

# A Course-based Project Report On

**Hotel Management System**

## Submitted in partial fulfillment of the requirement for the completion of the

**Database Management Systems Laboratory course**

# B.Tech Computer Science and Engineering

## of VNRVJIET

**By**

|  |  |
| --- | --- |
| **1. Arja.Harshitha** | **- 23071A05N0** |
| **2. K Sruthi Priya** | **- 23071A05Q3** |
| **3. Kondeti Suchita** | **- 23071A05Q7** |
|  |  |

## VALLURPALLI NAGESWARARAO

**VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS INSTITUTE)**

## NAAC ACCREDITED WITH ‘A++ GRADE

**VignanaJyothi Nagar, Bachupally, Nizampet (s.o), Hyderabad 500090 Phone no: 040-23042758/59/60, Fax: 040-23042761**

## [Email:postbox@vnrvjiet.ac.in](mailto:postbox@vnrvjiet.ac.in) Website: [www.vnrvjiet.ac.in](http://www.vnrvjiet.ac.in/) 2024-2025

1

# A Project Report On

**Hotel Management System**

## Submitted in partial fulfillment of requirement for the completion of the

**Database Management Systems Laboratory course.**

# B.Tech Computer Science and Engineering

## Under the Guidance of

**Dr. M. Madhu Bala Professor**

**CSE Department**



## VALLURPALLI NAGESWARARAO

**VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS INSTITUTE)**

## NAAC ACCREDITED WITH ‘A++ GRADE

**VignanaJyothi Nagar, Bachupally, Nizampet (s.o), Hyderabad 500090 Phone no: 040-23042758/59/60, Fax: 040-23042761**

## [Email:postbox@vnrvjiet.ac.in](mailto:postbox@vnrvjiet.ac.in) Website: [www.vnrvjiet.ac.in](http://www.vnrvjiet.ac.in/) 2024-2025

2



# VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

**(AUTONOMOUS INSTITUTE) NAAC ACCREDITED WITH ‘A++’ GRADE**

# CERTIFICATE

This is to certify that the project entitled **“Hotel Booking Management System”**

submitted in partial fulfillment for the course of Database Management System being

offered for the award of B.Tech (CSE) by VNR VJIET is a result of the bonafide carried out by A.Harshitha (23071A05N0) , K.SruthiPriya (23071A05Q3) ,

K.Suchita (23071A05Q7) during the year **2024-2025**. This has not been submitted for any other certificate or course.

## Signature of Faculty Signature of Head of the Department

**ACKNOWLEDGEMENT**

An endeavor over a long period can be successful only with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them.

We wish to express our profound gratitude to our honorable **Principal and HOD, CSE** department**, VNR Vignana Jyothi Institute of Engineering and Technology** for their constant and dedicated support towards our career molding and development.

With great pleasure, we express our gratitude to the internal guide **Dr. M. Madhu Bala, Professor, CSE department** for her timely help, constant guidance, cooperation, support, and encouragement throughout this project as it has urged us to explore many new things.

Finally, we wish to express my deep sense of gratitude and sincere thanks to our parents, friends and all our well-wishers who have technically and non-technically contributed for the successful completion of this course-based project.

# DECLARATION

We hereby declare that this Project Report titled **“Hotel Management System”** submitted by us of Computer Science & Engineering in **VNR Vignana Jyothi Institute of Engineering and Technology** is a bonafide work undertaken by us and it is not submitted for any other certificate /Course or published any time before.

Signature of the Student Date:

23071A05N0 ARJA HARSHITHA

23071A05Q3 KANURI SRUTHI PRIYA

23071A05Q7 KONDETI SUCHITA

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **PAGE** |
|  | Abstract | 7 |
| 1 | Introduction | 8-9 |
| 2 | ER Diagram | 10 |
| 3 | Database Schema | 11-12 |
| 4 | Implementation and Results | 13-25 |
| 5 | References | 26 |

# ABSTRACT

The Hotel Management System is a database-driven application developed to streamline operations within the hospitality industry by managing reservations, guest services, payments, staff assignments, and room availability. This system enables hotel staff to efficiently handle bookings, track services availed by guests, manage payments, and maintain guest information. It supports multiple user roles such as administrators, front desk staff, and service managers, providing secure access tailored to specific job functions.

The system is structured around six primary database tables: Bookings, Services, Payments, Staff, Guests, and Rooms. These tables establish essential relationships, allowing smooth and efficient data retrieval and updates across various hotel functions. Key features include real-time room availability updates, booking confirmations and cancellations, service management for guests, payment processing, and staff scheduling. SQL queries enable crucial operations, such as generating booking histories, tracking service usage, and producing financial reports, making data readily accessible and actionable for management.

With its modular design, the Hotel Management System is scalable and can be adapted to meet the needs of hotels of various sizes, reducing manual effort, improving accuracy, and enhancing the overall guest experience. This system is positioned as a comprehensive tool that can help hotels optimize resource allocation, improve operational efficiency, and maintain a high standard of service.

# INTRODUCTION

In the dynamic and competitive hospitality industry, efficient management of hotel bookings and guest information is essential for delivering high-quality and personalized service. As hotels face increasing volumes of booking data and guest preferences, the need for a robust and streamlined system for organizing, storing, and retrieving this information becomes critical. The Hotel Booking Management System (HBMS) emerges as a comprehensive solution to address the complexities and challenges associated with managing bookings, services, payments, and room availability.

The HBMS represents a shift from traditional paper-based record-keeping to a digital, integrated, and secure approach. Its implementation signals a new era in hotel administration, promising enhanced accuracy, accessibility, and security of guest and booking information. By leveraging advanced technologies, the HBMS improves the efficiency of hotel staff, optimizes resource allocation, and enhances guest satisfaction through faster and more accurate service. Overall, the HBMS contributes to better operational efficiency and helps hotels deliver an elevated experience to their guests, positioning the system as an indispensable tool in the modern hospitality landscape.

## APPLICATIONS

A Hotel Management System (HMBS) implemented within a Database Management System (DBMS) provides several applications that streamline various hotel operations and enhance overall guest experience. Below are key applications of a hotel management booking system within a DBMS:

1. Reservation Management

Online Bookings: Guests can reserve rooms through a website or mobile application, with the system updating availability in real-time.

Cancellation and Modifications: The system allows guests to cancel or modify their reservations, automatically updating the database and freeing up rooms for other guests.

2. Room Management

Real-Time Availability: The system tracks the availability of rooms, displaying current statuses (available, booked, under maintenance) to staff and guests.

Room Assignment: The database can facilitate room assignments based on guest preferences and availability, optimizing occupancy rates.

3. Guest Management

Profile Creation: Guests can create profiles that store personal information, preferences, and past stays, allowing for personalized services.

Check-In and Check-Out Processes: The system streamlines check-in and check-out procedures, reducing wait times and improving guest satisfaction.

4. Billing and Payment Processing

Invoice Generation: The system automatically generates invoices upon check-out, including room charges and additional services used.

Payment Tracking: It manages various payment methods (credit card, cash, online payment) and records transactions against bookings for accurate financial tracking.

5. Service Management

Additional Services: The system allows guests to book additional services such as spa treatments, restaurant reservations, and room service, integrating these with their bookings.

Service Billing: All additional services opted by guests are recorded and included in their final bill, ensuring accurate charges.

QUERIES WE WANT TO ADDRESS

**Query-1:**

**How can we fetch the first names, last names, and booking details for all guests who have  bookings?**

**Query-2:**

**How can we fetch room numbers and their corresponding payment amounts for all  bookings?**

**Query-3:**

**Query to find guests who have stayed in the most expensive room type.**

**Query-4:**

**Query to find** **Average Room Price by Type.**

**Query-5:**

**List the query to find the people who have booked between the 01-01-2024 to 31-12-2024.**

**Query-6:**

**Query to find the service that has been opted for by the most guests.**

**Query-7:**

**Query to get all room numbers that are either booked or currently occupied.**

**Query-8:**

**Give the names of guests who have not chosen any payment methods.**

**Query-9:**

**What is the occupancy rate of each room type?**

**Query-10:**

**List the guests who have booked the most recently**

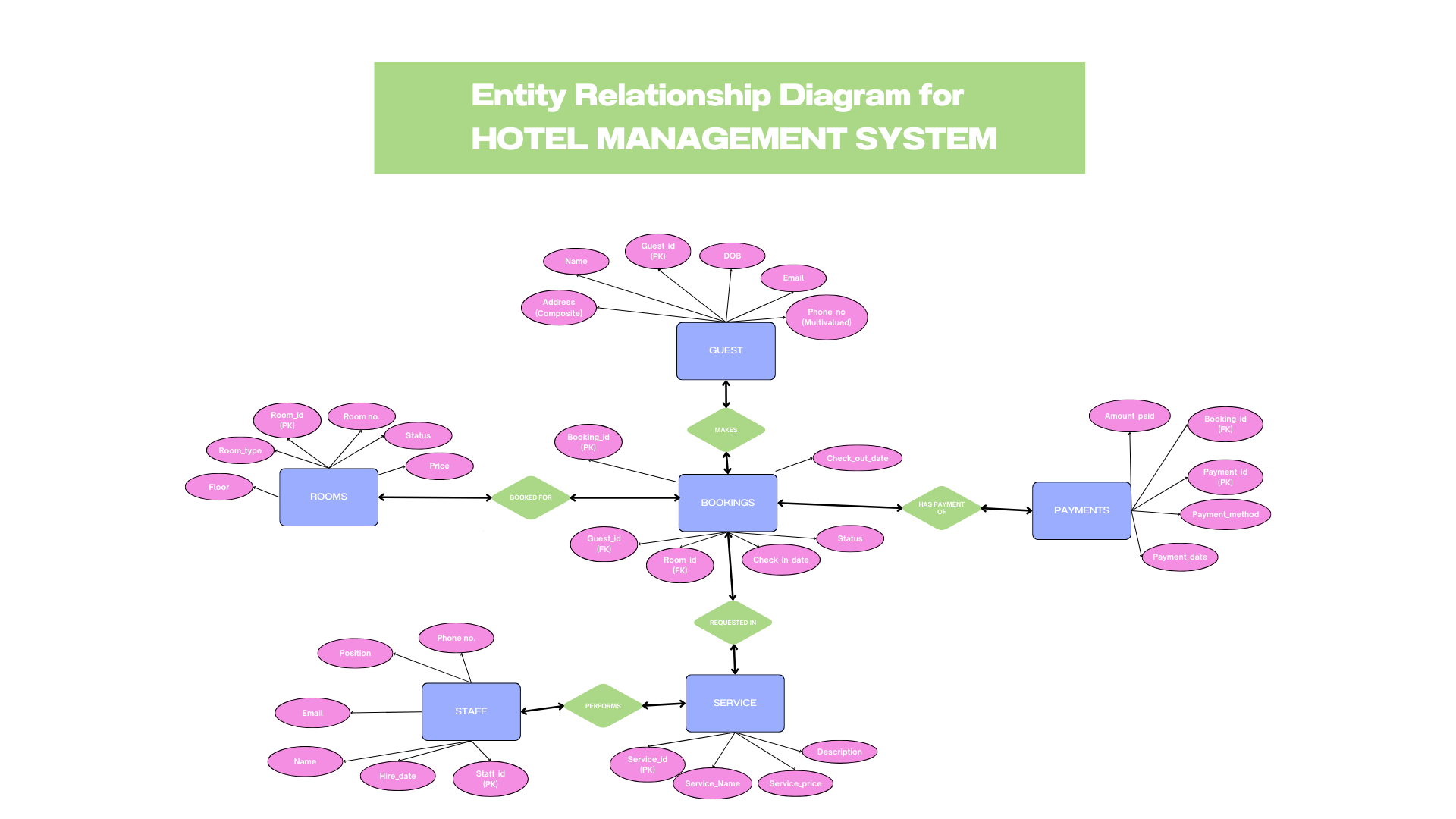
**Query-11:**

**Create a trigger that logs each new booking into the BookingLogs table**

# Tools used :

We used MySQl to create database and execute our queries.

**ER DIAGRAM**

****

Description of ER Diagram

ENTITIES:

Represented as : 

Entities used in the ER Diagram:

* Guests
* Rooms
* Bookings
* Payments
* Staff
* Services

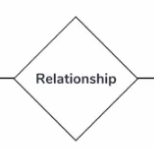
ATTRIBUTES:

Represented by: 

Attributes used in the ER Diagram:

* Guests: (Guest\_id , Name, Phone Number, Email, Address, DOB)
* Rooms: (Room\_id, Room No., Room\_type, Status, Price, floor)
* Bookings: (Bookings\_id, Guest\_id, Room\_id,Check\_in\_Date,Check\_out\_Date, Status)
* Payments: (Payment\_id, Booking\_id, Amount\_Paid, Amount\_date,Payment\_method)
* Staff : (Staff\_id, Name,Position, Phone No., Email, Hire\_Date)
* Service: (Service\_id, Service\_Name, Service\_Price, Description)

RELATIONS:

Represented by: 

Relations used in the ER Diagram:

**Guests** ↔ **Bookings**

* **Relationship Name**: *Makes*
* **Type**: One-to-Many
* **Description**: A guest can make multiple bookings, but each booking is associated with one guest.

**Rooms** ↔ **Bookings**

* **Relationship Name**: *Assigned to*
* **Type**: One-to-Many
* **Description**: A room can be booked multiple times (over different periods), but each booking refers to a single room.

**Bookings** ↔ **Payments**

* **Relationship Name**: *Has Payment*
* **Type**: One-to-One or One-to-Many
* **Description**: A booking can have one or more associated payments (e.g., installment payments).

**Bookings** ↔ **Booking Details**

* **Relationship Name**: *Includes Details*
* **Type**: One-to-Many (weak relationship)
* **Description**: A booking can include multiple details (e.g., additional services booked). **Booking Details** is a weak entity that depends on **Bookings**.

**Services** ↔ **Booking Details**

* **Relationship Name**: *Provides*
* **Type**: One-to-Many
* **Description**: A service can appear in multiple booking details, but each booking detail refers to only one service.

**Staff** ↔ **Services**

* **Relationship Name**: *Performs*
* **Type**: One-to-Many
* **Description**: A staff member can perform multiple services (e.g., room cleaning, spa treatment), but each service instance is performed by one staff member.

# DATABASE SCHEMA

# Guests table:

---------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------------+--------------+------+-----+---------+-------+

| guest\_id | int | NO | PRI | NULL | |

| first\_name | varchar(50) | YES | | NULL | |

| last\_name | varchar(50) | YES | | NULL | |

| phone\_number | varchar(15) | YES | | NULL | |

| email | varchar(100) | YES | | NULL | |

| address | varchar(255) | YES | | NULL | |

| date\_of\_birth | varchar(15) | YES | | NULL | |

+---------------+--------------+------+-----+---------+-------+

# Rooms table:

+-----------------+---------------+------+-----+-----------+-------+

| Field | Type | Null | Key | Default | Extra |

+-----------------+---------------+------+-----+-----------+-------+

| room\_id | int | NO | PRI | NULL | |

| room\_number | varchar(10) | YES | UNI | NULL | |

| room\_type | varchar(50) | YES | | NULL | |

| status | varchar(20) | YES | | Available | |

| price\_per\_night | decimal(10,2) | YES | | NULL | |

| floor | int | YES | | NULL | |

+-----------------+---------------+------+-----+-----------+-------+

# Bookings table

+----------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+-------+

| booking\_id | int | NO | PRI | NULL | |

| guest\_id | int | YES | MUL | NULL | |

| room\_id | int | YES | MUL | NULL | |

| check\_in\_date | date | YES | | NULL | |

| check\_out\_date | date | YES | | NULL | |

| status | varchar(20) | YES | | Booked | |

+----------------+-------------+------+-----+---------+-------+

# Payments table

+----------------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+---------------+------+-----+---------+-------+

| payment\_id | int | NO | PRI | NULL | |

| booking\_id | int | YES | MUL | NULL | |

| amount\_paid | decimal(10,2) | YES | | NULL | |

| payment\_date | date | YES | | NULL | |

| payment\_method | varchar(50) | YES | | NULL | |

+----------------+---------------+------+-----+---------+-------+

# Staff table

+--------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------------+--------------+------+-----+---------+-------+

| staff\_id | int | NO | PRI | NULL | |

| first\_name | varchar(50) | YES | | NULL | |

| last\_name | varchar(50) | YES | | NULL | |

| position | varchar(50) | YES | | NULL | |

| phone\_number | varchar(15) | YES | | NULL | |

| email | varchar(100) | YES | | NULL | |

| hire\_date | date | YES | | NULL | |

+--------------+--------------+------+-----+---------+-------+

# Services table

+---------------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------------+---------------+------+-----+---------+-------+

| service\_id | int | NO | PRI | NULL | |

| service\_name | varchar(100) | YES | | NULL | |

| service\_price | decimal(10,2) | YES | | NULL | |

| description | varchar(255) | YES | | NULL | |

+---------------+---------------+------+-----+---------+-------+

# BookingLogs table

# IMPLEMENTATION AND RESULTS

Guests table

CREATE TABLE Guests (

guest\_id INT PRIMARY KEY ,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

phone\_number VARCHAR(15),

email VARCHAR(100),

address VARCHAR(255),

date\_of\_birth VARCHAR(15)

);

--inserting values into Guests table

INSERT INTO Guests (GuestID, first\_name, last\_name, phone\_number, email, address, date\_of\_birth) VALUES

(1, 'John', 'Doe', '1234567890', 'john.doe@example.com', '123 Main St', '1990-01-01'),

(2, 'Jane', 'Smith', '2345678901', 'jane.smith@example.com', '456 Elm St', '1985-02-02'),

(3, 'Alice', 'Johnson', '3456789012', 'alice.johnson@example.com', '789 Maple St', '1992-03-03'),

(4, 'Bob', 'Brown', '4567890123', 'bob.brown@example.com', '101 Pine St', '1980-04-04'),

(5, 'Charlie', 'Davis', '5678901234', 'charlie.davis@example.com', '202 Oak St', '1988-05-05'),

(6, 'Emily', 'Clark', '6789012345', 'emily.clark@example.com', '303 Cedar St', '1975-06-06'),

(7, 'David', 'Taylor', '7890123456', 'david.taylor@example.com', '404 Birch St', '1982-07-07'),

(8, 'Sarah', 'Wilson', '8901234567', 'sarah.wilson@example.com', '505 Spruce St', '1995-08-08'),

(9, 'Michael', 'Moore', '9012345678', 'michael.moore@example.com', '606 Fir St', '1978-09-09'),

(10, 'Laura', 'Martin', '0123456789', 'laura.martin@example.com', '707 Willow St', '1981-10-10'),

(11, 'Paul', 'Allen', '1234509876', 'paul.allen@example.com', '808 Ash St', '1993-11-11'),

(12, 'Nancy', 'Roberts', '2345610987', 'nancy.roberts@example.com', '909 Palm St', '1986-12-12'),

(13, 'Steven', 'White', '3456721098', 'steven.white@example.com', '1010 Oakwood St', '1997-03-13'),

(14, 'Betty', 'Harris', '4567832109', 'betty.harris@example.com', '1111 Elmwood St', '1991-04-14'),

(15, 'Christopher', 'Jackson', '5678943210', 'christopher.jackson@example.com', '1212 Maplewood St', '1984-05-15');

select \*from Guests;

output:



-- Rooms table

CREATE TABLE Rooms (

room\_id INT PRIMARY KEY ,

room\_number VARCHAR(10) UNIQUE,

room\_type VARCHAR(50),

status VARCHAR(20) DEFAULT 'Available',

price\_per\_night DECIMAL(10, 2),

floor number(1)

);

-- Insert values into Rooms table

INSERT INTO Rooms (RoomID, room\_number, room\_type, status, price\_per\_night, floor) VALUES

(1, '101', 'Single', 'Available', 100.00, 1),

(2, '102', 'Double', 'Occupied', 150.00, 1),

(3, '103', 'Suite', 'Available', 300.00, 1),

(4, '104', 'Single', 'Available', 100.00, 1),

(5, '105', 'Double', 'Available', 150.00, 1),

(6, '201', 'Single', 'Available', 100.00, 2),

(7, '202', 'Double', 'Occupied', 150.00, 2),

(8, '203', 'Suite', 'Available', 300.00, 2),

(9, '204', 'Single', 'Available', 100.00, 2),

(10, '205', 'Double', 'Available', 150.00, 2),

(11, '301', 'Single', 'Available', 100.00, 3),

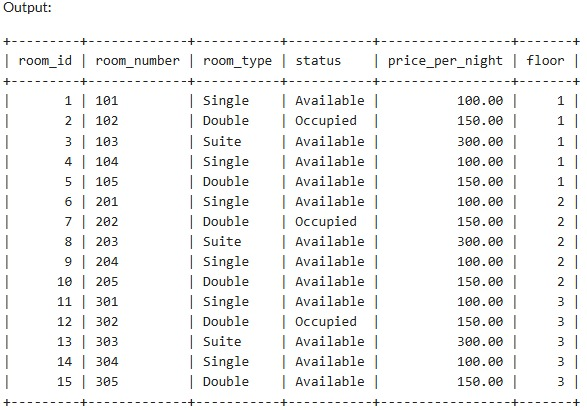
(12, '302', 'Double', 'Occupied', 150.00, 3),

(13, '303', 'Suite', 'Available', 300.00, 3),

(14, '304', 'Single', 'Available', 100.00, 3),

(15, '305', 'Double', 'Available', 150.00, 3);

select \*from Rooms;



-- Bookings table

CREATE TABLE Bookings (

booking\_id INT PRIMARY KEY ,

guest\_id INT,

room\_id INT,

check\_in\_date DATE,

check\_out\_date DATE,

status VARCHAR(20) DEFAULT 'Booked',

FOREIGN KEY (guest\_id) REFERENCES Guests(guest\_id),

FOREIGN KEY (room\_id) REFERENCES Rooms(room\_id)

);

-- Insert values into the Bookings table

INSERT INTO Bookings (booking\_id, guest\_id, room\_id, check\_in\_date, check\_out\_date, status) VALUES

(1, 1, 1, '2024-01-01', '2024-01-07', 'Booked'),

(2, 2, 2, '2024-02-01', '2024-02-05', 'Booked'),

(3, 3, 3, '2024-03-01', '2024-03-10', 'Checked Out'),

(4, 4, 4, '2024-04-01', '2024-04-07', 'Booked'),

(5, 5, 5, '2024-05-01', '2024-05-07', 'Cancelled'),

(6, 6, 6, '2024-06-01', '2024-06-07', 'Booked'),

(7, 7, 7, '2024-07-01', '2024-07-05', 'Checked Out'),

(8, 8, 8, '2024-08-01', '2024-08-10', 'Booked'),

(9, 9, 9, '2024-09-01', '2024-09-07', 'Cancelled'),

(10, 10, 10, '2024-10-01', '2024-10-05', 'Booked'),

(11, 11, 11, '2024-11-01', '2024-11-07', 'Booked'),

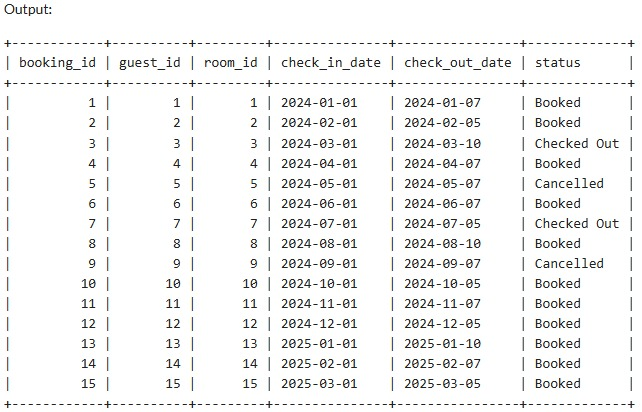
(12, 12, 12, '2024-12-01', '2024-12-05', 'Booked'),

(13, 13, 13, '2025-01-01', '2025-01-10', 'Booked'),

(14, 14, 14, '2025-02-01', '2025-02-07', 'Booked'),

(15, 15, 15, '2025-03-01', '2025-03-05', 'Booked');

Select \*from Bookings;



-- Payments table

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY ,

booking\_id INT,

amount\_paid DECIMAL(10, 2),

payment\_date DATE,

payment\_method VARCHAR(50),

FOREIGN KEY (booking\_id) REFERENCES Bookings(booking\_id)

);

-- Insert values into the Payments table

INSERT INTO Payments (payment\_id, booking\_id, amount\_paid, payment\_date, payment\_method) VALUES

(1, 1, 600.00, '2024-01-07', 'Credit Card'),

(2, 2, 750.00, '2024-02-05', 'Debit Card'),

(3, 3, 900.00, '2024-03-10', 'Cash'),

(4, 4, 700.00, '2024-04-07', 'Credit Card'),

(5, 5, 0.00, '2024-05-07', 'N/A'),

(6, 6, 600.00, '2024-06-07', 'Credit Card'),

(7, 7, 750.00, '2024-07-05', 'Debit Card'),

(8, 8, 900.00, '2024-08-10', 'Cash'),

(9, 9, 0.00, '2024-09-07', 'N/A'),

(10, 10, 750.00, '2024-10-05', 'Credit Card'),

(11, 11, 700.00, '2024-11-07', 'Credit Card'),

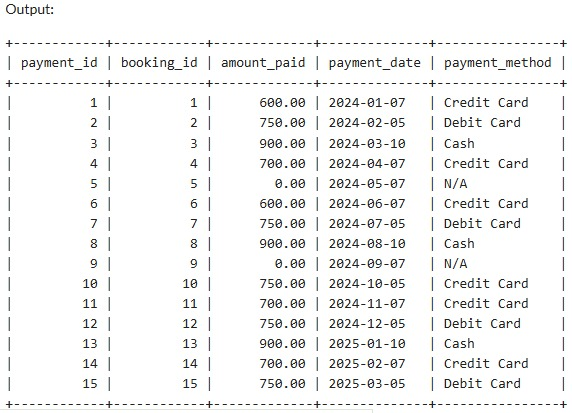
(12, 12, 750.00, '2024-12-05', 'Debit Card'),

(13, 13, 900.00, '2025-01-10', 'Cash'),

(14, 14, 700.00, '2025-02-07', 'Credit Card'),

(15, 15, 750.00, '2025-03-05', 'Debit Card');

select \*from Payments;



--Staff table

CREATE TABLE Staff (

staff\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

position VARCHAR(50),

phone\_number VARCHAR(15),

email VARCHAR(100),

hire\_date DATE

);

-- Insert values into the Staff table

INSERT INTO Staff (staff\_id, first\_name, last\_name, position, phone\_number, email, hire\_date) VALUES

(1, 'Emily', 'Clark', 'Manager', '6789012345', 'emily.clark@example.com', '2015-06-01'),

(2, 'David', 'Taylor', 'Receptionist', '7890123456', 'david.taylor@example.com', '2017-07-01'),

(3, 'Sarah', 'Wilson', 'Housekeeping', '8901234567', 'sarah.wilson@example.com', '2018-08-01'),

(4, 'Michael', 'Moore', 'Chef', '9012345678', 'michael.moore@example.com', '2016-09-01'),

(5, 'Laura', 'Martin', 'Security', '0123456789', 'laura.martin@example.com', '2019-10-01'),

(6, 'Mark', 'Lee', 'Receptionist', '1234098765', 'mark.lee@example.com', '2020-01-11'),

(7, 'Nina', 'Brown', 'Housekeeping', '2345109876', 'nina.brown@example.com', '2016-02-21'),

(8, 'Oliver', 'Jones', 'Manager', '3456210987', 'oliver.jones@example.com', '2017-03-15'),

(9, 'Peter', 'Miller', 'Chef', '4567321098', 'peter.miller@example.com', '2019-04-10'),

(10, 'Rachel', 'Davis', 'Security', '5678432109', 'rachel.davis@example.com', '2018-05-09'),

(11, 'Sam', 'Wilson', 'Receptionist', '6789543210', 'sam.wilson@example.com', '2021-06-08'),

(12, 'Tina', 'White', 'Housekeeping', '7890654321', 'tina.white@example.com', '2022-07-07'),

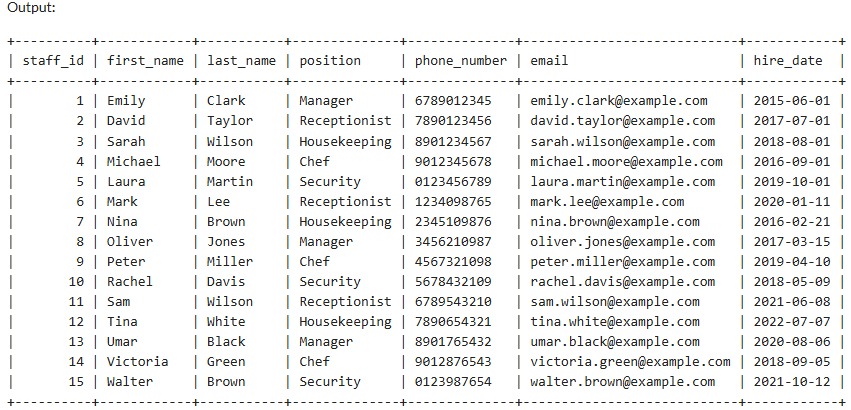
(13, 'Umar', 'Black', 'Manager', '8901765432', 'umar.black@example.com', '2020-08-06'),

(14, 'Victoria', 'Green', 'Chef', '9012876543', 'victoria.green@example.com', '2018-09-05'),

(15, 'Walter', 'Brown', 'Security', '0123987654', 'walter.brown@example.com', '2021-10-12');

select \*from Staff;

Output:



--Services table

CREATE TABLE Services (

service\_id INT PRIMARY KEY ,

service\_name VARCHAR(100),

service\_price DECIMAL(10, 2),

description VARCHAR(255)

);

-- Insert values into the Services table

INSERT INTO Services (service\_id, service\_name, service\_price, description) VALUES

(1, 'Room Service', 50.00, 'In-room dining service'),

(2, 'Laundry', 20.00, 'Laundry and dry cleaning service'),

(3, 'Spa', 100.00, 'Relaxation and therapy treatments'),

(4, 'Gym', 0.00, 'Access to fitness center'),

(5, 'WiFi', 0.00, 'Complimentary internet access'),

(6, 'Breakfast', 15.00, 'Daily breakfast service'),

(7, 'Parking', 10.00, 'Valet parking service'),

(8, 'Airport Shuttle', 30.00, 'Transportation to and from the airport'),

(9, 'Concierge', 0.00, 'Assistance with bookings and recommendations'),

(10, 'Pet Care', 25.00, 'Pet sitting and walking services'),

(11, 'Business Center', 0.00, 'Access to business facilities'),

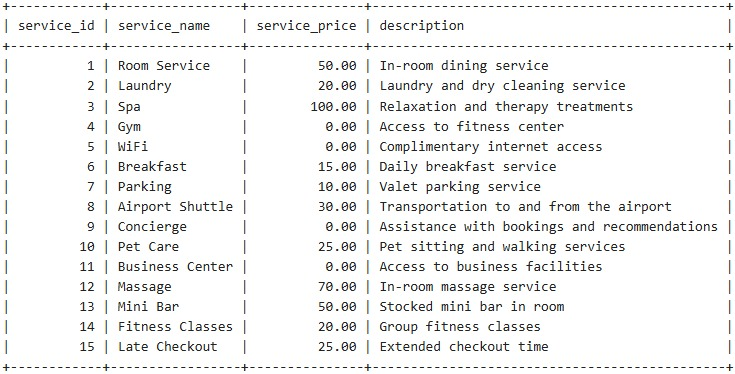
(12, 'Massage', 70.00, 'In-room massage service'),

(13, 'Mini Bar', 50.00, 'Stocked mini bar in room'),

(14, 'Fitness Classes', 20.00, 'Group fitness classes'),

(15, 'Late Checkout', 25.00, 'Extended checkout time');

Select \* from Services;



--BookingLogs table

CREATE TABLE BookingLogs (

log\_id NUMBER PRIMARY KEY,

booking\_id INT,

guest\_id INT,

room\_id INT,

log\_message VARCHAR2(255),

log\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Functional Dependencies and Normalization**

FUNCTIONAL DEPENDENCIES:

**1.Guests Table**

guest\_id → first\_name, last\_name, phone\_number, email, address, date\_of\_birth

(The guest\_id uniquely determines all other guest information.)

**2. Rooms Table**

room\_id → room\_number, room\_type, status, price\_per\_night, floor

(The room\_id uniquely determines all other room details.)

room\_number → room\_id, room\_type, status, price\_per\_night, floor

(Assuming room numbers are unique, room\_number can also determine all other room details.)

**3. Bookings Table**

booking\_id → guest\_id, room\_id, check\_in\_date, check\_out\_date, status

(The booking\_id uniquely determines all other booking information.)

(guest\_id, check\_in\_date) → booking\_id, room\_id, check\_out\_date, status

(A guest and check-in date combination can uniquely determine a booking.)

(room\_id, check\_in\_date) → booking\_id, guest\_id, check\_out\_date, status

(A room and check-in date combination can uniquely determine a booking.)

**4. Payments Table**

payment\_id → booking\_id, amount\_paid, payment\_date, payment\_method

(The payment\_id uniquely determines all other payment information.)

booking\_id → payment\_id, amount\_paid, payment\_date, payment\_method

(Assuming each booking can have only one payment, booking\_id can determine payment details.)

**5. Staff Table**

staff\_id → first\_name, last\_name, position, phone\_number, email, hire\_date

(The staff\_id uniquely determines all other staff information.)

**6. Services Table**

service\_id → service\_name, service\_price, description

(The service\_id uniquely determines all other service information.)

service\_name → service\_id, service\_price, description

(If service\_name is unique, it can also determine the other attributes.)

NORMALISATIONS:

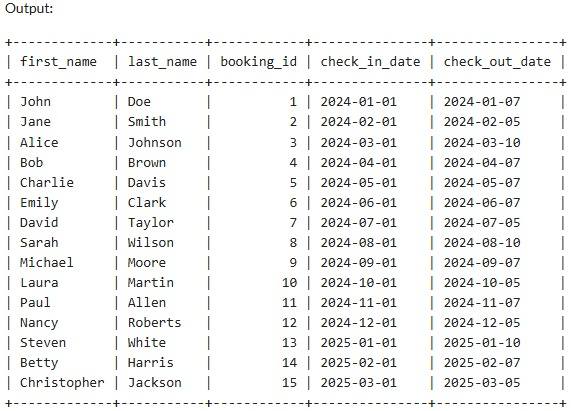
* The Guests table is normalized up to the Third Normal Form (3NF).
* The Rooms table is normalized up to the Third Normal Form (3NF).
* The Bookings table is normalized up to the Third Normal Form (3NF).
* The Payments table is normalized up to the Third Normal Form (3NF).
* The Staff table is normalized up to the Third Normal Form (3NF).
* The Services table is normalized up to Third Normal Form(3NF).
* The BookingLogs table is normalized up to the Third Normal Form (3NF).

**Query 1: How can we fetch the first names, last names, and booking details for all guests who have  bookings?**

SELECT Guests.first\_name, Guests.last\_name, Bookings.booking\_id, Bookings.check\_in\_date, Bookings.check\_out\_date

FROM Guests

JOIN Bookings ON Guests.Guest\_ID = Bookings.guest\_id;



**Description:**

This query retrieves the first names and last names of guests along with their booking IDs, check-in dates, and checkout dates. The JOIN operation is used to combine rows from the **Guests** and  **Bookings** tables based on a related column between them, which in this case is GuestID.

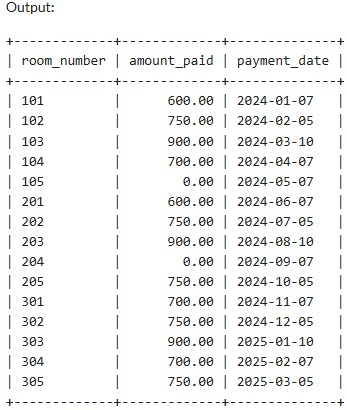
**Query-2:** **How can we fetch room numbers and their corresponding payment amounts for all  bookings?**

SELECT Rooms.room\_number, Payments.amount\_paid, Payments.payment\_date

FROM Rooms

JOIN Bookings ON Rooms.Room\_ID = Bookings.room\_id

JOIN Payments ON Bookings.booking\_id = Payments.booking\_id;



**Description:**

This query retrieves the room numbers along with the corresponding payment amounts and payment dates for all bookings. It uses two JOIN operations: the first joins the **Rooms** and  **Bookings** tables on RoomID, and the second joins the **Bookings** and **Payments** tables on booking\_id.

**Query-3:** **Query to find guests who have stayed in the most expensive room type**

SELECT g.first\_name, g.last\_name

FROM Guests g

WHERE g.guest\_id IN (

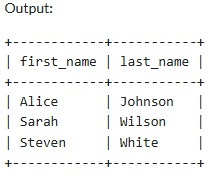
SELECT b.guest\_id

FROM Bookings b

JOIN Rooms r ON b.room\_id = r.room\_id

WHERE r.price\_per\_night = (SELECT MAX(price\_per\_night) FROM Rooms)

);



**Description:**

Outer Query:

Selects the first\_name and last\_name of guests from the Guests table where the guest\_id matches the result of the subquery.

Subquery:

Retrieves guest\_id from the Bookings table by joining with the Rooms table to find rooms that have the highest price\_per\_night.

Nested Subquery:

(SELECT MAX(price\_per\_night) FROM Rooms) finds the maximum room price from the Rooms table.

Result:

The outer query returns the names of guests who have stayed in rooms with the maximum price per night.

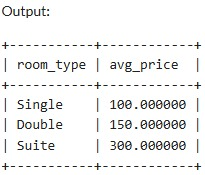
Use Case: This type of subquery is useful for identifying specific records based on a calculated value, like finding guests who have experienced the highest-priced accommodation.

**Query 4:-** **Query to find** **Average Room Price by Type**

SELECT room\_type, AVG(price\_per\_night) AS avg\_price

FROM Rooms

GROUP BY room\_type;

****

**Description:**

**SELECT room\_type, AVG(price\_per\_night) AS avg\_price**: This part of the query specifies that we want to select the room type and calculate the average price per night for each room type.

**FROM Rooms**: Indicates that we're pulling data from the **Rooms** table.

**GROUP BY room\_type**: Groups the results by room type so that the average price is calculated

for each type.

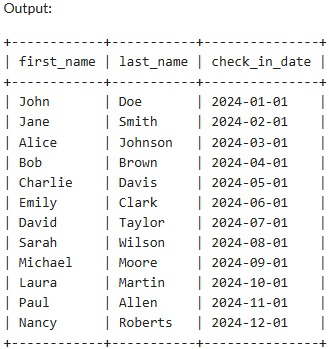
**Query 5:-** **List the query to find the people who have booked between the 01-01-2024 to 31-12-2024**

SELECT g.first\_name, g.last\_name, b.check\_in\_date

FROM Guests g

JOIN Bookings b ON g.guest\_id = b.guest\_id

WHERE b.check\_in\_date BETWEEN '2024-01-01' AND '2024-12-31';



**Description:**

JOIN clause: Connects the Guests table with the Bookings table using guest\_id.

WHERE clause: Filters bookings to include only those made between the specified start date (2024-01-01) and end date (2024-12-31).

Result: Displays the first and last names of guests, along with their booking date, who have bookings within the date range.

**Query 6:- Query to find the service that has been opted for by the most guests:**

SELECT s.service\_name, COUNT(bs.booking\_id) AS times\_opted

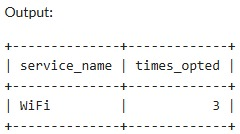
FROM Services s

JOIN Bookings bs ON s.service\_id = bs.booking\_id

GROUP BY s.service\_name

ORDER BY times\_opted DESC

LIMIT 1;



**Description:**

JOIN clause: Connects the Services table with the Bookings table using service\_id.

COUNT(): Counts how many times each service has been booked.

GROUP BY: Groups the results by service\_name to get a count for each service.

ORDER BY times\_opted DESC: Orders the services from most to least opted for.

LIMIT 1: Returns only the top service that was opted for the most.

Result: The query returns the name of the service and the number of times it was chosen, showing the most popular service among guests.

**Query 7:-** **Query to get all room numbers that are either booked or currently occupied**

SELECT room\_number

FROM Rooms

WHERE status = 'Occupied'

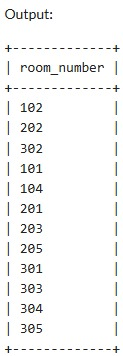
UNION

SELECT r.room\_number

FROM Bookings b

JOIN Rooms r ON b.room\_id = r.room\_id

WHERE b.status = 'Booked';

****

**Description:**

The overall effect of this query is to produce a combined list of room numbers from both Rooms and Bookings, specifically:

All rooms that are currently Occupied.

All rooms that are currently Booked.

The UNION ensures that if a room is both occupied and booked, it will only appear once in the final result set.

Use Case

This query is useful in hotel management to quickly identify all rooms that are currently either occupied or booked. This information can be critical for operational decisions, such as managing housekeeping schedules, checking availability, and understanding occupancy rates.

**Query 8:-** **Give the names of guests who have not chosen any payment methods**

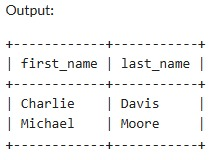
SELECT g.first\_name, g.last\_name

FROM Guests g

JOIN Bookings b ON g.Guest\_ID = b.guest\_id

LEFT JOIN Payments p ON b.booking\_id = p.booking\_id

WHERE p.payment\_method='NULL';



**Description:**

LEFT JOINs:

The first LEFT JOIN connects the Guests table to the Bookings table based on the guest\_id, allowing us to include all guests even if they don't have any bookings.

The second LEFT JOIN connects the Bookings table to the Payments table based on the booking\_id. This allows us to identify any payments related to the bookings.

WHERE Clause:

The condition p.payment\_id IS NULL filters the results to include only those guests who have not made any payments. If there are no corresponding entries in the Payments table for a guest's bookings, the payment\_id will be NULL.

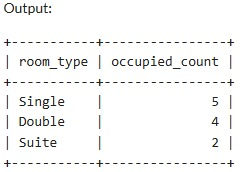
**Query 9:-** **What is the occupancy rate of each room type?**

SELECT r.room\_type, COUNT(b.room\_id) AS occupied\_count

FROM Rooms r

LEFT JOIN Bookings b ON r.Room\_ID = b.room\_id AND b.status = 'Booked'

GROUP BY r.room\_type;

****

**Description:**

SELECT r.room\_type, COUNT(b.room\_id) AS occupied\_count: This part specifies that we want to select the room type and count the number of times each room type has been booked, naming this count occupied\_count.

FROM Rooms r: Indicates that we are using the Rooms table, aliasing it as r.

LEFT JOIN Bookings b ON [r.RoomID](https://r.roomid/)= b.room\_id AND [b.status](https://b.status/)= 'Booked': We perform a left join with the Bookings table. This join links each room with its bookings, but only includes bookings where the status is 'Booked'. The left join ensures all rooms are included even if they have no

bookings.

GROUP BY r.room\_type: Groups the results by room type so that the count of bookings is

calculated for each room type.

**Query 10:-** **List the guests who have booked the most recently**

SELECT g.guest\_id, g.first\_name, g.last\_name, b.check\_in\_date, b.check\_out\_date

FROM Guests g

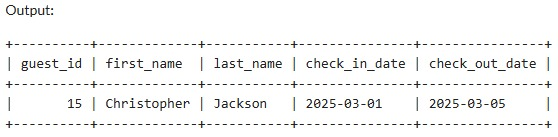
JOIN Bookings b ON g.guest\_id = b.guest\_id

WHERE b.check\_in\_date = (

SELECT MAX(check\_in\_date)

FROM Bookings

);



**Description:**

Inner Query:

The subquery SELECT MAX(booking\_date) FROM Bookings retrieves the most recent booking date from the Bookings table.

Main Query:

The main query selects guest information (guest\_id, first\_name, last\_name) and booking dates (check\_in\_date, check\_out\_date) from the Guests and Bookings tables.

It uses an INNER JOIN to connect the Guests table to the Bookings table on the guest\_id.

The WHERE clause filters results to include only those bookings that match the most recent booking date obtained from the subquery.

This query will return the guests who have made the most recent booking(s). If there are multiple guests with bookings on the same most recent date, all of them will be listed.

**Query-11:**

**Create a trigger that logs each new booking into the BookingLogs table**

CREATE OR REPLACE TRIGGER AfterRoomBookingInsert

AFTER UPDATE ON Rooms

FOR EACH ROW

BEGIN

IF :OLD.status != 'Booked' AND :NEW.status = 'Booked' THEN

INSERT INTO BookingLogs (booking\_id, guest\_id, room\_id, log\_message)

VALUES (bookinglogs\_seq.NEXTVAL, 123, :NEW.room\_id, 'Room booked');

END IF;

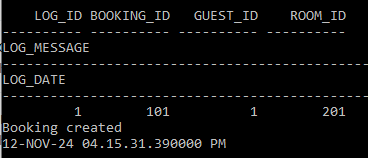
END;

/

INSERT INTO BookingLogs (log\_id, booking\_id, guest\_id, room\_id, log\_message) VALUES (bookinglogs\_seq.NEXTVAL, 101, 1, 201, 'Booking created');

select \* from bookingLogs;

Output:



**Description:**

The trigger **AfterRoomBookingInsert** logs room bookings by inserting a record into the BookingLogs table whenever the room's status is updated to "Booked". It checks if the room's previous status was not "Booked" and the new status is "Booked". If true, the trigger inserts a new log entry with the room\_id, a hardcoded guest\_id (123), and a log message. The booking\_id is generated using a sequence bookinglogs\_seq.NEXTVAL. This automatically tracks new bookings in the BookingLogs table.

## REFERENCES

1. Database Management Systems, Raghu Ramakrishnan, Johannes Gehrke, Tata

McGraw-Hill

2. Fundamentals of Database Systems, Elmasri Navathe, Pearson Education

3. Database Systems Design, Implementation, and Management, Peter Rob & Carlos

Coronel, 7th Edition, Cengage Learning

ONLINE RESOURCES:

1. https://www.w3schools.com/sql/default.asp

2. <https://www.javatpoint.com/dbms-tutorial>

3. <https://www.kaggle.com/datasets/mojtaba142/hotel-booking>

Source code available at: <https://github.com/KSuchita/DBMS-CBP-REPORT>