Cambodia Academy of Digital Technology

Institute of Digital Technology

Faculty of Digital Engineering

Department of Computer Science

DATABASE REPORT OF

Ren Sodalin , Leng Menghan , Soeun Sokvipor , Kruy Seyha

Date: April 7 , 2025

Lecturer : KAK SOKY

Topic : Hotel Reservation System

PHNOM PENH

TABLE OF CONTENTS

I. INTRODUCTION

the Hotel Reservation System, simplifies hotel bookings by allowing guests to check room availability, make reservations, and manage their bookings online. It also helps hotel administrators efficiently track reservations, manage guest details, and improve overall service quality.

1.1 Project Description

In today's fast-paced world, convenience is key when it comes to travel planning. A **Hotel Reservation System** streamlines the booking process, enabling guests to search for available rooms, make reservations, and manage their stays effortlessly. This system acts as a 24/7 digital front desk, ensuring a seamless experience for both travelers and hotel administrators.For hotel managers, the system provides efficient tools to handle room availability, reservations, guest information, and staff coordination**.** Additionally, features such as customer reviews, special requests, and personalized preferences enhance guest satisfaction and improve service quality.By integrating automation and real-time updates, the Hotel Reservation System eliminates the hassle of manual booking processes, reduces errors, and optimizes hotel operations—making it an essential tool for modern hospitality management.

1.2 objectives

The primary objective of this project is to develop a user-friendly, efficient, and secure database system for hotel reservations. The system aims to:

- Facilitate easy room booking for guests.
- Provide real-time availability updates.
- Automate reservation and payment processing.
- Maintain a structured database for efficient hotel management.
- Improve customer experience and operational efficiency.

II. DATABASE ANALYSIS

2.2 User and Project Requirements

2.2.1 User

The system caters to two main user groups:

- **Guests:** They should be able to search for available rooms, make reservations, modify bookings, and make payments.
- **Adminwistrators:** They require access to manage reservations, room status, payments, guest reviews, and staff assignments

2.2.2 Project Requirements

The Hotel Reservation System is designed to efficiently manage hotel bookings, ensuring a seamless experience for both guests and administrators. The system keeps track of guests, rooms, reservations, payments, reviews, special requests, staff operations, and partnerships. Each room is uniquely identified by a Room ID and has attributes such as room type (Single, Double, Suite), bed count, price per night, floor number, status (Available/Booked), and amenities like WiFi, air conditioning, and television.

Guests register in the system by providing a Guest ID, full name, email, phone number, address, and date of birth. When a guest books a room, a Reservation ID is generated, linking the reservation to the guest and the selected room. The reservation includes check-in and check-out dates, number of nights, status (Confirmed/Canceled), and total cost. Payments are recorded under Payment ID, linked to a reservation, and include details such as payment date, amount paid, payment method (Credit Card, Cash, Online), and a transaction ID for verification.

Guests can submit reviews, which include a Review ID, rating (1-5 stars), comments, and submission date, linking them to both the guest and the room. Additionally, guests can request special services, such as early check-in or extra towels, recorded under a Request ID, with details like request type, description, status, and date requested.

The hotel employs multiple staff members, each identified by a Staff ID, full name, email, contact number, hire date, roles (Administrator or Employee), and shift schedules. Staff members handle different operations: Administrators manage system functions, while employees handle reservations and housekeeping.

The system also manages partnerships with external businesses, identified by a Partner ID, and includes partner name, type of partnership, contact person, contact details, email, and commission rate. Partnerships involve services such as car rentals, tour agencies, and exclusive guest deals.

The system ensures real-time room availability tracking, allowing guests to view and book available rooms instantly. By integrating these functionalities, the Hotel Reservation System enhances guest satisfaction and improves operational efficiency.

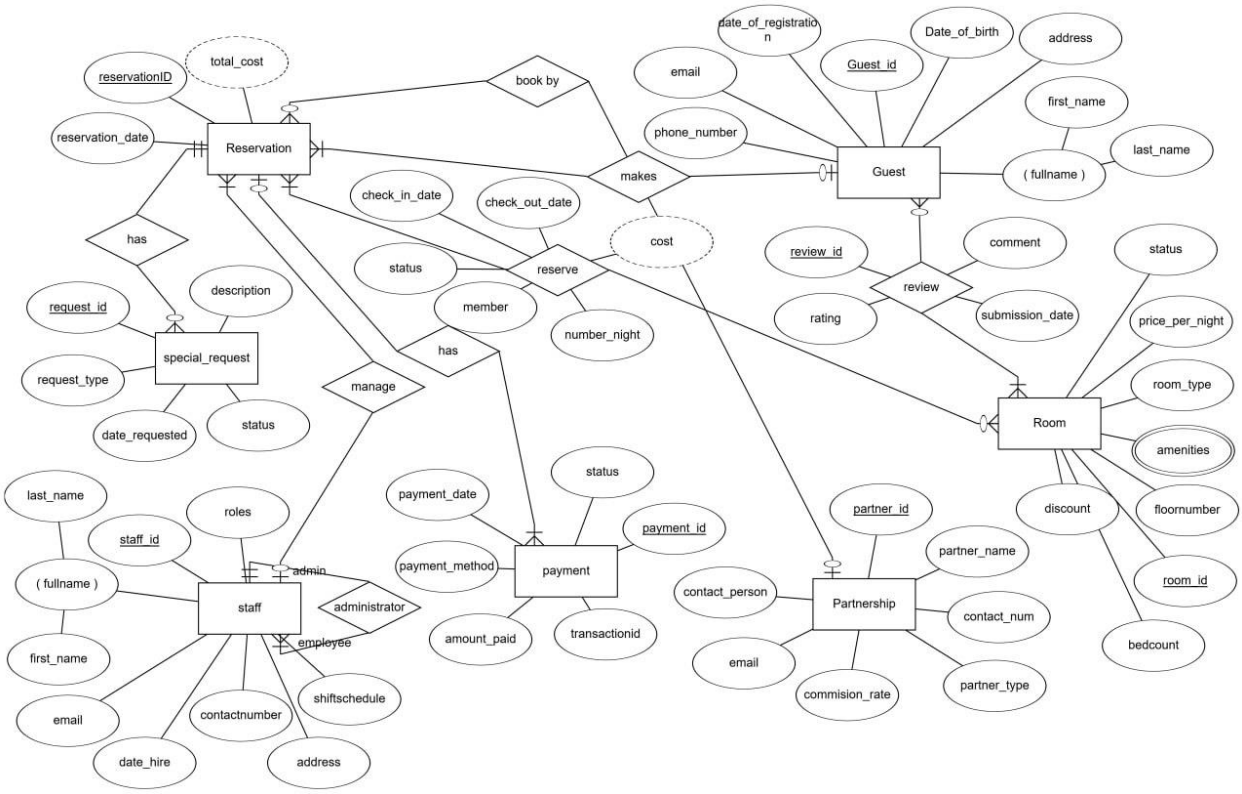
2.2.3 Feature of This project

Features of Hotel Reservation System:
• Room Management
• Reservation Management
• Guest Management

- Staff & Admin Management
- Customer Reviews & Ratings
- Special Requests & Preference

2.2 ER Diagram

Below is the **Entity-Relationship Diagram (ERD)**, which illustrates the relationships between different entities in the system:



2.3 Relationship model

1. Staff

- Primary Key: staff_id
- Foreign Key: admin_id → references Staff(staff_id) (admin)

Attribute	Data Type	Description
staff_id	INT	Primary Key
staff_role	VARCHAR(50)	Staff role (e.g., Receptionist, Manager)
first_name	VARCHAR(50)	Staff first name
last_name	VARCHAR(50)	Staff last name
address	VARCHAR(100)	Staff address
email	VARCHAR(50)	Staff email address
date_hire	DATE	Date the staff member was hired
contact_number	VARCHAR(20)	Staff contact number
shift_schedule	VARCHAR(50)	Work schedule of the staff
admin_id	INT	Foreign Key referencing Staff(staff_id) (admin staff)

2.Special_Request

- Primary Key: request_id
- Foreign Key: reservation_id → references Reservation(reservation_id)

Attribute	Data Type	Description
request_id	INT	Primary Key
reservation_id	INT	Foreign Key referencing Reservation(reservation_id)
request_type	VARCHAR(100)	Type of special request (e.g., Extra Bed)
date_request	DATE	Date when the request was made
description	TEXT	Description of the request
status	VARCHAR(50)	Status of the request

3.Guest

- Primary Key: guest_id

Attribute	Data Type	Description
guest_id	INT	Primary Key
first_name	VARCHAR(50)	Guest first name
last_name	VARCHAR(50)	Guest last name
address	VARCHAR(100)	Guest address
date_of_birth	DATE	Guest's date of birth
date_of_register	DATE	Date the guest registered
phone_number	VARCHAR(20)	Guest phone number
email	VARCHAR(50)	Guest email address

4.Payment

- Primary Key: payment_id
- Foreign Key: reservation_id → references Reservation(reservation_id)

Attribute	Data Type	Description
payment_id	INT	Primary Key
payment_date	DATE	Date of payment
payment_method	VARCHAR(100)	Method of payment (e.g., Credit Card, Cash)
amount_paid	DECIMAL(10,2)	Amount paid by the guest
trancation_id	VARCHAR(50)	Transaction ID
status	VARCHAR(50)	Payment status
reservation_id	INT	Foreign Key referencing Reservation(reservation_id)

5.Reservation

- Primary Key: reservation_id
- Foreign Keys:
    - staff_id → references Staff(staff_id)
    - guest_id → references Guest(guest_id)
    - partner_id → references Partner_ship(partner_id)

Attribute	Data Type	Description
reservation_id	INT	Primary Key
reservation_date	DATE	Date when the reservation was made
staff_id	INT	Foreign Key referencing Staff(staff_id)
guest_id	INT	Foreign Key referencing Guest(guest_id)
partner_id	INT	Foreign Key referencing Partner_ship(partner_id)

6. Partner_ship

- Primary Key: partner_id

Attribute	Data Type	Description
partner_id	INT	Primary Key
partner_name	VARCHAR(100)	Name of the partner
contact_number	VARCHAR(20)	Contact number of the partner
partner_type	VARCHAR(100)	Type of partnership
commision_rate	DECIMAL(4,2)	Commission rate (default 0)
email	VARCHAR(50)	Partner email address
contact_person	VARCHAR(100)	Contact person from the partner company

7.Room

- Primary Key: room_id

Attribute	Data Type	Description
room_id	INT	Primary Key
bed_count	INT	Number of beds in the room
floor_number	INT	Floor where the room is located
price_per_night	DECIMAL(10,2)	Price per night
room_type	VARCHAR(100)	Type of room (e.g., Deluxe, Single)
discount	DECIMAL(4,2)	Discount percentage (default 0)
status	VARCHAR(50)	Room status (e.g., Available, Booked)

8. Review

- Primary Key: review_id
- Foreign Keys:
  - guest_id → references Guest(guest_id)
  - room_id → references Room(room_id)

Attribute	Data Type	Description
review_id	INT	Primary Key
submission_date	DATE	Date the review was submitted
comment	TEXT	Review comment
rating	INT	Rating (0 to 5)
guest_id	INT	Foreign Key referencing Guest(guest_id)
room_id	INT	Foreign Key referencing Room(room_id)

9.Room_amenity

- Primary Key: (room_id, amenity_id) *(composite key)*

- Foreign Keys:
    - room_id → references Room(room_id)
    - amenity_id → references Amenities(amenity_id)

Attribute	Data Type	Description
room_id	INT	Foreign Key referencing Room(room_id)
amenity_id	INT	Foreign Key referencing Amenities(amenity_id)
(PK)	Composite	Primary Key (room_id, amenity_id)

10.Amenities

- Primary Key: amenity_id

Attribute	Data Type	Description
amenity_id	INT	Primary Key
amenity_name	VARCHAR(100)	Name of the amenity

11. Reservation_Room

- Primary Key: (reservation_id, room_id) *(composite key)*
- Foreign Keys:
    - reservation_id → references Reservation(reservation_id)
    - room_id → references Room(room_id)

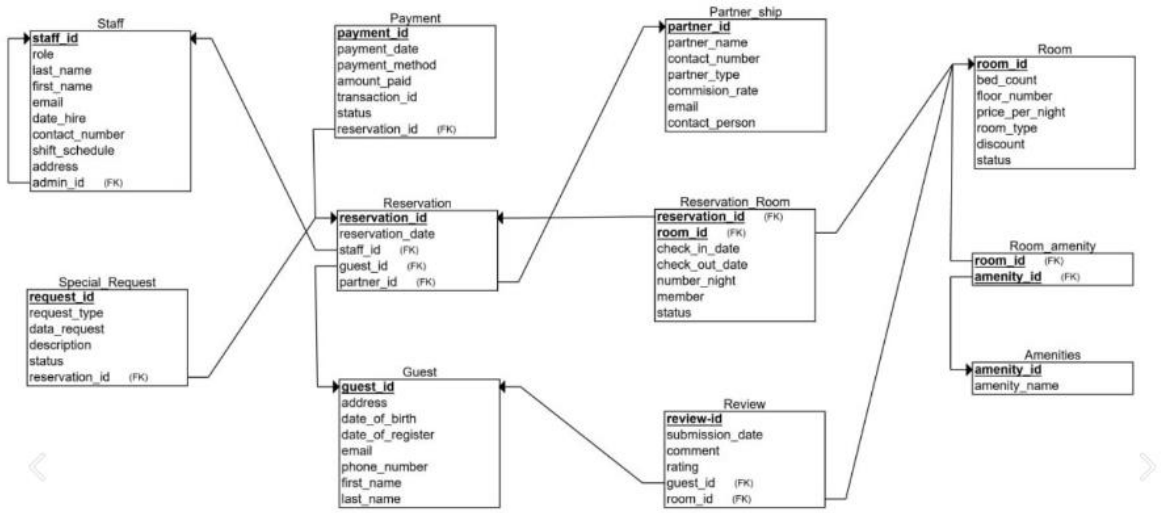
Attribute	Data Type	Description
reservation_id	INT	Foreign Key referencing Reservation(reservation_id)
room_id	INT	Foreign Key referencing Room(room_id)
check_in_date	DATE	Check-in date
check_out_date	DATE	Check-out date
number_night	INT	Number of nights stayed
member	INT	Number of guests staying
status	VARCHAR(50)	Status (e.g., Confirmed, Checked-in)

6. Partner_ship

- Primary Key: partner_id
- Used as Foreign Key in:
    - Reservation.partner_id → references Partner_ship(partner_id)

Attribute	Data Type	Description
partner_id	INT	Primary Key
partner_name	VARCHAR(100)	Name of the partner
contact_number	VARCHAR(20)	Contact number of the partner
partner_type	VARCHAR(100)	Type of partnership
commision_rate	DECIMAL(4,2)	Commission rate (default 0)
email	VARCHAR(50)	Partner email address
contact_person	VARCHAR(100)	Contact person from the partner company

The relational model represents how the database tables are structured and linked:



3.1 DDL (Data Definition Language)

The system consists of the following tables:

- Guest: Stores guest details.
- Room: Contains room information.
- Reservation: Manages booking details.
- Payment: Tracks payment transactions.
- Review: Stores guest feedback.
- SpecialRequest: Manages special accommodations requested by guests.
- Staff: Stores hotel staff details.
- Admin: Contains administrator accounts.

Hotel Database Entities and Relationships

This database includes the following entities and relationships:

- Staff
    - Attributes: staff_id (Primary Key), first_name, last_name, role, contact_number, email, date_hired, shift_schedule, admin_id.
    - Relationship: A self-referencing foreign key (staff_fk) where admin_id references staff_id to capture hierarchy.

```
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY AUTO_INCREMENT,
    staff_role VARCHAR(50),
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    address VARCHAR(100),
    email VARCHAR(50),
    date_hire DATE,
    contact_number VARCHAR(20),
    shift_schedule VARCHAR(50),
    admin_id INT NULL,
    FOREIGN KEY (admin_id) REFERENCES Staff(staff_id) ON DELETE SET NULL
);
```

- **Explanation**: The **Staff** table holds essential information about hotel staff. The admin_id field is a self-referencing foreign key, which creates a hierarchical relationship between staff members, allowing one staff member to be the supervisor of another.

- Room
    - Attributes: room_id (Primary Key), room_type, bedcount, price_per_night, status, floor_number, amenities , discount .

```
CREATE TABLE Room (
    room_id INT PRIMARY KEY AUTO_INCREMENT,
    bed_count INT,
    floor_number INT,
    price_per_night DECIMAL(10,2),
    room_type VARCHAR(100),
    discount DECIMAL(4,2) DEFAULT 0,
    status VARCHAR(50)
);
```

Explanation: The **Room** table contains information about the rooms available in the hotel, including the bed count, price, and room type. The discount column allows for any discounts to be applied to the room price.

- Guest
    - Attributes: guest_id (Primary Key), first_name, last_name, email, phone_number, address, date_of_birth, date_of_register, prefer_payment_method.

```
CREATE TABLE Guest (
    guest_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    address VARCHAR(100),
    date_of_birth DATE,
    date_of_register DATE,
    phone_number VARCHAR(20),
    email VARCHAR(50)
);
```

Explanation: The **Guest** table stores details about the guests who use the hotel's services. This includes their personal information and contact details.

- Reservation
    - Attributes: reservation_id (Primary Key), reservation_date, , guest_id, staff_id partner_id .
    - Foreign Keys:
        - partner_id references Partnership(partner_id).
        - guest_id references Guest(guest_id).
        - staff_id references Staff(staff_id).

```
CREATE TABLE Reservation (
    reservation_id INT PRIMARY KEY AUTO_INCREMENT,
    reservation_date DATE,
    staff_id INT,
    guest_id INT,
    partner_id INT,
    FOREIGN KEY (staff_id) REFERENCES Staff(staff_id) ON DELETE SET NULL,
    FOREIGN KEY (guest_id) REFERENCES Guest(guest_id) ON DELETE SET NULL,
    FOREIGN KEY (partner_id) REFERENCES Partner_ship(partner_id) ON DELETE SET NULL
);
```

Explanation: The **Reservation** table captures the reservation details, such as the reservation date, associated staff, guest, and partner. Foreign keys ensure referential integrity with the **Staff**, **Guest**, and **Partner_ship** tables.

- Reservation_room
    - Attributes: reservation_id (Primary Key), room_id, , Check_in_date, check_out_date,number_night , member .
    - Foreign Keys:
        - Reservation_id references Partnership(reservation).
        - Room_id references Room(room_id).

```
CREATE TABLE Reservation_Room (
    reservation_id INT,
    room_id INT,
    check_in_date DATE,
    check_out_date DATE,
    number_night INT,
    member INT,
    status VARCHAR(50),
    PRIMARY KEY (reservation_id, room_id),
    FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id),
    FOREIGN KEY (room_id) REFERENCES Room(room_id)
);
```

Explanation: The **Reservation_Room** table manages the relationship between rooms and reservations. It tracks which rooms are assigned to which reservations, including check-in/check-out dates.

- Payment
    - Attributes: payment_id (Primary Key), payment_date, payment_method, transaction_id, amount_paid, reservation_id.
    - Foreign Key: reservation_id references Reservation(reservation_id).

```
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    payment_date DATE,
    payment_method VARCHAR(100),
    amount_paid DECIMAL(10,2),
    transaction_id VARCHAR(50),
    status VARCHAR(50),
    reservation_id INT,
    FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id) ON DELETE SET NULL
);
```

Explanation: The **Payment** table records the payment transactions for reservations. Each payment is linked to a reservation, and the table tracks the payment method and amount.

- SpecialRequest
  - Attributes: request_id (Primary Key), request_type, description, status, date_requested, guest_id.
  - Foreign Key: guest_id references Guest(guest_id).

```
CREATE TABLE Special_Request (
    request_id INT PRIMARY KEY AUTO_INCREMENT,
    reservation_id INT,
    request_type VARCHAR(100),
    date_request DATE,
    description TEXT,
    status VARCHAR(50),
    FOREIGN KEY (reservation_id) REFERENCES Reservation(reservation_id) ON DELETE SET NULL
);
```

Explanation: The **Special_Request** table records special accommodation requests by guests. This may include extra beds or specific room arrangements. The table references the **Reservation** table.

- Review
    - Attributes: review_id (Primary Key), rating, review_date, comments, guest_id, room_id.
    - Foreign Keys:
        - guest_id references Guest(guest_id).
        - room_id references Room(room_id).

```
CREATE TABLE Review (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    submission_date DATE,
    comment TEXT,
    rating INT CHECK (rating >= 0 AND rating <= 5),
    guest_id INT,
    room_id INT,
    FOREIGN KEY (guest_id) REFERENCES Guest(guest_id) ON DELETE SET NULL,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE SET NULL
);
```

- **Explanation**: The **Review** table stores guest feedback about their stay. Reviews are associated with both guests and rooms, with a rating between 0 and 5.

The system comprises several essential tables that interact with each other using foreign keys and primary keys to maintain referential integrity. Below is the corrected and updated SQL code, which establishes the foundation of the database schema.

3.2 Data Dictionaries (Table Descriptions)

Each table in the database is designed with appropriate attributes, data types, and constraints to ensure referential integrity, data consistency, and performance optimization. The tables are normalized to reduce data redundancy and enforce relational connections between entities such as guests, rooms, reservations, staff, and partners.

Below is a summary of the key tables in the system:

1. Guest

Stores information about hotel guests.

	Field	Type	Null	Key	Default	Extra
1	guest_id	int	NO	PRI	[NULL]	auto_increment
2	first_name	varchar(50)	NO		[NULL]	
3	last_name	varchar(50)	NO		[NULL]	
4	address	varchar(100)	NO		[NULL]	
5	date_of_birth	date	NO		[NULL]	
6	date_of_registe	date	NO		[NULL]	
7	phone_number	varchar(20)	NO		[NULL]	
8	email	varchar(50)	NO		[NULL]	

2. Room

Contains details about rooms available for reservation.

	Field	Type	Null	Key	Default	Extra
1	room_id	int	NO	PRI	[NULL]	auto_increment
2	bed_count	int	NO		[NULL]	
3	floor_number	int	NO		[NULL]	
4	price_per_night	decimal(10,2)	NO		[NULL]	
5	room_type	varchar(100)	NO		[NULL]	
6	discount	decimal(4,2)	NO		0.00	
7	status	varchar(50)	NO		[NULL]	

3. Reservation

Records reservation information made by guests.

	Field	Type	Null	Key	Default	Extra
1	reservation_id	int	NO	PRI	[NULL]	auto_increment
2	reservation_dat	date	NO		[NULL]	
3	staff_id	int	YES	MUL	[NULL]	
4	guest_id	int	YES	MUL	[NULL]	
5	partner_id	int	YES	MUL	[NULL]	

4. Reservation_Room

Associates rooms with reservations (many-to-many relationship).

	Field	Type	Null	Key	Default	Extra
1	reservation_id	int	NO	PRI	[NULL]	
2	room_id	int	NO	PRI	[NULL]	
3	check_in_date	date	NO		[NULL]	
4	check_out_date	date	NO		[NULL]	
5	number_night	int	NO		[NULL]	
6	member	int	NO		[NULL]	
7	status	varchar(50)	NO		[NULL]	

5. Staff

Stores data about hotel staff members.

	Field	Type	Null	Key	Default	Extra
1	staff_id	int	NO	PRI	[NULL]	auto_increment
2	staff_role	varchar(50)	NO		[NULL]	
3	first_name	varchar(50)	NO		[NULL]	
4	last_name	varchar(50)	NO		[NULL]	
5	address	varchar(100)	NO		[NULL]	
6	email	varchar(50)	NO		[NULL]	
7	date_hire	date	NO		[NULL]	
8	contact_numbe	varchar(20)	NO		[NULL]	
9	shift_schedule	varchar(50)	NO		[NULL]	
10	admin_id	int	YES	MUL	[NULL]	

6. Partner_Ship

Contains information about third-party partners working with the hotel.

	Field	Type	Null	Key	Default	Extra
1	partner_id	int	NO	PRI	[NULL]	auto_increment
2	partner_name	varchar(100)	NO		[NULL]	
3	contact_numbe	varchar(20)	NO		[NULL]	
4	partner_type	varchar(100)	NO		[NULL]	
5	commision_rate	decimal(4,2)	NO		0.00	
6	email	varchar(50)	NO		[NULL]	
7	contact_person	varchar(100)	NO		[NULL]	

7.Special_request

Contain information guest request for extra.

	Field	Type	Null	Key	Default	Extra
1	request_id	int	NO	PRI	[NULL]	auto_increment
2	reservation_id	int	YES	MUL	[NULL]	
3	request_type	varchar(100)	NO		[NULL]	
4	date_request	date	NO		[NULL]	
5	description	text	NO		[NULL]	
6	status	varchar(50)	NO		[NULL]	

8. Amenities

Lists available room amenities.

	Field	Type	Null	Key	Default	Extra
1	amenity_id	int	NO	PRI	[NULL]	auto_increment
2	amenity_name	varchar(100)	NO		[NULL]	

9.Room_Amenity

Links rooms to their available amenities (many-to-many relationship).

	Field	Type	Null	Key	Default	Extra
1	room_id	int	NO	PRI	[NULL]	
2	amenity_id	int	NO	PRI	[NULL]	

10. Review

Captures guest reviews for rooms or stays.

	Field	Type	Null	Key	Default	Extra
1	review_id	int	NO	PRI	[NULL]	auto_increment
2	submission_dat	date	NO		[NULL]	
3	comment	text	YES		[NULL]	
4	rating	int	NO		[NULL]	
5	guest_id	int	YES	MUL	[NULL]	
6	room_id	int	YES	MUL	[NULL]	

7. Payment

Manages special payment methods or discounts linked to reservations.

	Field	Type	Null	Key	Default	Extra
1	payment_id	int	NO	PRI	[NULL]	auto_increment
2	payment_date	date	NO		[NULL]	
3	payment_meth	varchar(100)	NO		[NULL]	
4	amount_paid	decimal(10,2)	NO		[NULL]	
5	trancation_id	varchar(50)	NO		[NULL]	
6	status	varchar(50)	NO		[NULL]	
7	reservation_id	int	YES	MUL	[NULL]	

The full data dictionary above outlines the relational design of the system and highlights how different entities interact with each other. These well-structured tables form the backbone of the hotel reservation system and allow for data validation, business logic enforcement, and reporting functionalities.

3.3 DML (Insert, Update, Delete Operations)

- **INSERT:** Used for adding new guests, reservations, and payments.
- **UPDATE:** Allows modifications of reservations, room status, and guest information.
- **DELETE:** Removes outdated reservations and other unnecessary records.

Insert Data Operations

1. Insert More Data into Staff Table

INSERT INTO Staff (staff_role, first_name, last_name, address, email, date_hire, contact_number, shift_schedule, admin_id)

VALUES

('Housekeeper', 'Rachel', 'Adams', '500 Birch St, City', 'rachel.adams@example.com', '2024-02-01', '555-9988', 'Evening', 4),

('Receptionist', 'Samuel', 'Clark', '150 Oak St, City', 'samuel.clark@example.com', '2024-03-05', '555-7766', 'Morning', 2),

('Security', 'William', 'Harris', '250 Pine St, City', 'william.harris@example.com', '2024-04-01', '555-4455', 'Night', NULL),

('Concierge', 'Lily', 'Walker', '350 Maple St, City', 'lily.walker@example.com', '2024-03-20', '555-5566', 'Evening', 5);

2. Insert More Data into Guest Table

INSERT INTO Guest (first_name, last_name, address, date_of_birth, date_of_register, phone_number, email)

VALUES

('Olivia', 'Martins', '500 Maple St, City', '1987-04-15', '2024-03-25', '555-7788', 'olivia.martins@example.com'),

('Ethan', 'Garcia', '123 Oak St, City', '1992-09-09', '2024-03-15', '555-9988', 'ethan.garcia@example.com'),

('Ava', 'Scott', '777 Birch St, City', '1995-12-03', '2024-02-18', '555-1122', 'ava.scott@example.com'),

('Isabella', 'Rodriguez', '222 Cedar St, City', '1989-11-11', '2024-03-08', '555-3344', 'isabella.rodriguez@example.com');

3. Insert More Data into Room Table

INSERT INTO Room (bed_count, floor_number, price_per_night, room_type, discount, status)

VALUES

(2, 3, 170.00, 'Double', 0.05, 'Available'),

(1, 4, 130.00, 'Single', 0.10, 'Occupied'),

(3, 5, 220.00, 'Suite', 0.15, 'Available'),

(1, 2, 110.00, 'Single', 0.05, 'Occupied'),

(2, 3, 160.00, 'Double', 0.10, 'Available');

4. Insert More Data into Reservation Table

INSERT INTO Reservation (reservation_date, staff_id, guest_id, partner_id)

VALUES

('2024-03-25', 2, 7, 4),

('2024-03-26', 3, 8, 6),

('2024-03-27', 2, 9, 5),

('2024-03-28', 1, 10, 2);

Update Data Operations

1.Updating Room Status

UPDATE Room

SET status = 'Occupied'

WHERE room_id = 3;

2. Updating Guest Information

UPDATE Guest

SET phone_number = '555-9999', email = 'olivia.updated@example.com'

WHERE first_name = 'Olivia' AND last_name = 'Martins';

3. Updating Reservation Details

UPDATE Reservation

SET reservation_date = '2024-03-29'

WHERE reservation_id = 7;

Delete Data Operations

1.Deleting an Outdated Reservation

D ELETE FROM Reservation

WHERE reservation_date < '2024-01-01';

2.Deleting a Guest Record

DELETE FROM Guest

WHERE guest_id = 15;

3. Deleting a Room Record

DELETE FROM Room

WHERE room_id = 20;

DML operations play a crucial role in managing the database by enabling efficient data insertion, updating, and deletion. Proper validation should be enforced to prevent invalid data entries, ensuring data consistency and integrity within the system.

3.4 DQL (Data Query Language)

Data Query Language (DQL) is primarily used to retrieve and analyze data from the hotel reservation system efficiently. It supports both operational and managerial decision-making by providing detailed, real-time insights from the database. Below are key types of queries implemented:

1. Room Availability and Details

- Retrieve all available rooms.
- Find available rooms below a specific price threshold.
- Check room occupancy reports and monthly occupancy rates.
- Calculate room occupancy rate per room type.

2. Guest Management

- Fetch guest reservation history.
- List guests who registered after a certain date.
- Identify guests with multiple bookings.
- Retrieve all reviews submitted by specific guests.
- Identify high-spending guests and top 5 guests by total spending.

3. Reservation Insights

- Get all reservations made by each guest in a specific year.
- View detailed reservations and their associated guests.
- Retrieve reservations with specific guest names.
- List reservations with special requests (e.g., Early Check-out).
- Calculate average booking duration by room type.

4. Reviews and Feedback

- Display all reviews by room or guest.
- Retrieve reviews for specific room numbers.

5. Staff and Shift Reports

- View all staff members and their assigned roles and shifts.

6. Special Requests

- List all pending special requests.
- Find special requests linked to a specific room.

7. Partner and Agency Collaboration

- Retrieve reservations made through specific partners or travel agencies.

8. Payments and Revenue

- Retrieve completed payment transactions.
- Identify late payments (made after check-out).
- Calculate total revenue from bookings.
- Generate monthly revenue reports.
- Analyze revenue per room type.

These DQL queries play a critical role in monitoring operations, generating reports, and ensuring better customer service through data-driven decisions. They allow hotel staff and managers to stay updated on guest behaviors, revenue trends, occupancy levels, and service quality.

1. retrieves the total number of reservations made by each guest in 2024

2. select all staff members

3.Find All Guests Who Registered After January 1, 2024

4.Find Available Rooms with Price Below 150

5. List All Reviews for Room 1

6. list all review

7. Get Special Requests Pending Approval

8. Find Reservations Made by Guest 'Michael Taylor'

9. Get Payments for Completed Reservations

10. Get Room Amenities for Room 2

11. Find All Special Requests for Room 1

12. Get Reservations with Special Requests for Early Check-out

13. List All Staff with Their Roles and Shift Schedules

14. Get All Reviews for a Specific Guest (e.g., Guest ID 1)

15. . Find All Reservations by Partner 'TravelAgencyA'

16. retrieves the payment records where the payment was made before the guest checked out

17. List Available Rooms

18. Get Booking Details for a Guest

19. Total Revenue from Bookings

20. retrieve information about payments made for reservations

21. Calculate Occupancy Rate for Each Room Type

22. Find All Guests with Multiple Bookings

23. Calculate Average Booking Duration by Room Type

24. Total Revenue per Room Type

25. Identify High-Spending Guests (Guests Who Have Spent More Than a Given Amount)

26. Find Most Popular Room Type Based on Bookings

27. Identify Top 5 Guests by Spending

3.5 Views

Views are virtual tables created to simplify data access, enhance reporting, and improve the readability of complex queries. They serve as a user-friendly interface for interacting with the underlying database structure while hiding complex joins and calculations. The following views were designed to support hotel management operations:

1. Guest Reservation Summary View (vw_guest_reservations)

This view provides a summary of each guest's reservation activity. It combines guest details with their reservation history, displaying:

- Guest ID and full name
- Contact information (email, phone)
- Total number of reservations
- Total nights stayed
- The date of their most recent visit

This view is particularly useful for guest profiling, customer relationship management, and identifying loyal or frequent visitors.

2. Room Occupancy View (vw_room_occupancy)

This view shows occupancy data for each room, including:

- Room ID, type, floor, and price
- Number of bookings made during the current year
- Total nights occupied
- Booking percentage (based on total bookings in the current year)

It assists management in tracking room utilization, setting pricing strategies, and identifying underperforming rooms.

3. Revenue Analysis View (vw_revenue_analysis)

This view provides monthly revenue statistics, including:

- Total revenue earned per month
- Revenue from partner referrals vs. direct bookings
- Total unique guests and reservations per month

This view is essential for financial reporting, forecasting, and evaluating the impact of partnerships.

4. Staff Performance View (vw_staff_performance)

This view evaluates each staff member's contributions, showing:

- Number of reservations handled
- Revenue generated from those reservations
- Number of unique guests served
- Average guest rating based on reviews

This data helps assess staff efficiency, reward high performers, and identify training needs.

5. Room Amenities View (vw_room_amenities)

This view consolidates amenities per room into a readable format, displaying:

- Room details (ID, type, price)
- A comma-separated list of amenities

Useful for generating room listings and descriptions for websites or brochures.

6. Partner Performance View (vw_partner_performance)

This view tracks the performance of third-party partners by showing:

- Number of reservations they referred
- Total revenue generated
- Commissions earned
- Unique guests referred

This view supports partnership evaluation and commission tracking.

7. Upcoming Reservations View (vw_upcoming_reservations)

This operational view lists all upcoming reservations, including:

- Guest name and room information
- Check-in/check-out dates
- Reservation status
- Staff member assigned

This is particularly helpful for front desk planning and daily operational management.

8. Guest Reviews View (guest_reviews)

This view collects reviews left by guests along with:

- Guest name
- Review comment and rating
- Room type reviewed

It supports quality control and guest satisfaction analysis.

Summary:

These views provide structured access to key data, reducing query complexity and improving operational decision-making. For example:

- vw_guest_reservations allows quick guest profiling.
- vw_room_occupancy supports occupancy and revenue optimization.
- vw_staff_performance provides insight into employee contributions.

Each view has been designed to align with real-world business requirements in hotel management, streamlining data access for decision-makers and analysts.

3.6 Stored Procedures and Functions

Stored Procedures

Stored procedures are used to automate and streamline various operations within the hotel reservation system. These operations include tasks such as reservation confirmation, payment processing, and report generation. The stored procedures are essential for ensuring consistency, reducing redundancy, and improving overall efficiency.

Examples of Stored Procedures:

1. Adding a New Room
   The add_new_room stored procedure inserts a new room into the system by providing details such as bed count, floor number, price per night, room type, discount, and status. This ensures that new rooms are added efficiently and consistently.
2. Updating Room Status
   The update_room_status stored procedure allows the status of a room (e.g., Available, Occupied) to be updated based on specific criteria. It helps manage the room's availability in real time.
3. Canceling a Reservation
   The cancel_reservation procedure handles the cancellation of reservations by updating the room status to "Available," deleting the room assignment from the reservation, and removing the reservation record itself.
4. Processing Payments
   The ProcessPayment procedure is used to record payments against reservations. It tracks the payment method, amount paid, and associated transaction ID, and marks the status as "Completed."
5. Assigning Rooms to Reservations
   The AssignRoom procedure assigns specific rooms to a reservation. It calculates the number of nights based on check-in and check-out dates, updates room status to "Occupied," and inserts the assignment into the Reservation_Room table.

6. Adding Special Requests
   The add_special_request procedure allows guests to make special requests, such as early check-in or additional services, linked to their reservation. It adds these requests into the Special_Request table, with a "Pending" status.
7. Refunding Payments
   The refund_payment procedure updates the status of a payment to "Refunded," ensuring accurate financial records are maintained in the system.

By using these stored procedures, operations such as room assignment, payment processing, and guest management are automated, reducing human error and improving the speed of processing.

Functions

Functions are used to return specific values or perform calculations based on input parameters. Unlike stored procedures, functions are typically used within SQL queries, enabling dynamic and reusable data processing without modifying the database state.

In this hotel reservation system, several functions have been implemented to support calculations, validations, and real-time insights. Below are key examples:

1. Calculating Total Reservation Cost

A function named get_total_reservation_cost is used to compute the total cost of a reservation. It calculates the price by multiplying the number of nights with the room’s price per night and applies any available discount. This ensures consistent and accurate billing during the reservation process.

SELECT get_total_reservation_cost(reservation_id);

2. Counting Staff Reservations

The function get_staff_reservation_count returns the number of reservations made by a specific staff member. This helps in tracking staff performance and analyzing workload distribution.

SELECT get_staff_reservation_count(staff_id);

3. Calculating Partner Earnings

The function partner_ship_cost calculates the total commission earned by a partner within a specific date range. This is useful for reporting and revenue-sharing purposes with third-party partners.

SELECT partner_ship_cost(partner_id, 'start_date', 'end_date');

4. Checking Room Availability

The check_available_room function determines whether a specific room is available. It checks the room's status in the system and returns 1 if the room is available, otherwise 0. This enables real-time availability checks.

SELECT check_available_room(room_id);

5. Getting Total Revenue Within Date Range

The get_total_price_reservation function calculates the total revenue from all reservations made between two specific dates. This function supports managerial reporting and financial analysis.

SELECT get_total_price_reservation('start_date', 'end_date');

6. Finding the Most Booked Room

The function get_room_most_book identifies the room that has been booked the most frequently. This can guide promotional strategies and room management.

SELECT get_room_most_book();

7. Finding the Guest with the Most Bookings

The get_guest_most_book function returns the ID and full name of the guest who made the most bookings. This helps recognize frequent guests and could be used for loyalty programs.

SELECT get_guest_most_book();

Function Benefits in the System

By using functions for lightweight and frequent operations like calculations, availability checks, and validations, the system maintains high performance and consistency. Functions offer:

- Reusability: They can be reused in multiple queries without duplication of logic.
- Accuracy: Ensures consistent results across different modules.
- Performance: Enhances efficiency by executing faster than complex procedures for simple tasks

3.7 Trigger

Definition

A trigger is a block of procedural code that automatically executes in response to specific events on a table or view in a database. Triggers help maintain data consistency, integrity, and automation by enforcing business rules directly at the database level.

In the hotel management system, triggers are used to enforce automatic updates, maintain logs, and ensure synchronization between related tables without requiring manual intervention

Purpose of Using Triggers

- Automatically update related records based on certain actions.
- Log important changes (INSERT, UPDATE, DELETE) for auditing.
- Maintain data integrity across interdependent tables.
- Minimize human errors by automating database responses.

Examples of Triggers Implemented

1. Staff Audit Triggers

- Trigger Names: staff_after_insert, staff_after_update, staff_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Staff table
- Function: Inserts historical data into Audit_Staff table whenever staff data is added, updated, or removed.
- Purpose: Maintains a detailed history of all staff changes for administrative control.

2. Guest Audit Triggers

- Trigger Names: guest_after_insert, guest_after_update, guest_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Guest table
- Function: Tracks all guest records by saving them into Audit_Guest for monitoring or rollback.
- Purpose: Ensures traceability of guest information and actions taken.

3. Room Audit Triggers

- Trigger Names: room_after_insert, room_after_update, room_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Room table
- Function: Logs all room details into Audit_Room to monitor changes such as status, price, and type.

- Purpose: Maintains an audit trail for room inventory and pricing.

4. Reservation Audit Triggers

- Trigger Names: reservation_after_insert, reservation_after_update, reservation_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Reservation table
- Function: Records reservation actions in Audit_Reservation, preserving history even if original data changes.
- Purpose: Ensures a complete log of reservations for reporting and backup purposes.

5. Reservation_Room Audit Triggers

- Trigger Names: reservation_room_after_insert, reservation_room_after_update, reservation_room_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Reservation_Room table
- Function: Archives room booking details to Audit_Reservation_Room for each reservation.
- Purpose: Tracks room assignment per reservation for accuracy and investigation.

6. Payment Audit Triggers

- Trigger Names: payment_after_insert, payment_after_update, payment_after_delete
- Events: AFTER INSERT, UPDATE, DELETE on Payment table
- Function: Saves each payment action into Audit_Payment, including status and transaction ID.
- Purpose: Provides transparency for all payment activities, aiding in financial auditing.

Trigger Testing

Each trigger was tested by performing corresponding actions on the database (e.g., adding or deleting a staff member, making a payment, updating room info). The respective audit tables were successfully populated with correct historical data, confirming the expected behavior and reliability of the triggers.

III. Conclusion and Future Work

4.1 Outcome / Completed

The project successfully delivers:

- A structured relational database for hotel management, ensuring seamless data organization.
- Functional queries and operations for efficient reservation processing, including room bookings and guest management.
- Secure data handling and optimization techniques for future scalability, allowing the system to handle increasing data volumes.

4.2 Strong/Weak Points

Strong Points:

- Efficient management of reservations and payments, reducing operational overhead.
- Real-time tracking of room availability, allowing for immediate updates on guest status.
- A user-friendly database structure, ensuring ease of use for staff and future system integrations.

Weak Points:

- Limited automation of certain administrative tasks, requiring manual intervention in some cases.
- Absence of integration with third-party payment gateways, restricting the payment options available to guests.

4.3 Challenges / Difficulty

- Ensuring data consistency across multiple tables, particularly in scenarios involving frequent updates to reservations and guest information.
- Managing concurrent bookings without conflicts, especially during high-demand periods.
- Implementing secure payment transactions to protect sensitive guest information and prevent fraud.

4.4 Future Work

- **Integration with an online booking platform**: Expanding the system to support external booking systems, allowing for direct reservations from hotel websites.
- **Mobile application support**: Developing a mobile app for guests to manage reservations, check room availability, and make payments.
- **AI-driven room pricing recommendations**: Incorporating machine learning models to adjust room pricing based on demand, seasonality, and customer preferences, improving profitability.

4.5 Conclusion

The Hotel Reservation System provides an efficient, scalable solution for hotel management, enhancing reservation processing and room availability tracking. With further enhancements, such as integrating third-party payment systems, adding mobile app support, and incorporating AI, it has the potential to evolve into a fully automated, intelligent booking platform. These future developments will ensure a seamless and enhanced experience for both guests and hotel staff.