK (payment\_method IN ('Credit Card', 'Debit Card', 'PayPal', 'Cash', 'Bank Transfer'))

);

-- Table: Services

CREATE TABLE services (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

category VARCHAR(100) NOT NULL,

price DECIMAL(10,2) NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Available', 'Booked'))

);

-- Table: Booking\_Services (Many-to-Many between Bookings & Services)

CREATE TABLE booking\_services (

booking\_id INT NOT NULL REFERENCES bookings(id) ON DELETE CASCADE,

service\_id INT NOT NULL REFERENCES services(id) ON DELETE CASCADE,

PRIMARY KEY (booking\_id, service\_id)

);

-- Table: Staff

CREATE TABLE staff (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

role VARCHAR(100) NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Active', 'On Leave', 'Inactive'))

);

-- Table: Inventory

CREATE TABLE inventory (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

category VARCHAR(100) NOT NULL,

quantity INT NOT NULL CHECK (quantity >= 0),

status VARCHAR(50) NOT NULL CHECK (status IN ('In Stock', 'Low Stock', 'Out of Stock')),

managed\_by INT REFERENCES staff(id) ON DELETE SET NULL

);

-- Table: Security

CREATE TABLE security (

id SERIAL PRIMARY KEY,

name VARCHAR(255) UNIQUE NOT NULL,

status VARCHAR(50) NOT NULL CHECK (status IN ('Enabled', 'Disabled', 'Limited'))

);

-- Table: Staff\_Security (Many-to-Many between Staff & Security)

CREATE TABLE staff\_security (

staff\_id INT NOT NULL REFERENCES staff(id) ON DELETE CASCADE,

security\_id INT NOT NULL REFERENCES security(id) ON DELETE CASCADE,

PRIMARY KEY (staff\_id, security\_id)

);

-- Table: Settings

CREATE TABLE settings (

id SERIAL PRIMARY KEY,

name VARCHAR(255) UNIQUE NOT NULL,

value VARCHAR(255) NOT NULL

);

-- Table: Reports

CREATE TABLE reports (

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

date DATE NOT NULL DEFAULT CURRENT\_DATE,

status VARCHAR(50) NOT NULL CHECK (status IN ('Completed', 'Pending'))

);

**Relationships**

* **Guests → Bookings (1:N)**: A guest can have multiple bookings.
* **Rooms → Bookings (1:N)**: A room can have multiple bookings over time.
* **Bookings → Billing (1:1)**: Each booking has a single billing entry.
* **Bookings → Services (M:N)**: A booking can request multiple services, and services can belong to multiple bookings.
* **Staff → Inventory (1:N)**: A staff member manages multiple inventory items.
* **Staff → Security (M:N)**: A staff member can have multiple security permissions.
* **Global settings** apply system-wide (no foreign keys needed).

**Indexes for Performance**

sql

CopyEdit

CREATE INDEX idx\_guest\_email ON guests(email);

CREATE INDEX idx\_room\_status ON rooms(status);

CREATE INDEX idx\_booking\_status ON bookings(status);

CREATE INDEX idx\_billing\_status ON billing(status);

**How to Use**

1. **Connect to PostgreSQL**

sh

CopyEdit

psql -U your\_user -d hotel\_db

1. **Run the Schema**  
   Copy & paste this SQL into your PostgreSQL console.

Now your database is **ready** with all relationships! 🚀 Let me know if you need modifications!