**HOTEL RESERVATION SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

AP22110010067-B.THANMAI

AP22110010070-M.LAKSHMI AASRITHA

AP22110010083-V.BHAGYA SATYA SRI

**A picture containing text

Description automatically generated**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[December, 2022]**

**INDEX**

Background………………………………………………………………….………3

Description of the Project…………………………………………………….3

ER Diagram…………………………………………………………………………..7

Realational model………………………………………………………………..7

Description of ER Diagram……………………………………………………8

ER Diagram in the form of Tables……………..………………………..10

Description of Tables…………...……………………………………….…..11

Normalization…………………………………………………………………...13

Creation of Data in the tables……………….…………………………..15

SQL Queries on the created tables….…………………………………18

Creation of views using tables…..………………………………..…….21

**Project Background:**The background of the project outlines the primary objectives and scope of developing a database management system (DBMS) tailored for efficiently managing hotel reservations. Below is a detailed description of the project background:

Introduction: The hospitality industry, particularly hotel management, involves intricate operations requiring meticulous attention to detail. Hotel reservations, one of the core aspects of hotel management, encompass a wide range of activities, from booking rooms to managing customer information, handling reservations, and facilitating check-in and check-out processes. A robust database management system tailored specifically for hotel reservations can significantly enhance operational efficiency, streamline processes, and improve customer satisfaction.

Objectives: The primary objectives of the project are as follows:

1. Efficient Reservation Handling: Develop a system capable of efficiently handling hotel room reservations, ensuring accuracy, and minimizing errors.
2. Customer Information Management: Implement a robust system for managing customer information, including personal details, contact information, and booking history.
3. Reservation Management: Create functionalities for managing reservations, including booking, modification, cancellation, and tracking.
4. Check-in and Check-out Processes: Facilitate seamless check-in and check-out processes, reducing waiting times and enhancing the overall guest experience.

**Description of the Project**

The system will allow users to search for available rooms based on their preferences, book rooms for specific dates, manage bookings, and handle payments. It will also include features for managing customer information, room types, pricing, and availability.

**1. Room Availability Search:**

* Users can search for available rooms based on their preferences, such as room type, check-in and check-out dates, and any additional criteria like price range or amenities.
* The system retrieves and displays a list of available rooms matching the user's criteria, along with relevant details such as room number, type, price per night, and availability status.

**2. Room Booking:**

* Once users find a suitable room, they can proceed to book it for specific dates.
* The system allows users to select the desired room and specify the check-in and check-out dates.
* Users may also have the option to add special requests or preferences, such as extra beds or specific room features.

**3. Booking Management:**

* Users can view and manage their bookings through the system's interface.
* They can modify booking details, such as changing the check-in or check-out dates, updating room preferences, or canceling bookings.
* The system automatically adjusts room availability and pricing based on booking modifications or cancellations.

**4. Payment Handling:**

* Upon confirming a booking, users proceed to make payments securely through the system.
* The system supports various payment methods, including credit/debit cards, online payment gateways, or cash payments at the hotel.
* Users receive payment confirmation and booking details via email or SMS.

**5. Customer Information Management:**

* The system maintains a centralized database of customer information, including personal details, contact information, and booking history.
* Users can create accounts or profiles within the system to store their information securely and expedite future bookings.
* Hotel staff can access and update customer records as needed, ensuring accurate and up-to-date information.

**6. Room Type Management:**

* The system provides features for managing room types, including defining different room categories (e.g., standard, deluxe, suite) and setting corresponding prices and availability.
* Hotel administrators can add, modify, or remove room types based on changing requirements or seasonal variations.
* Each room type may have associated attributes such as capacity, amenities, and special features, which are displayed to users during the booking process.

**7. Pricing and Availability Management:**

* The system dynamically manages room pricing and availability based on factors such as demand, seasonality, and promotional offers.
* Hotel administrators can set flexible pricing rules, discounts, and special rates for different room types, booking periods, and customer segments.
* Real-time availability updates ensure that users receive accurate information about room availability and pricing at all times.

**8. Reporting and Analytics:**

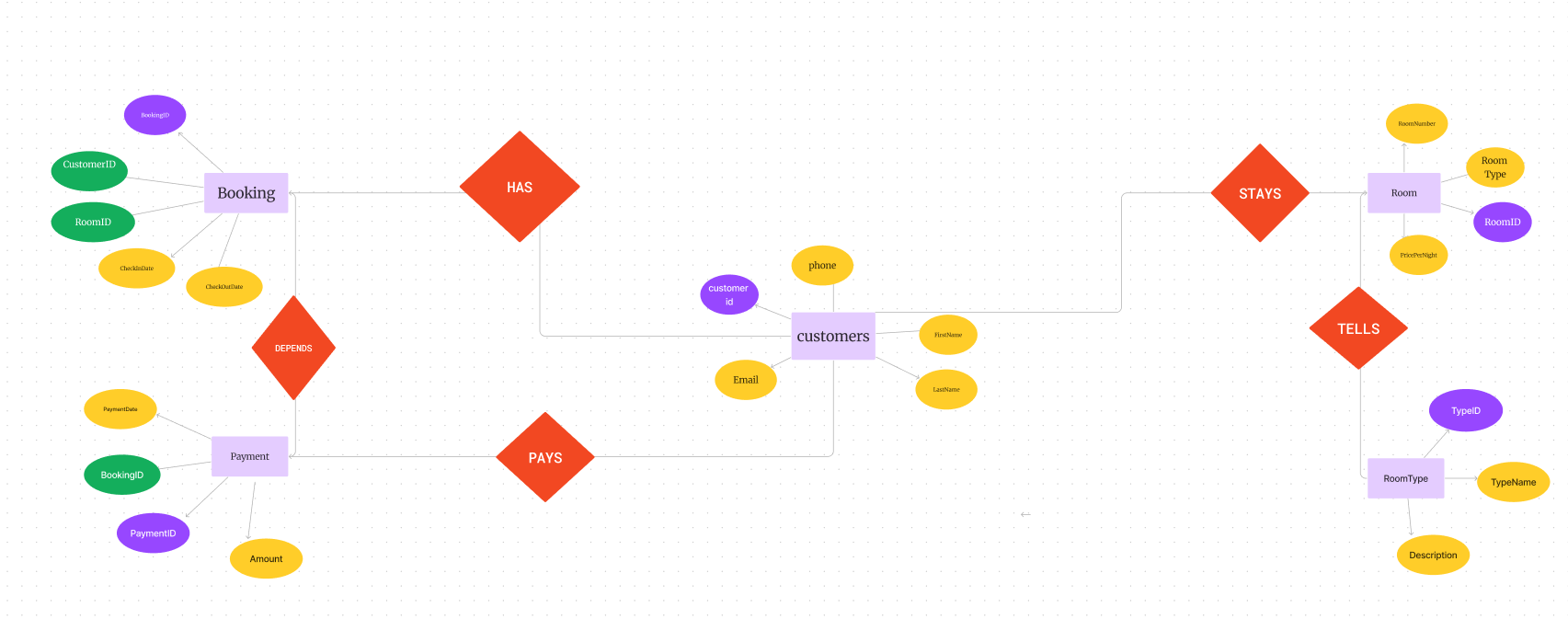
* The system generates comprehensive reports and analytics to track key performance indicators such as occupancy rates, revenue, booking trends, and customer satisfaction scores.
* Hotel administrators can access insights and analytics dashboards to make data-driven decisions, optimize pricing strategies, and enhance overall operational efficiency.

**9. User Authentication and Security:**

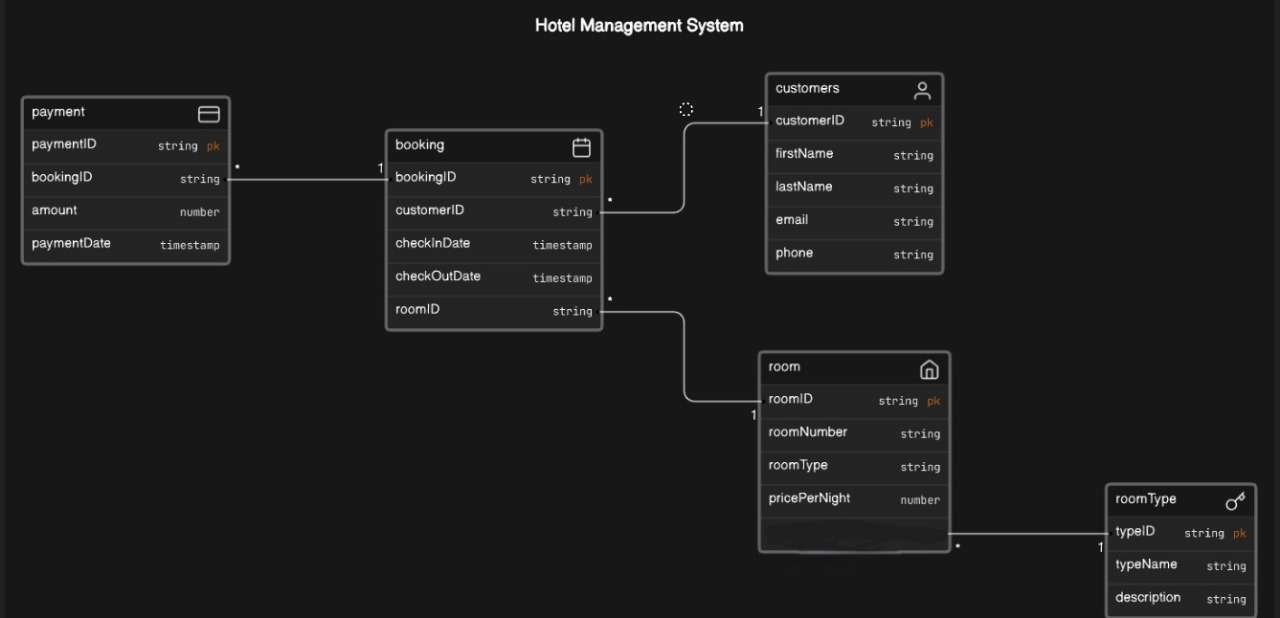
* The system implements robust user authentication mechanisms to ensure secure access to sensitive information and functionalities.
* Role-based access control (RBAC) allows administrators, staff, and customers to access only the features and data relevant to their roles.
* Data encryption, secure transmission protocols, and regular security audits enhance the system's overall security posture and protect against data breaches or unauthorized access.

**10. Integration and Scalability:**

* The system is designed to integrate seamlessly with other hotel management systems, third-party APIs, or external services such as online travel agencies (OTAs) and payment gateways.
* It is built on scalable architecture to accommodate growth and expansion, supporting additional features, properties, or user volume as needed.

**ER DIAGRAM**

**Relational Model**



**DESCRIPTION OF ER DIAGRAM**

**Entities and Attributes:**

1. **Customers**:
   * **Attributes**: CustomerID (Primary Key), FirstName, LastName, Email, Phone.
2. **Room**:
   * **Attributes**: RoomID (Primary Key), RoomNumber, RoomType, PricePerNight.
3. **Booking**:
   * **Attributes**: BookingID (Primary Key), CustomerID (Foreign Key referencing Customers), RoomID (Foreign Key referencing Room), CheckInDate, CheckOutDate.
4. **Payment**:
   * **Attributes**: PaymentID (Primary Key), BookingID (Foreign Key referencing Booking), Amount, PaymentDate.
5. **RoomType**:
   * **Attributes**: TypeID (Primary Key), TypeName, Description.

**Relationships:**

1. **One-to-Many Relationship between Customers and Booking**:
   * Each customer can have multiple bookings, but each booking belongs to only one customer.
2. **One-to-Many Relationship between Room and Booking**:
   * Each room can have multiple bookings, but each booking is associated with only one room.
3. **One-to-Many Relationship between Booking and Payment**:
   * Each booking can have multiple payments, but each payment is related to only one booking.
4. **Many-to-One Relationship between RoomType and Room**:
   * Multiple rooms can have the same room type, but each room belongs to only one room type.

**Foreign Keys:**

* The CustomerID in the Booking table references the CustomerID in the Customers table, establishing a foreign key relationship to maintain data integrity.
* The RoomID in the Booking table references the RoomID in the Room table, establishing a foreign key relationship to maintain data integrity.
* The BookingID in the Payment table references the BookingID in the Booking table, establishing a foreign key relationship to maintain data integrity.

**Key Constraints:**

* Each table has a primary key constraint to ensure the uniqueness of key attributes (CustomerID, RoomID, BookingID, PaymentID, TypeID).
* Foreign key constraints maintain referential integrity between related tables (e.g., CustomerID in Booking referencing CustomerID in Customers).

**Attributes Descriptions:**

* **CustomerID**: Unique identifier for each customer.
* **FirstName**: First name of the customer.
* **LastName**: Last name of the customer.
* **Email**: Email address of the customer.
* **Phone**: Phone number of the customer.
* **RoomID**: Unique identifier for each room.
* **RoomNumber**: Number assigned to the room.
* **RoomType**: Type of the room (e.g., Single, Double, Suite).
* **PricePerNight**: Price per night for the room.
* **BookingID**: Unique identifier for each booking.
* **CheckInDate**: Date when the booking starts.
* **CheckOutDate**: Date when the booking ends.
* **PaymentID**: Unique identifier for each payment transaction.
* **Amount**: Amount paid for the booking.
* **PaymentDate**: Date when the payment was made.
* **TypeID**: Unique identifier for each room type.
* **TypeName**: Name of the room type.
* **Description**: Description of the room type.

**Conversion of ER diagram into Tables**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(15)

);

CREATE TABLE Room (

RoomID INT PRIMARY KEY,

RoomNumber VARCHAR(10),

RoomType VARCHAR(50),

PricePerNight DECIMAL(10, 2)

);

CREATE TABLE Booking (

BookingID INT PRIMARY KEY,

CustomerID INT,

RoomID INT,

CheckInDate DATE,

CheckOutDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (RoomID) REFERENCES Room(RoomID)

);

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY,

BookingID INT,

Amount DECIMAL(10, 2),

PaymentDate DATE,

FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)

);

CREATE TABLE RoomType (

TypeID INT PRIMARY KEY,

TypeName VARCHAR(50),

Description TEXT

);

**DESCRIPTION OF TABLES**

**1. Customers Table:**

* **CustomerID**: Unique identifier for each customer. It serves as the primary key of the table.
* **FirstName**: First name of the customer. It stores the customer's given name.
* **LastName**: Last name of the customer. It stores the customer's surname.
* **Email**: Email address of the customer. It stores the contact email for communication.
* **Phone**: Phone number of the customer. It stores the contact phone number.

**2. Room Table:**

* **RoomID**: Unique identifier for each room. It serves as the primary key of the table.
* **RoomNumber**: Number assigned to the room. It identifies the room within the hotel.
* **RoomType**: Type of the room (e.g., Single, Double, Suite). It categorizes rooms based on their accommodation features.
* **PricePerNight**: Price per night for the room. It indicates the cost of staying in the room for one night.

**3. Booking Table:**

* **BookingID**: Unique identifier for each booking. It serves as the primary key of the table.
* **CustomerID**: Foreign key referencing the CustomerID in the Customers table. It links the booking to the customer who made the reservation.
* **RoomID**: Foreign key referencing the RoomID in the Room table. It identifies the room booked for the reservation.
* **CheckInDate**: Date when the customer checks into the room. It specifies the start date of the booking.
* **CheckOutDate**: Date when the customer checks out of the room. It specifies the end date of the booking.

**4. Payment Table:**

* **PaymentID**: Unique identifier for each payment transaction. It serves as the primary key of the table.
* **BookingID**: Foreign key referencing the BookingID in the Booking table. It links the payment to the corresponding booking.
* **Amount**: Amount paid for the booking. It indicates the total payment amount for the reservation.
* **PaymentDate**: Date when the payment was made. It specifies the date and time of the payment transaction.

**5. RoomType Table:**

* **TypeID**: Unique identifier for each room type. It serves as the primary key of the table.
* **TypeName**: Name of the room type. It provides a descriptive label for each room category.
* **Description**: Description of the room type. It offers additional information about the features, amenities, and specifications of each room category.

**Normalization of tables up to 3-NF**

Normalization is a series of tests that individual relational schemas go through to

ensure that the database is normalized to avoid redundancy as much as possible. It

is a process of organizing data in database.

Normalization plays a crucial role in the designing of efficient(GOOD) database

schema. It is in three forms

**1) First Normal Form (1NF):**

It is the test of the normalization process.

It ensures that the there are no multi-valued Attributes.

To Qualify this test, all columns should be atomic in nature(i.e.., No groups or

arrays of attributes)

As we’ve seen before, there are no multi-valued attributes in the relations of our

Database Schema. So it passes 1NF test.

**2) Second Normal Form (2NF):**

It is the second test in the normalization process. It ensures that the relation has no

partial dependency.

Partial dependency is functional dependency condition in which, despite the

removal an attribute from Y, the FD Y->X holds. In other words, it is a condition in

which a non-prime attribute depends on some prime attributes only, instead of

depending on all prime attributes.

In our database schema, as all the relations have exactly one prime attribute, It

passed 2NF test too.

**3) Third Normal Form(3NF):**

It is the third test in the normalization process. It ensures that the relation has no

transitive dependency.

Transitive Dependency is a functional dependency condition in which any two

non-prime attributes involve in a functional dependency relationship.

In all the relations of the database schema, same kind of functional relationship can

be observed

i.e.., Y->X where Y is a single attribute which is the ID (primary key) and X is the set

of all other attributes.

So, this database schema passed 3NF test too.

With this, it marks the successful normalization process of the database schema.

**Normalization of Database**

Customers Table:

1NF: The table satisfies 1NF as it contains atomic values without repeating groups.

2NF: All non-key attributes (CustomerID, FirstName, LastName, Email, Phone) are fully functionally dependent on the entire primary key (CustomerID), so it satisfies 2NF.

3NF: There are no transitive dependencies; each non-key attribute depends only on the primary key (ID), so it satisfies 3NF.

Booking Table:

1NF: The table satisfies 1NF as it contains atomic values without repeating groups.

2NF: All non-key attributes (BookingID, CustomerID, RoomID, CheckInDate, CheckOutDate) are fully functionally dependent on the entire primary key (BookingID), so it satisfies 2NF.

3NF: There are no transitive dependencies; each non-key attribute depends only on the primary key (ID), so it satisfies 3NF.

Room Table:

1NF: The table satisfies 1NF as it contains atomic values without repeating groups.

2NF: All non-key attributes (RoomID, RoomNumber, RoomType, PricePerNight) are fully functionally dependent on the entire primary key (RoomID), so it satisfies 2NF.

3NF: There are no transitive dependencies; each non-key attribute depends only on the primary key (ID), so it satisfies 3NF.

RoomType Table:

1NF: The table satisfies 1NF as it contains atomic values without repeating groups.

2NF: All non-key attributes (TypeID, TypeName, Description) are fully functionally dependent on the entire primary key (TypeID), so it satisfies 2NF.

3NF: There are no transitive dependencies; each non-key attribute depends only on the primary key (ID), so it satisfies 3NF.

Payment Table:

1NF: The table satisfies 1NF as it contains atomic values without repeating groups.

2NF: All non-key attributes (PaymentID, BookingID, Amount, PaymentDate) are fully functionally dependent on the entire primary key (PaymentID), so it satisfies 2NF.

3NF: There are no transitive dependencies; each non-key attribute depends only on the primary key (ID), so it satisfies 3NF.

**Creation of Data in the tables**

-- Insert data into Customers table

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone)

VALUES

(1, 'John', 'Doe', 'john.doe@gmail.com ', '1234567890'),

(2, 'Jane', 'Smith', 'jane.smith@gmail.com', '9876543210'),

(3, 'Alice', 'Johnson', 'alice.johnson@gmail.com', '5551234567');

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CuatomerID | FirstName | LastName | Email | Phone |
| 1 | John | Doe | john.doe@gmail.com | 1234567890 |
| 2 | Jane | Smith | jane.smith@gmail.com | 9876543210 |
| 3 | Alice | Johnson | alice.johnson@gmail.com | 5551234567 |

-- Insert data into Room table

INSERT INTO Room (RoomID, RoomNumber, RoomType, PricePerNight)

VALUES

(101, '101', 'Single', 100.00),

(102, '102', 'Single', 100.00),

(201, '201', 'Double', 150.00),

(202, '202', 'Double', 150.00),

(301, '301', 'Suite', 250.00);

|  |  |  |  |
| --- | --- | --- | --- |
| RoomID | RoomNumber | RoomType | PricePerNight |
| 101 | 101 | Single | 100.00 |
| 102 | 102 | Single | 100.00 |
| 201 | 201 | Double | 150.00 |
| 201 | 202 | Double | 150.00 |
| 301 | 301 | Suite | 250.00 |

-- Insert data into Booking table

INSERT INTO Booking (BookingID, CustomerID, RoomID, CheckInDate, CheckOutDate)

VALUES

(1, 1, 101, '2024-05-01', '2024-05-03'),

(2, 2, 201, '2024-05-10', '2024-05-15'),

(3, 3, 301, '2024-06-01', '2024-06-05');

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BookingID | CustomerID | RoomID | CheckInDate | CheckOutDate |
| 1 | 1 | 101 | 2024-05-01 | 2024-05-03 |
| 2 | 2 | 201 | 2024-05-10 | 2024-05-15 |
| 3 | 3 | 301 | 2024-06-01 | 2024-06-05 |

-- Insert data into Payment table

INSERT INTO Payment (PaymentID, BookingID, Amount, PaymentDate)

VALUES

(1, 1, 200.00, '2024-05-01'),

(2, 2, 750.00, '2024-05-10'),

(3, 3, 1250.00, '2024-06-01');

|  |  |  |  |
| --- | --- | --- | --- |
| PaymentID | BookingID | Amount | PaymentDate |
| 1 | 1 | 200.00 | 2024-05-01 |
| 2 | 2 | 750.00 | 2024-05-10 |
| 3 | 3 | 1250.00 | 2024-06-01 |

-- Insert data into RoomType table

INSERT INTO RoomType (TypeID, TypeName, Description)

VALUES

(1, 'Single', 'A single occupancy roo m'),

(2, 'Double', 'A room with two beds'),

(3, 'Suite', 'A luxurious suite with additional amenities');

|  |  |  |
| --- | --- | --- |
| TypeID | TypeName | Description |
| 1 | Single | A single occupancy room |
| 2 | Double | A room with two beds |
| 3 | Suite | A luxurious suite with additional amenities |

**Few sql queries on the created tables**

1. Basic Select Queries:

- Retrieve all data from a specific table:

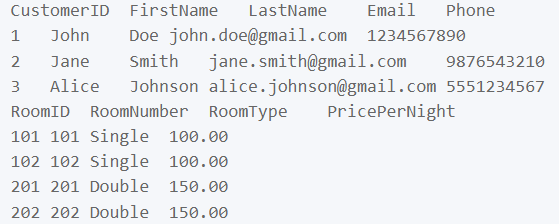
SELECT \* FROM Customers;

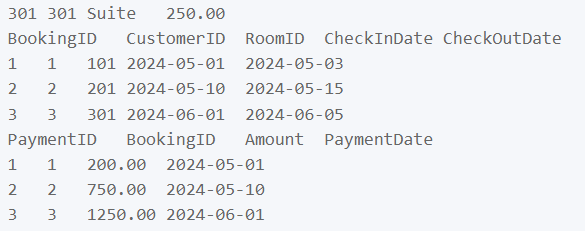
SELECT \* FROM Room;

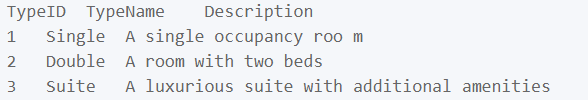
SELECT \* FROM Booking;

SELECT \* FROM Payment;

SELECT \* FROM RoomType;







2. Filtering Queries:

- Retrieve customers with a specific email:

SELECT \* FROM Customers WHERE Email = 'john.doe@gmail.com ';



- Retrieve rooms with a specific room type:

SELECT \* FROM Room WHERE RoomType = 'Suite';



- Retrieve bookings for a specific date range:

SELECT \* FROM Booking WHERE CheckInDate >= '2024-06-01' AND CheckOutDate <= '2024-06-05';



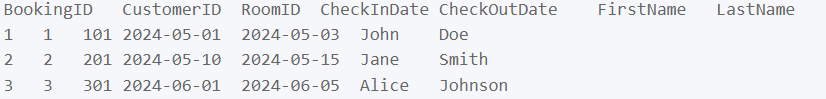
3. Join Queries:

- Retrieve bookings with customer information:

SELECT Booking.\*, Customers.FirstName, Customers.LastName

FROM Booking

INNER JOIN Customers ON Booking.CustomerID = Customers.CustomerID;



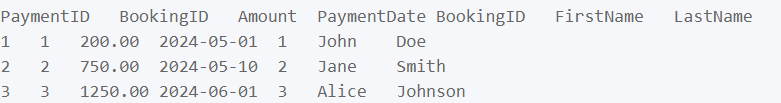
- Retrieve payments with booking and customer information:

SELECT Payment.\*, Booking.BookingID, Customers.FirstName, Customers.LastName

FROM Payment

INNER JOIN Booking ON Payment.BookingID = Booking.BookingID

INNER JOIN Customers ON Booking.CustomerID = Customers.CustomerID;



4. Aggregate Queries:

- Calculate the total number of bookings:

SELECT COUNT(\*) AS TotalBookings FROM Booking;



- Calculate the average price per night of rooms:

SELECT AVG(PricePerNight) AS AveragePrice FROM Room;



- Calculate the total amount paid for all bookings:

SELECT SUM(Amount) AS TotalAmountPaid FROM Payment;



5. Subqueries:

- Retrieve bookings made by customers with a specific email:

SELECT \* FROM Booking WHERE CustomerID IN (SELECT CustomerID FROM Customers WHERE Email = 'example@example.com');



6. Sorting Queries:

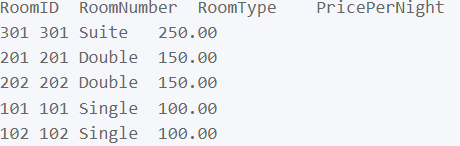
- Retrieve customers sorted by last name:

SELECT \* FROM Customers ORDER BY LastName;



- Retrieve rooms sorted by price per night in descending order:

SELECT \* FROM Room ORDER BY PricePerNight DESC;

****

**Creation of 5 views using the tables**

Certainly! Here are five views that you can create based on the tables in the database:

1. CustomerBookingsView:

This view shows the details of customers along with their booking information.

CREATE VIEW CustomerBookingsView AS

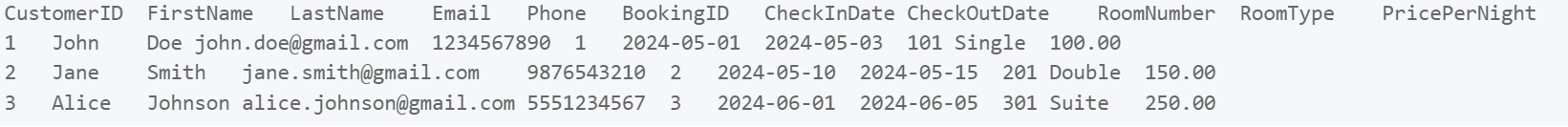
SELECT c.CustomerID, c.FirstName, c.LastName, c.Email, c.Phone,

b.BookingID, b.CheckInDate, b.CheckOutDate, r.RoomNumber, r.RoomType, r.PricePerNight

FROM Customers c

JOIN Booking b ON c.CustomerID = b.CustomerID

JOIN Room r ON b.RoomID = r.RoomID;



2. AvailableRoomsView: This view displays available rooms for a given date range and room type.

CREATE VIEW AvailableRoomsView AS

SELECT r.RoomID, r.RoomNumber, r.RoomType, r.PricePerNight

FROM Room r

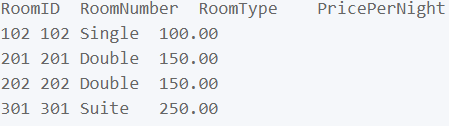
WHERE r.RoomID NOT IN (

SELECT b.RoomID

FROM Booking b

WHERE '2024-05-01' BETWEEN b.CheckInDate AND b.CheckOutDate

);



3. RevenueByMonthView: This view summarizes the revenue generated by month.

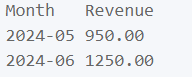
CREATE VIEW RevenueByMonthView AS

SELECT DATE\_FORMAT(b.CheckInDate, '%Y-%m') AS Month, SUM(p.Amount) AS Revenue

FROM Booking b

JOIN Payment p ON b.BookingID = p.BookingID

GROUP BY Month;



4. RoomTypeDetailsView: This view provides details about each room type.

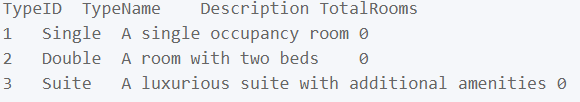
CREATE VIEW RoomTypeDetailsView AS

SELECT rt.TypeID, rt.TypeName, rt.Description, COUNT(r.RoomID) AS TotalRooms

FROM RoomType rt

LEFT JOIN Room r ON rt.TypeID = r.RoomType

GROUP BY rt.TypeID;



5. PaymentDetailsView: This view shows payment details along with booking and customer information.

CREATE VIEW PaymentDetailsView AS

SELECT p.PaymentID, p.Amount, p.PaymentDate,

b.BookingID, b.CheckInDate, b.CheckOutDate,

c.CustomerID, c.FirstName, c.LastName, c.Email, c.Phone

FROM Payment p

JOIN Booking b ON p.BookingID = b.BookingID

JOIN Customers c ON b.CustomerID = c.CustomerID;

