HOTEL BOOKING SYSTEM

Database System Project

Alexandria University
Faculty of Computers and Data Science
Data Science Department

ROOM207🤫🔥

PRESENTED BY

Rowan Hussein
Salsabel Osama
Shahd Mohamed
Shahd Ahmed

TABLE OF CONTENTS

INTRODUCTION

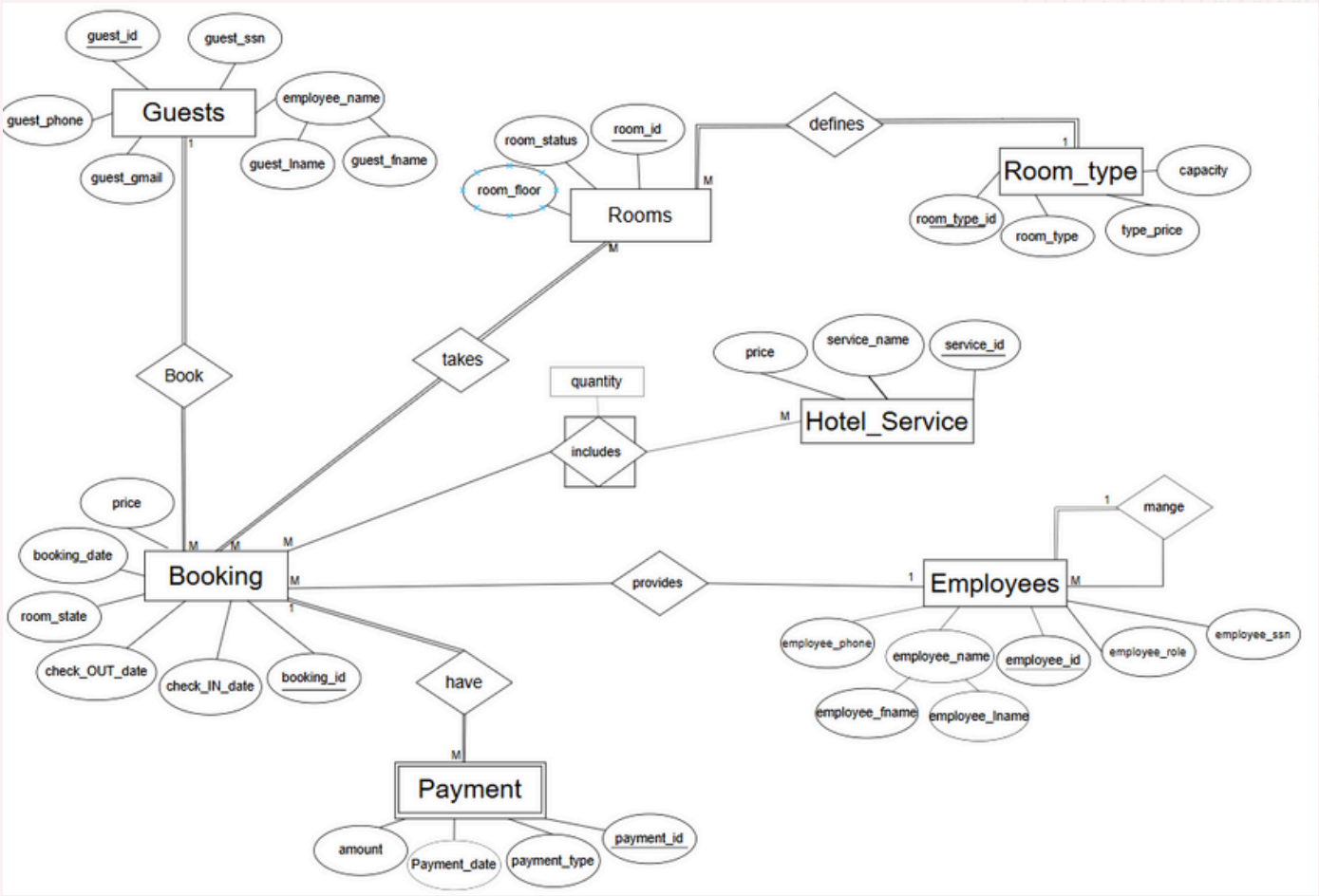
ROOM207🤫🔥

The ROOM207 Database is designed to simulate the operations of a hotel system inspired by the series ROOM207. This database aims to efficiently manage the core aspects of hotel management, including guest information, room availability, employee details, bookings, payments, room types, and additional services.
The database is structured around several key entities:

1. Guests – stores personal and contact information for each hotel guest.
2. Rooms – tracks room details such as floor and current status (available, occupied, or under maintenance).
3. Employees – maintains staff records including their roles and contact information.
4. Booking – records guest reservations, including check-in/check-out dates, pricing, and booking status.
5. Room Types – defines different room categories with their price and capacity.
6. Payment – tracks financial transactions related to bookings.
7. Services – manages additional hotel services offered to guests.

This database provides a comprehensive framework for managing hotel operations, ensuring data integrity, and facilitating queries for reporting and analysis. The design allows for clear relationships between guests, rooms, bookings, and payments, enabling smooth management and operational efficiency.

ENTITY RELATIONSHIPS DIAGRAM



ER Diagram Description – ROOM207 Database

The ROOM207 database is designed to manage hotel operations efficiently. It consists of seven main entities, each representing a core aspect of the hotel management system:

1. Guests
   - Description: Represents all customers who book rooms in the hotel.
   - Attributes: guest_id (primary key), guest_fname, guest_lname, guest_ssn (unique identifier), guest_gmail, guest_phone.
   - Purpose: To store and manage guest information for bookings and services.
2. Rooms
   - Description: Represents all rooms in the hotel.
   - Attributes: room_id (primary key), room_floor, room_status (available, occupied, etc.).
   - Purpose: To track room availability and assignment for bookings.
3. Employees
   - Description: Represents all hotel staff members.
   - Attributes: employee_id (primary key), employee_fname, employee_lname, employee_role (e.g., receptionist, manager), employee_phone, employee_ssn.
   - Purpose: To manage employee data and assign roles in hotel operations.

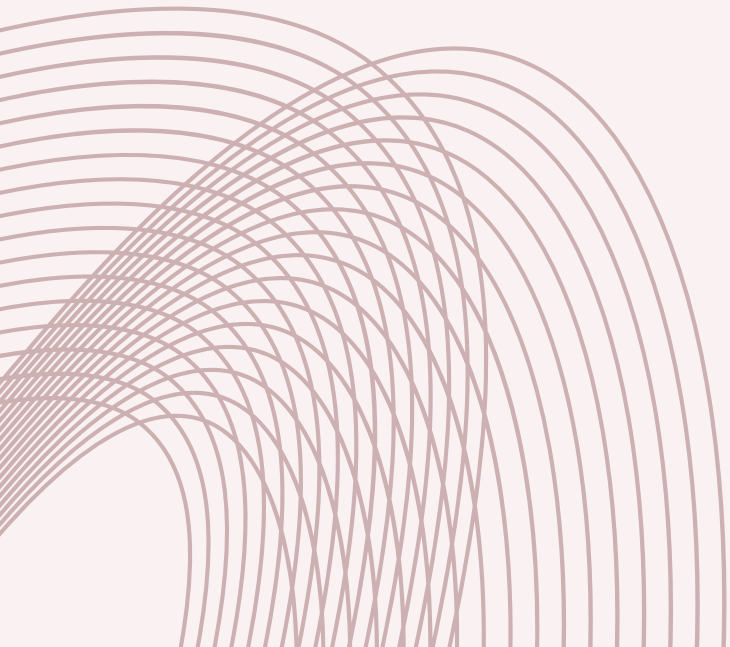
4.Booking
- Description: Represents room reservations made by guests.
- Attributes: booking_id (primary key), check_IN_date, check_OUT_date, Price, booking_date, Status (e.g., confirmed, cancelled).
- Purpose: To record booking details, associate guests with rooms, and manage reservation status.

5.Room_type
- Description: Represents types of rooms available in the hotel.
- Attributes: room_type_id (primary key), type (e.g., single, double, suite), type_price, capacity.
- Purpose: To categorize rooms and determine pricing and capacity.

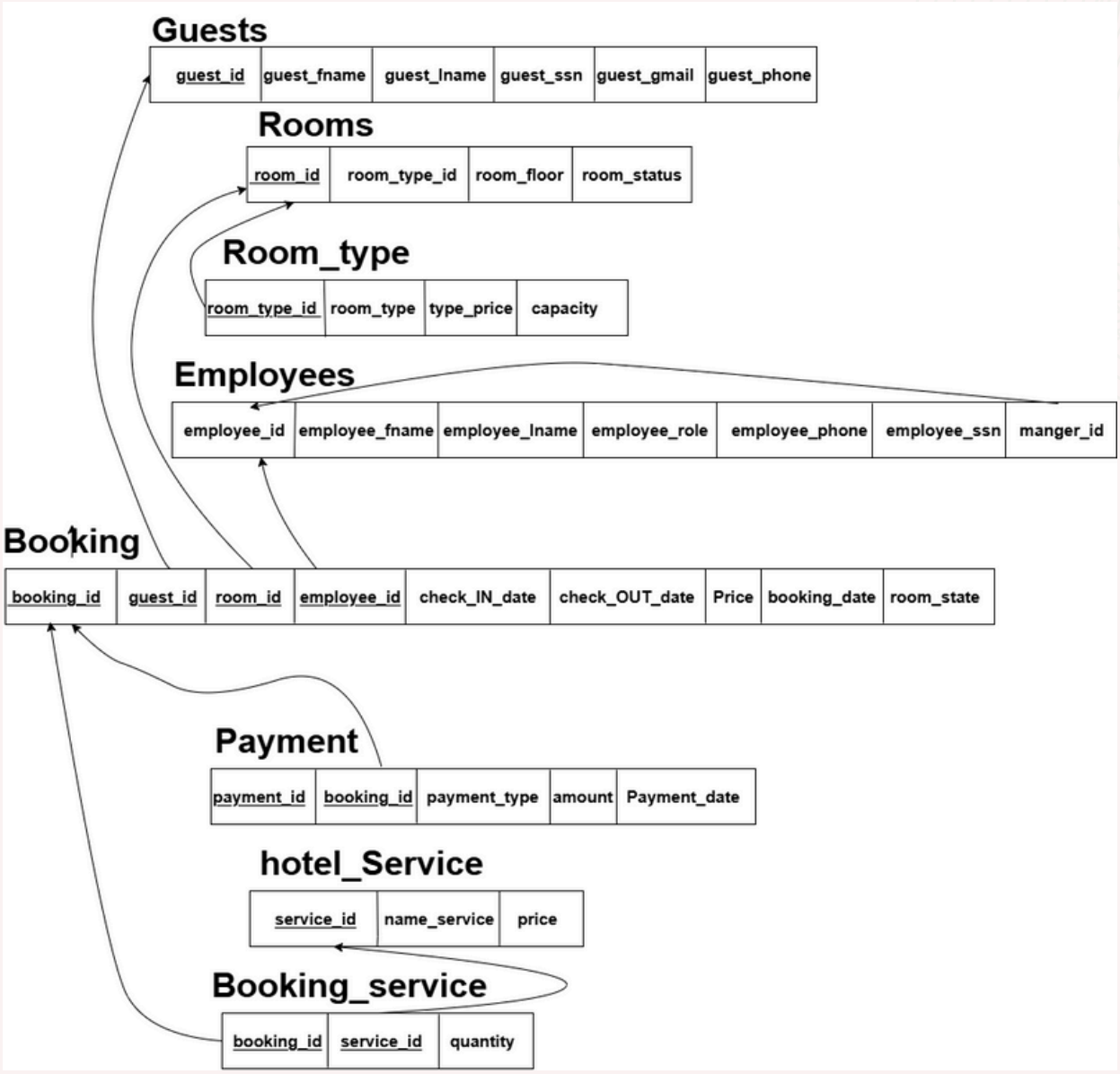
6.Payment
- Description: Represents payments made by guests for bookings or services.
- Attributes: payment_id (primary key), payment_type (cash, card, online), amount, Payment_date.
- Purpose: To track and manage all financial transactions within the hotel.

7.Services
- Description: Represents additional services offered by the hotel.
- Attributes: service_id (primary key), service_name (e.g., spa, room service), price.
- Purpose: To provide optional services to guests and manage associated charges.

Relationships Overview:
- Each Guest can have multiple Bookings.
- Each Booking is associated with one Room and may involve multiple Services.
- Payments are linked to Bookings to manage the financial aspect of reservations.
- Room_type categorizes Rooms to differentiate pricing and capacity.
- Employees manage hotel operations and may assist with Bookings or Services.



MAPPING



The ROOM207 ER diagram has been converted into relational tables as follows:

1. Guests Table
   - Mapping: Entity Guests → Table Guests.
   - Primary Key: guest_id
   - Foreign Keys: None
   - Note: Stores guest details, ready to link to Booking through guest_id.
2. Rooms Table
   - Mapping: Entity Rooms → Table Rooms.
   - Primary Key: room_id
   - Foreign Keys: room_type_id → Room_type(room_type_id)
   - Note: Each room is assigned a room type; room_status indicates availability.

3.Employees Table

- Mapping: Entity Employees → Table Employees.
- Primary Key: employee_id
- Foreign Keys: manager_id → Employees(employee_id) (self-referencing)
- Note: Employee hierarchy is maintained through manager_id.

4.Booking Table

- Mapping: Entity Booking → Table Booking.
- Primary Key: booking_id
- Foreign Keys:
- guest_id → Guests(guest_id)
- room_id → Rooms(room_id)
- employee_id → Employees(employee_id)
- Note: Associates guests, rooms, and employees. Contains booking details like dates, price, and room state.

5.Room_type Table

- Mapping: Entity Room_type → Table Room_type.
- Primary Key: room_type_id
- Foreign Keys: None
- Note: Defines types of rooms, their price, and capacity.

6.Payment Table

- Mapping: Entity Payment → Table Payment.
- Primary Key: Composite Key (payment_id, booking_id)
- Foreign Keys: booking_id → Booking(booking_id)
- Note: Each payment is linked to a specific booking; supports multiple payment methods per booking.

7.hotel_Service Table

- Mapping: Entity hotel_Service → Table hotel_Service.
- Primary Key: service_id
- Foreign Keys: None
- Note: Stores hotel services available for booking.

8.Booking_service Table

- Mapping: Many-to-Many Relationship Booking ↔ hotel_Service → Table Booking_service.
- Primary Key: Composite Key (booking_id, service_id)
- Foreign Keys:
- booking_id → Booking(booking_id)
- service_id → hotel_Service(service_id)

Note: Captures which services are used for each booking and their quantities.

Summary of Mapping Rules:

Each Entity becomes a Table.

Attributes of entities become columns in the table.

Primary Keys (PK) uniquely identify records in each table.

Foreign Keys (FK) enforce relationships between tables.

Many-to-Many relationships (like Booking ↔ Services) are implemented via junction tables with composite keys.

DATA DICTIONARY

Entities

Entity Name	Description
Guests	Stores information about hotel guests, including personal details and contact info.
Room_type	Defines different types of rooms with price and capacity.
Rooms	Contains hotel room details including type, floor, and availability.
Employees	Stores employee data including manager, role, and contact info.
Booking	Records each booking including guest, room, employee, dates, price, and room state.
hotel_Service	Lists hotel services and their prices.
Booking_service	Tracks services used for each booking and their quantities.
Payment	Stores payment information for each booking including type, amount, and date.

Relationships

Relationship	Entities	Type	Keys	Description
Guest_Booking	Guests - Booking	One-to-Many	Guests.guest_id -> Booking.guest_id	Each guest can have multiple bookings.
Room_Booking	Rooms - Booking	One-to-Many	Rooms.room_id -> Booking.room_id	Each room can be booked multiple times.
Employee_Booking	Employees - Booking	One-to-Many	Guests.guest_id -> Booking.guest_id	Each employee can handle multiple bookings.
Booking_Service	Booking - hotel_Service	Many-to-Many	Rooms.room_id -> Booking.room_id	Each booking can include multiple services.
Booking_Payment	Booking - Payment	One-to-One	Guests.guest_id -> Booking.guest_id	Each booking has one payment record.
Employee_Manager	Employees - Employees	One-to-Many	Guests.guest_id -> Booking.guest_id	An employee can manage multiple employees.

Guests

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
guest_id	INT	–	Unique guest ID	PK, IDENTITY(1,1)	No
guest_fname	VARCHAR	50	Guest first name	NN	No
guest_lname	VARCHAR	50	Guest last name	NN	No
guest_ssn	VARCHAR	14	Guest national ID	NN, UQ, CHK(LEN = 14 AND digits only)	No
guest_email	VARCHAR	100	Guest email address	–	Yes
guest_phone	VARCHAR	11	Guest phone number	CHK(digits only)	Yes

Employees

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
employee_id	INT	—	Unique employee ID	PK, IDENTITY(1,1)	No
manager_id	INT	—	ID of the manager (self-reference)	FK: Employees(employee_id)	Yes
employee_lname	VARCHAR	50	Employee last name	NN	No
employee_fname	VARCHAR	50	Employee first name	NN	No
employee_ssn	VARCHAR	14	Employee national ID	NN, UQ	No
employee_role	VARCHAR	100	Role/job title of the employee	—	Yes
employee_phone	VARCHAR	11	Employee phone number	—	Yes

Room_type

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
room_type_id	INT	—	Unique room type ID	PK	No
type_price	DECIMAL	103	Price for this room type	NN	No
capacity	INT	—	Maximum capacity of the room	NN	No
room_type	VARCHAR	30	Type of the room	CHK(room_type IN ('single','double','suite'))	Yes

Rooms

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
room_id	INT	—	Unique room ID	PK	No
room_type_id	INT	—	ID of room type (FK → Room_type)	FK, NN	No
room_floor	INT	—	Floor number of the room	NN	No
room_status	VARCHAR	30	Availability status of the room	NN, CHK('available','not available')	No

Booking

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
booking_id	INT	—	Unique booking ID	PK, IDENTITY(1,1)	No
guest_id	INT	—	ID of the guest	FK : Guests(guest_id), NN	No
room_id	INT	—	ID of the booked room	FK : Rooms(room_id), NN	No
employee_id	INT	—	Employee who handled the booking	FK : Employees(employee_id), NN	No
check_IN_date	DATE	—	Check-in date of the booking	NN	No
check_OUT_date	DATE	—	Check-out date of the booking	NN	No
Price	DECIMAL	103	Price of the booking	—	Yes
booking_date	DATE	—	Date when the booking was made	NN	No
room_state	VARCHAR	50	State of the room at the time of booking	—	Yes

hotel_Service

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
service_id	INT	—	Unique service ID	PK	No
name_service	VARCHAR	100	Name of the service	NN	No
price	DECIMAL	102	Price of the service	NN	No

Booking_service

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
booking_id	INT	—	ID of the booking	PK (part of composite key), FK : Booking(booking_id), NN	No
service_id	INT	—	ID of the service	PK (part of composite key), FK : hotel_Service(service_id), NN	No
quantity	INT	—	Quantity of the service for this booking	NN	No

Payment

Attribute Name	Data Type	Size	Description	Constraints	Allow Null
payment_id	INT	–	Unique payment ID	PK (part of composite key)	No
booking_id	INT	–	ID of the related booking	PK (part of composite key), FK : Booking(booking_id)	No
payment_type	VARCHAR	50	Type of payment (e.g., cash, card)	NN	No
amount	DECIMAL	103	Payment amount	–	Yes
payment_date	DATE	–	Date of the payment	–	Yes

DATABASE IMPLEMENTATION (DDL & DML)

1-Guests

Stores hotel guest information, including first name, last name, SSN, email, and phone. guest_id is the primary key.

2-Room_type

Defines types of rooms (single, double, suite), their price, and capacity. room_type_id is the primary key.

Rooms

3-Represents hotel rooms with floor number, status (available/not available), and type. room_type_id is a foreign key referencing Room_type.

4-Employees

Contains staff details including name, SSN, role, and phone. Supports manager hierarchy via manager_id referencing the same table.

```
create  table Guests (
    guest_id int primary key IDENTITY(1,1) ,
    guest_fname varchar(50) not null ,
    guest_lname varchar(50) not null ,
    guest_ssn varchar(14) not null unique  CHECK (LEN(guest_ssn) = 14 AND guest_ssn  LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') ,
    guest_email varchar(100) ,
    guest_phone varchar(11) CHECK (guest_phone LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
);

create table Room_type(
    room_type_id int primary key ,
    type_price decimal(10,3) not null,
    capacity int not null ,
    room_type varchar(30),check(room_type IN ('single','double','suite'))
);

create  table Rooms(
    room_id int primary key ,
    room_type_id int not null,
    room_floor int not null,
    room_status varchar(30) check(room_status IN ('available','not available')),
    foreign key (room_type_id) REFERENCES Room_type(room_type_id)
);

create table Employees(
    employee_id int primary key IDENTITY(1,1),
    manager_id int,
    employee_lname varchar(50) not null,
    employee_fname varchar(50)not null,
    employee_ssn varchar(14) not null unique  CHECK (LEN(employee_ssn) = 14 AND employee_ssn  LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    employee_role varchar(100),
    employee_phone varchar(11),

    foreign key (manager_id) REFERENCES Employees(employee_id)
);
```

5.Booking

Records reservations, linking guests, rooms, and employees. Includes check-in/out dates, price, booking date, and room state.

6.hotel_Service

Lists hotel services (e.g., spa, room service) and their prices.

7.Booking_service

Junction table for the many-to-many relationship between Booking and hotel_Service, storing service quantities per booking.

8.Payment

Tracks payments for bookings, including type, amount, and date. Uses a composite primary key (payment_id, booking_id) and links to Booking.

```sql
create table Booking (
    booking_id int primary key IDENTITY(1,1),
    guest_id int not null,
    room_id int not null,
    employee_id int not null,
    check_IN_date date not null,
    check_OUT_date date not null,
    Price decimal(10,3),
    booking_date date not null,
    room_state varchar(50),

    foreign key (guest_id) references Guests(guest_id),
    foreign key (room_id) references Rooms(room_id),
    foreign key (employee_id) references Employees(employee_id),
);

create table hotel_Service (
    service_id int primary key,
    name_service varchar(100) not null,
    price decimal(10,2) not null
);

create table Booking_service(
    booking_id int,
    service_id int,
    quantity int not null,

    primary key(booking_id,service_id),
    foreign key (booking_id) REFERENCES Booking(booking_id),
    foreign key (service_id) REFERENCES hotel_Service(service_id)
)

create table Payment(
    payment_id int ,
    booking_id int ,
    payment_type varchar(50) not null ,
    amount decimal(10,3) ,
    Payment_date date

    primary key(payment_id,booking_id),
    foreign key (booking_id) REFERENCES Booking(booking_id)
```

SYSTEM QUERIES AND TRANSACTIONS

Total Value of Extra Services Used by Each Guest

```sql
-- first query
-- Select the full name of each guest and calculate the total value of services they used
select concat(guest_fname,' ',guest_lname) as full_name , sum(BS.quantity * HS.price) as total_service
from Guests G
inner join booking B
ON G.guest_id = B.guest_id
inner join Booking_service BS
on B.booking_id = BS.booking_id
inner join hotel_service HS
on BS.service_id = HS.service_id
group by guest_fname, guest_lname
order by guest_fname
```

Employee and Their Manager Details

```sql
-- scond query
-- Retrieves each employee's name, their manager's name, and role using a self-join on Employees.
select concat(e.employee_fname,' ',e.employee_lname) as employee_name , concat(m.employee_fname,' ',m.employee_lname) as manager_name, e.employee_role
from Employees e inner join Employees m
on e.manager_id= m.employee_id
```

Guests Who Paid Above Average Amount

```sql
-- Third query
-- Retrieves guests who paid more than the average payment amount, showing guest name and payment.
select G.guest_id, concat(G.guest_fname, ' ', G.guest_lname) as guest_name , P.amount
from Payment P
inner JOIN Booking B on P.booking_id = B.booking_id
inner JOIN Guests G on B.guest_id = G.guest_id
where P.amount > (select AVG(amount) from Payment);
```

Monthly Revenue Report

```sql
-- foruth query
-- -- Calculates total payment amount for each month and year, grouped and ordered by date.
select year(payment_date) as year , month(payment_date) as month, sum(amount) as total_amount
from payment
group by year(payment_date) , month(payment_date)
ORDER BY Year, Month;
```

Count of Rooms per Room Type

```sql
-- fifth query
---- Counts the number of rooms for each room type and orders the results by count.
select RT.room_type , count(room_type) as count_room_type
from Room_type RT
group by room_type
order by count(room_type)
```

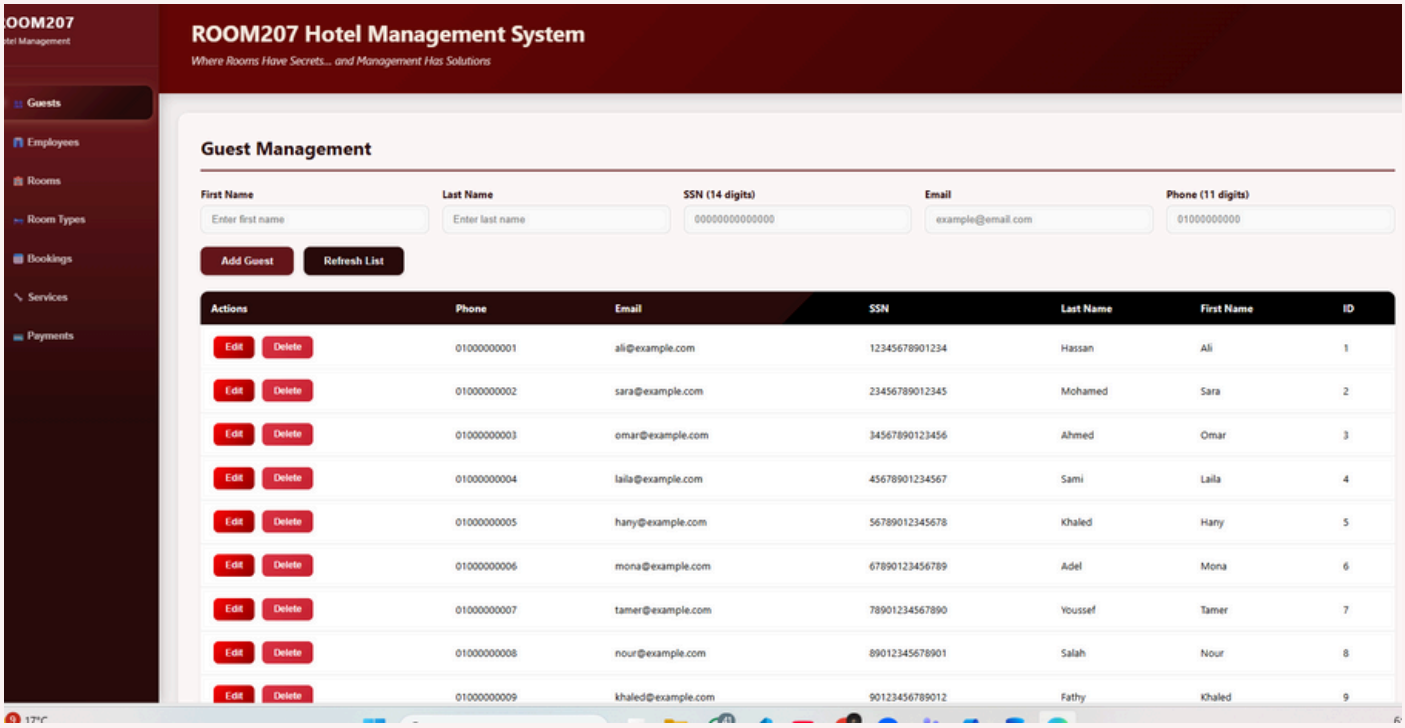
Booking Details: Guest, Employee, and Room Type

```sql
-- sixth query
-- Displays each guest with the employee who handled their booking and the type of room booked.
select concat(G.guest_fname, ' ', G.guest_lname) as guest_name , concat(e.employee_fname,' ',e.employee_lname) as employee_name , room_type
from guests G
inner join Booking B
on G.guest_id = B.guest_id
inner join Employees E
on E.employee_id = B.employee_id
inner join Rooms R
on R.room_id = B.room_id
inner join Room_type RT
on RT.room_type_id = R.room_type_id
```

Most Popular Extra Services

```sql
-- seventh query
-- Counts how many times each hotel service was used, ordered from most to least frequent.
select S.name_service  , count(name_service) as count_of_service
from hotel_Service S
inner join Booking_service BS
on S.service_id = BS.service_id
inner join Booking B
on B.booking_id = BS.booking_id
inner join Rooms R
on R.room_id = B.room_id
group by S.name_service
order by count(name_service) desc
```

GUI OVERVIEW

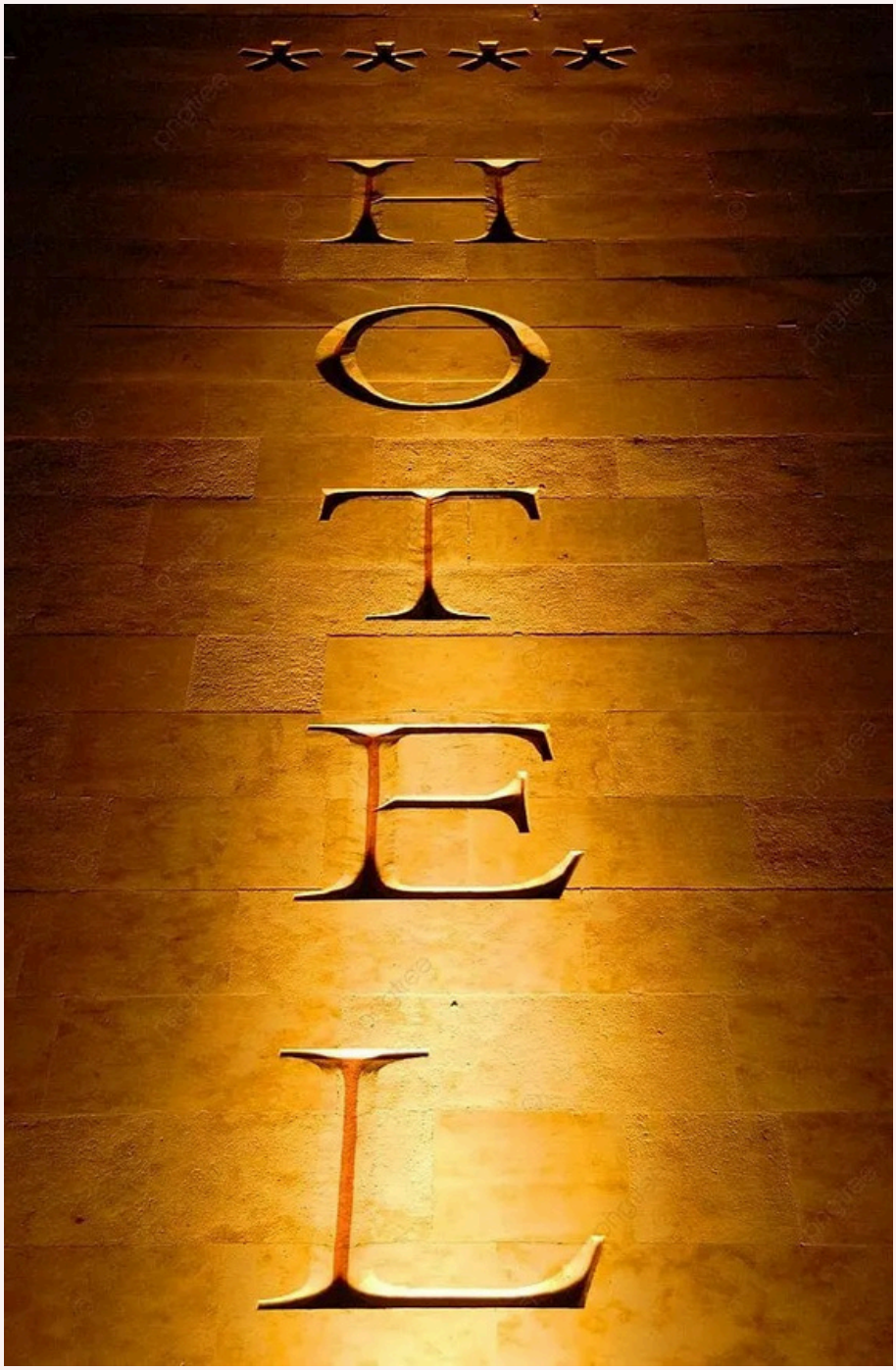


- The Room207 Hotel Management System GUI offers a simple and efficient interface for managing daily hotel operations.
- Its design is inspired by the Room 207 series, using a dark red theme that gives the system a clean and atmospheric look.
- The interface provides clear navigation, responsive forms, and organized data tables—making it easy to add, update, delete, and view records across all modules

The interface is developed using

- HTML for building the page structure
- CSS for styling and layout design
- JavaScript for interactivity and dynamic data updates
- Fetch API to communicate with the backend
- Flask (Python) for handling API requests
- PyODBC + SQL Server as the database layer

CONCLUSION



This project successfully delivered a complete, fully functional, and professional relational database system for a Hotel Booking System that meets and exceeds all requirements of the Database Systems course (Fall 2025/2026).
The system was developed following the full database development life cycle:

Accurate requirements analysis and rich business rules
Clean and detailed Enhanced Entity-Relationship (EER) diagram with correct cardinalities, participation, and hierarchical structure
Precise mapping to a normalized relational schema with all primary and foreign keys
Complete physical implementation using Microsoft SQL Server with strong DDL constraints (IDENTITY, NOT NULL, UNIQUE, CHECK, FOREIGN KEY with RESTRICT)
Comprehensive data integrity enforcement including Egyptian national ID format, room type restrictions, one-payment-per-booking rules
Realistic population with over 200 records across 8 tables
Eight diverse, complex, and practical SQL queries covering all operations and analytics
Prevention of overlapping bookings through robust date logic
Full documentation: Data Dictionary, Transaction Pathway Diagram, screenshots, and execution results

The final product is a robust, scalable, and production-ready database that can serve as the backbone of a real hotel management application. It demonstrates deep understanding of entity-relationship modeling, normalization, constraint enforcement, complex query design, and transaction processing.
This project not only fulfills every single item in the official project guideline checklist but also represents a high-quality, real-world database solution that reflects best practices in database design and implementation.