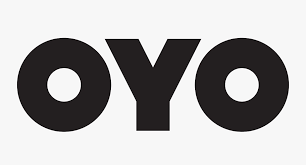
**OYO MANAGEMENT SYSTEM**

****

**OYO Management System**

An OYO Management System is a database project designed to streamline and automate the operations involved in managing hotels, rooms, customers, staff, bookings, payments, and customer feedback for the OYO hospitality chain. This system ensures efficient storage, retrieval, and analysis of data related to OYO’s business operations.

The goal of this project is to demonstrate the implementation of a relational database system for hotel management using MySQL. It involves designing normalized tables, inserting realistic data, performing SQL queries, and analysing key performance indicators such as occupancy rate, customer feedback, etc

**PROJECT AIM**

**Hotel Management:** Add, update, and manage hotel records. Track hotel details such as name, location (city/state), rating, features (e.g., Wi-Fi, pool), and star category.

**Room Management**: Maintain room inventory across hotels, including room types, prices, availability status, and occupancy capacity.

**Customer Management**: Store and manage customer records with personal details, contact information, and identity proof for verification purposes.

**Staff Management**: Maintain information on hotel staff including staff names, roles (e.g., manager, housekeeping), salaries, and hotel assignments.

**Booking Management**: Record customer bookings with check-in/check-out dates, status (booked, completed, cancelled), and linked room and hotel data.

**Payment Processing**: Track all payment transactions including booking ID, payment methods (UPI, card, cash), amount, and payment status (paid, pending, refunded).

**Feedback Tracking**: Collect and analyse customer feedback and ratings for hotels to monitor service quality and customer satisfaction.

**OBJECTIVES**

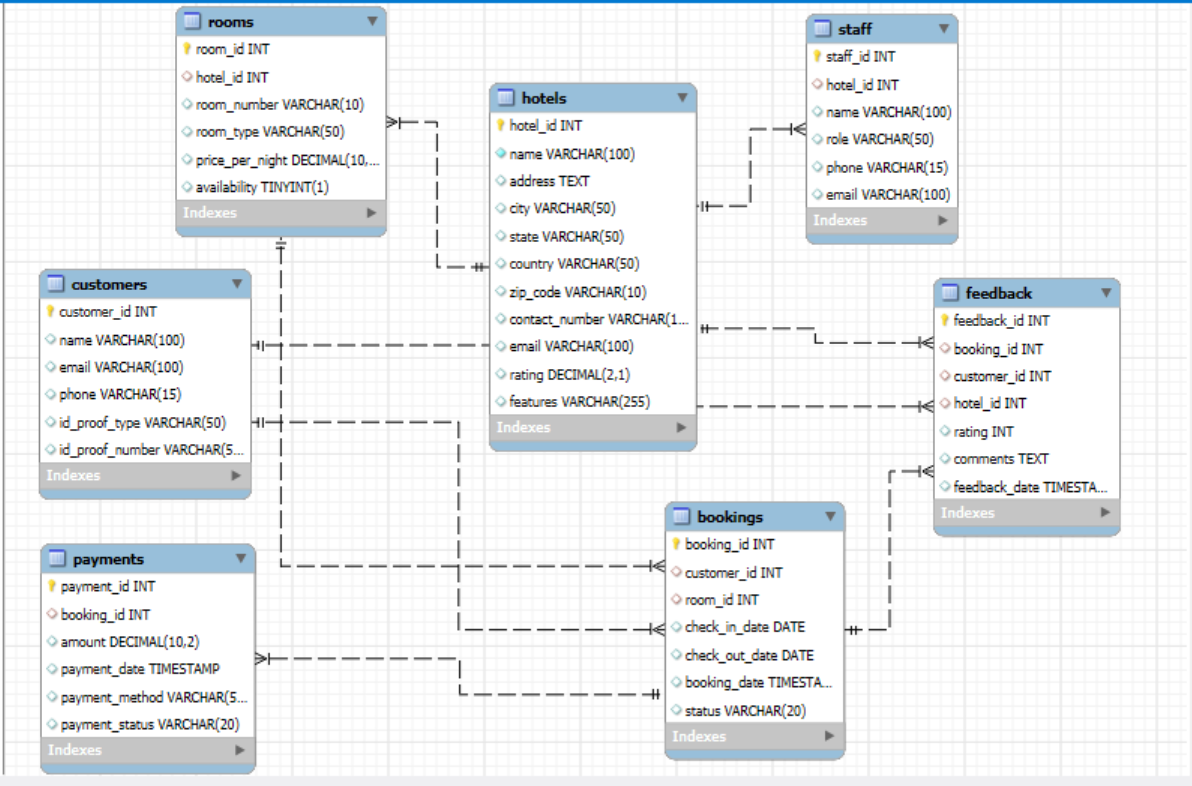
**Set up the OYO Management System Database**:  
Design and create normalized tables for hotels, rooms, customers, staff, bookings, payments, and feedback.

**CRUD Operations:**  
Perform Create, Read, Update, and Delete operations across all tables

**Advanced SQL Queries**:

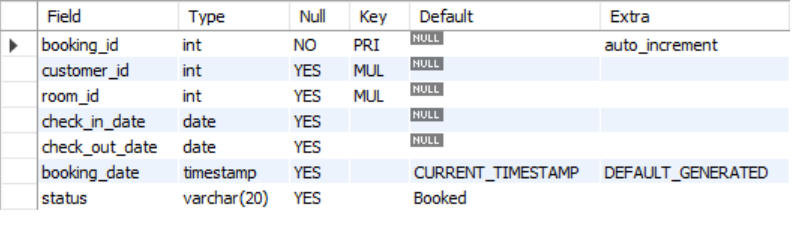
Develop complex queries to analyse and retrieve specific data.

**ER DIAGRAM FOR OYO MANAGEMENT SYSTEM**



**Table Description:**

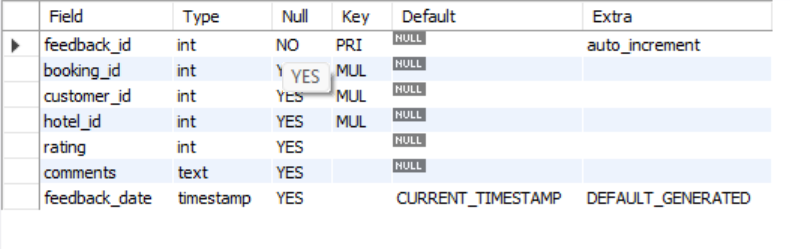
1. **BOOKINGS**

****

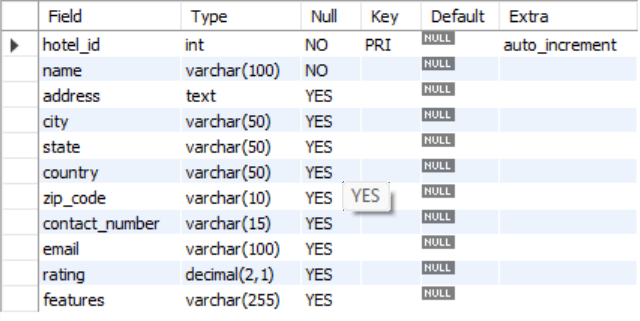
1. **CUTSOMERS**



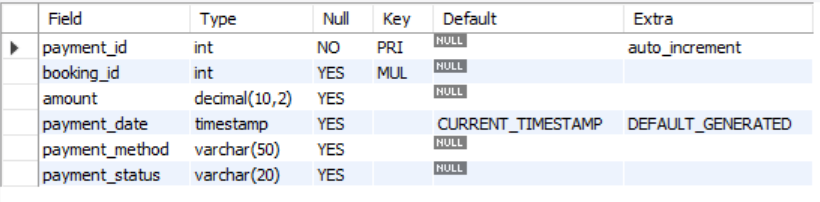
1. **FEEDBACK**

****

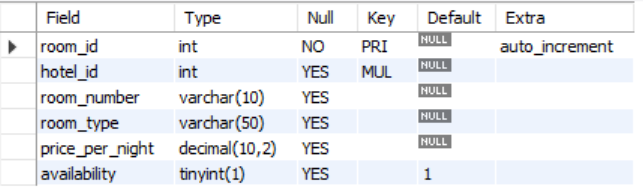
1. **HOTELS**

****

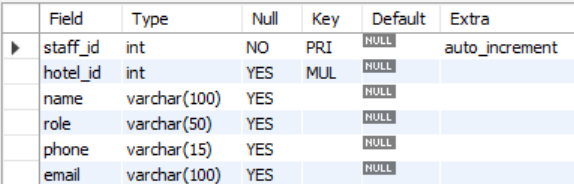
1. **PAYMENTS**

****

1. **ROOMS**

****

1. **STAFF**

****

**CREATING DATABASE:**

**CREATE DATABASE:** OYO\_MANAGEMENT\_SYSTEM

**USE:** OYO\_MANAGEMENT\_SYSTEM

**Table Creation & Insertion Commands**:

1. **CREATE TABLE Hotels:**

CREATE TABLE Hotels (

hotel\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

address TEXT,

city VARCHAR(50),

state VARCHAR(50),

country VARCHAR(50),

zip\_code VARCHAR(10),

contact\_number VARCHAR(15),

email VARCHAR(100),

rating DECIMAL(2, 1));

**Inserting into values Hotel:**

**INSERT INTO** **Hotels** (name, address, city, state, country, zip code, contact number, email, rating) **VALUES**

(1,'As Hotel', '8103 Jacob Common, East Gabrielaview, SC 74237', 'West Michaeltown', 'Maharashtre', 'India', '22334', '0656161001', 'hernandezkristin@gmail.com', 4.2),

(2,'Commercial Hotel', '8229 Benjamin Springs Suite 885, North Kyle, DC 56378', 'Port David', 'Goa', 'India', '33995', '1265553636', 'brownrebecca@yahoo.com', 4.8),

(3,'Stay Hotel', '74745 Sparks Shoals Apt. 835, Stacymouth, WI 89476', 'New Steven', ‘J&K’, 'India', '04641', '6766520345', 'ashleyellis@hotmail.com', 4.1),

(4,'Wave Inn', '4714 Michael Course, South Theresa, AZ 06754', 'Lake Dana', 'J&K', 'India', '31278', '5639871230', 'josephporter@taylor-thomas.com', 3.7),

(5,'Cozy Corner', '02747 Michael Haven Suite 231, North Michaelberg, MT 61499', 'South Daniel', 'Goa', 'India', '59340', '9988776655', 'connie31@gmail.com', 3.6),

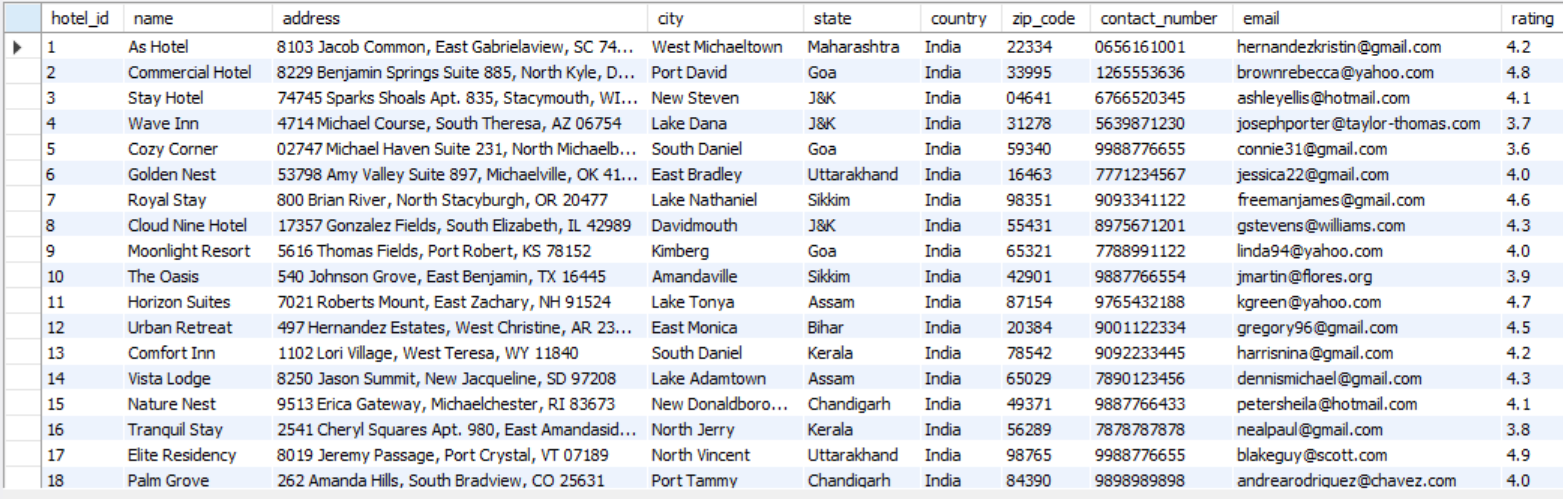
(6,'Golden Nest', '53798 Amy Valley Suite 897, Michaelville, OK 41535', 'East Bradley', 'Uttarakhand', 'India', '16463', '7771234567', 'jessica22@gmail.com', 4.0),

(7,'Royal Stay', '800 Brian River, North Stacyburgh, OR 20477', 'Lake Nathaniel', 'Sikkim', 'India', '98351', '9093341122', 'freemanjames@gmail.com', 4.6),

.

.

.



1. **CREATING TABLE Rooms:**

CREATE TABLE Rooms (

room\_id INT PRIMARY KEY AUTO\_INCREMENT,

hotel\_id INT,

room\_number VARCHAR(10),

room\_type VARCHAR(50), -- e.g., Single, Double, Deluxe

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

availability BOOLEAN DEFAULT TRUE,

FOREIGN KEY (hotel\_id) REFERENCES Hotels(hotel\_id));

**INSERTING INTO Rooms VALUES**

**INSERT INTO** Rooms (hotel\_id, room\_number, room\_type, price\_per\_night, availability) **VALUES**

(1, '101', 'Deluxe', 2599.99, TRUE),

(1, '102', 'Standard', 1799.50, TRUE),

(2, '201', 'Suite', 3500.00, FALSE),

(2, '202', 'Standard', 1900.75, TRUE),

(3, '301', 'Single', 1200.00, TRUE),

(3, '302', 'Double', 2100.00, FALSE),

(4, '401', 'Deluxe', 2500.00, TRUE),

(4, '402', 'Suite', 3700.00, TRUE),

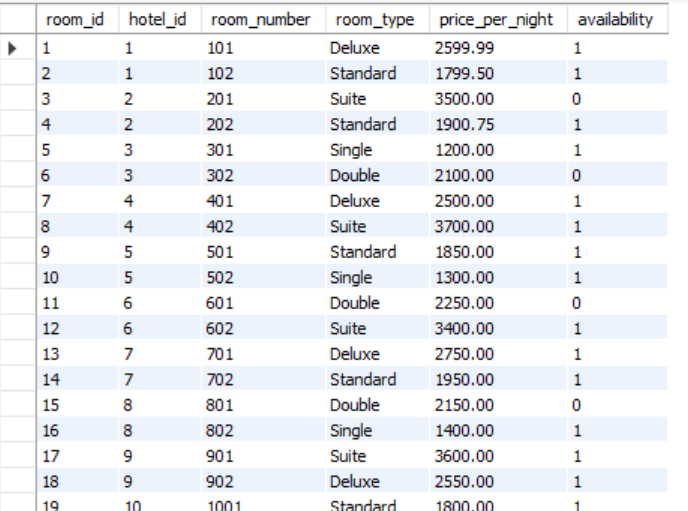
(5, '501', 'Standard', 1850.00, TRUE),

(5, '502', 'Single', 1300.00, TRUE),

(6, '601', 'Double', 2250.00, FALSE),

.

.



1. **CREATING TABLE Customers:**

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

email VARCHAR(100),

phone VARCHAR(15),

id\_proof\_type VARCHAR(50), -- e.g., Aadhar, Passport

id\_proof\_number VARCHAR(50));

**INSERTING INTO Customers VALUES:**

**INSERT INTO** Customers (name, email, phone, id\_proof\_type, id\_proof\_number) **VALUES**

('Ravi Sharma', 'ravi.sharma@gmail.com', '9876543210', 'Aadhar', '1234-5678-9012'),

('Priya Mehta', 'priya.mehta@yahoo.com', '8765432109', 'PAN', 'ABCDE1234F'),

('Aman Gupta', 'aman.gupta@outlook.com', '7654321098', 'Passport', 'P1234567'),

('Sneha Rao', 'sneha.rao@gmail.com', '6543210987', 'Voter ID', 'XYZ123456'),

('Karan Kapoor', 'karan.kapoor@live.com', '5432109876', 'Aadhar', '2345-6789-0123'),

('Divya Jain', 'divya.jain@gmail.com', '9123456780', 'PAN', 'AAAPL1234C'),

('Nikhil Arora', 'nikhil.arora@yahoo.com', '8899776655', 'Passport', 'N9876543'),

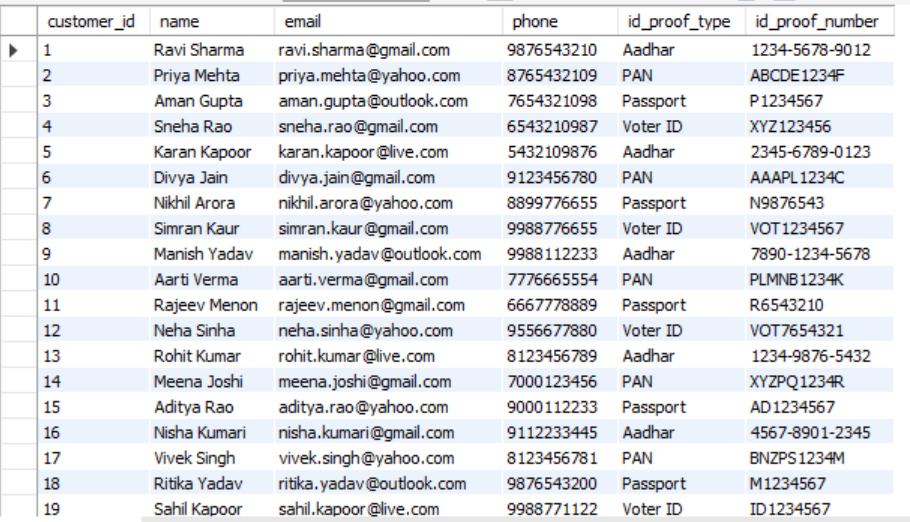
('Simran Kaur', 'simran.kaur@gmail.com', '9988776655', 'Voter ID', 'VOT1234567'),

('Manish Yadav', 'manish.yadav@outlook.com', '9988112233', 'Aadhar', '7890-1234-5678'),

('Aarti Verma', 'aarti.verma@gmail.com', '7776665554', 'PAN', 'PLMNB1234K'),

.

.



1. **CREATEING TABLE Bookings:**

CREATE TABLE Bookings (

booking\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

room\_id INT,

check\_in\_date DATE,

check\_out\_date DATE,

booking\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(20) DEFAULT 'Booked', -- Booked, Cancelled, Completed

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id),

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

**INSERTING INTO Bookings VALUES**

**INSERT INTO** Bookings (customer\_id, room\_id, check\_in\_date, check\_out\_date, status) **VALUES**

(1, 1, '2025-07-01', '2025-07-03', 'Completed'),

(2, 2, '2025-07-04', '2025-07-06', 'Booked'),

(3, 3, '2025-07-02', '2025-07-05', 'Cancelled'),

(4, 4, '2025-07-01', '2025-07-02', 'Completed'),

(5, 5, '2025-07-03', '2025-07-05', 'Booked'),

(6, 6, '2025-07-06', '2025-07-08', 'Completed'),

(7, 7, '2025-07-07', '2025-07-09', 'Booked'),

(8, 8, '2025-07-05', '2025-07-06', 'Cancelled'),

(9, 9, '2025-07-04', '2025-07-05', 'Completed'),

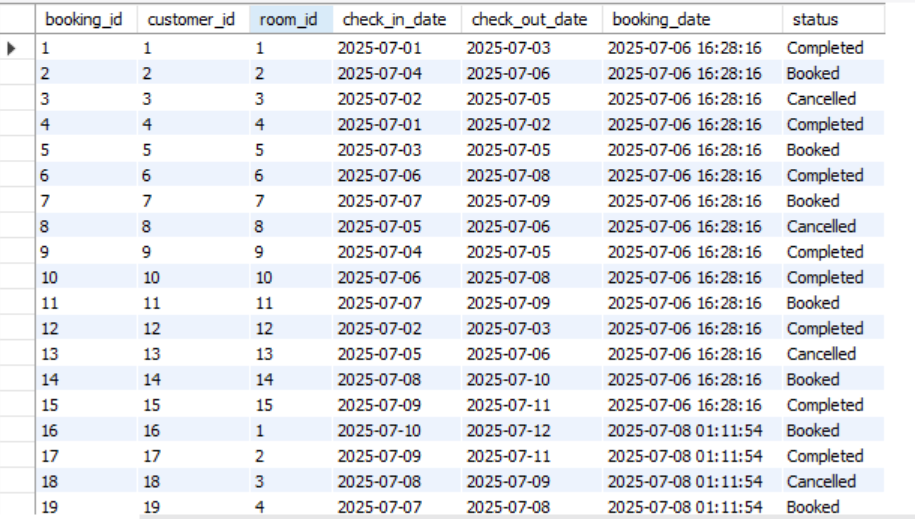
(10, 10, '2025-07-06', '2025-07-08', 'Completed'),

(11, 11, '2025-07-07', '2025-07-09', 'Booked'),

(12, 12, '2025-07-02', '2025-07-03', 'Completed'),

.

.



1. **CREATEING TABLE Payments:**

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

booking\_id INT,

amount DECIMAL(10, 2),

payment\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

payment\_method VARCHAR(50), -- e.g., Card, UPI, Cash

payment\_status VARCHAR(20), -- e.g., Paid, Failed, Pending

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

**INSERTING INTO Payments VALUES**

**INSERT INTO** Payments (booking\_id, amount, payment\_method, payment\_status) **VALUES**

(1, 5200.00, 'UPI', 'Paid'),

(2, 3600.00, 'Card', 'Pending'),

(3, 4800.00, 'Cash', 'Refunded'),

(4, 1200.00, 'UPI', 'Paid'),

(5, 3000.00, 'Card', 'Paid'),

(6, 4000.00, 'Cash', 'Paid'),

(7, 4200.00, 'UPI', 'Paid'),

(8, 1500.00, 'Card', 'Refunded'),

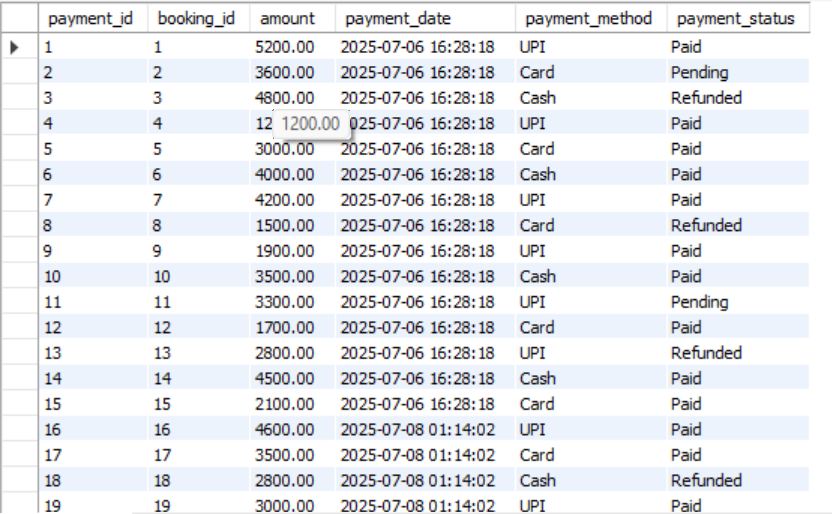
(9, 1900.00, 'UPI', 'Paid'),

(10, 3500.00, 'Cash', 'Paid'),

(11, 3300.00, 'UPI', 'Pending'),

.

.



1. **CREATEING TABLE Staff:**

CREATE TABLE Staff (

staff\_id INT PRIMARY KEY AUTO\_INCREMENT,

hotel\_id INT,

name VARCHAR(100),

role VARCHAR(50), -- e.g., Manager, Cleaner, Receptionist

phone VARCHAR(15),

email VARCHAR(100),

FOREIGN KEY (hotel\_id) REFERENCES Hotels(hotel\_id));

**INSERT INTO Staff VALUES**

**INSERT INTO** Staff (hotel\_id, name, role, phone, email) **VALUES**

(1, 'Suresh Patel', 'Manager', '9876543210', 'suresh.patel@oyo.com'),

(2, 'Aarti Singh', 'Receptionist', '8765432198', 'aarti.singh@oyo.com'),

(3, 'Rohan Das', 'Cleaner', '7654321987', 'rohan.das@oyo.com'),

(4, 'Priyanka Verma', 'Security', '6543219876', 'priyanka.verma@oyo.com'),

(5, 'Neeraj Pandey', 'Manager', '9432109876', 'neeraj.pandey@oyo.com'),

(6, 'Ankita Shah', 'Receptionist', '9123456781', 'ankita.shah@oyo.com'),

(7, 'Deepak Rawat', 'Cleaner', '8899776654', 'deepak.rawat@oyo.com'),

(8, 'Ritika Malhotra', 'Security', '9988776655', 'ritika.malhotra@oyo.com'),

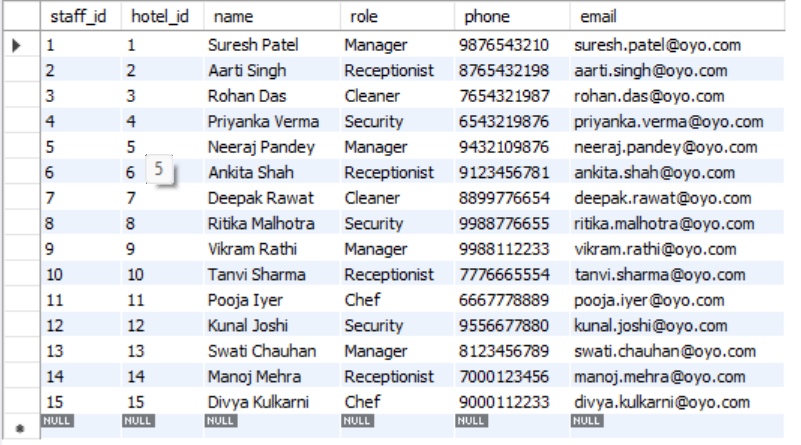
(9, 'Vikram Rathi', 'Manager', '9988112233', 'vikram.rathi@oyo.com'),

(10, 'Tanvi Sharma', 'Receptionist', '7776665554', 'tanvi.sharma@oyo.com'),

(11, 'Pooja Iyer', 'Chef', '6667778889', 'pooja.iyer@oyo.com'),

.

.



1. **CREATEING TABLE Feedback:**

CREATE TABLE Feedback (

feedback\_id INT PRIMARY KEY AUTO\_INCREMENT,

booking\_id INT,

customer\_id INT,

hotel\_id INT,

rating INT CHECK (rating BETWEEN 1 AND 5), -- Star rating: 1 to 5

comments TEXT,

feedback\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

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

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id),

FOREIGN KEY (hotel\_id) REFERENCES Hotels(hotel\_id));

**INSERT INTO Feedback VALUES**

INSERT INTO Feedback (booking\_id, customer\_id, hotel\_id, rating, comments) VALUES

(1, 1, 1, 5, 'Excellent stay and service.'),

(2, 2, 1, 4, 'Good room, could be cleaner.'),

(3, 3, 2, 3, 'Decent experience, average service.'),

(4, 4, 2, 4, 'Nice and quick check-in.'),

(5, 5, 3, 5, 'Wonderful hospitality!'),

(6, 6, 3, 2, 'Not satisfied with the cleanliness.'),

(7, 7, 4, 4, 'Good location and room.'),

(8, 8, 4, 3, 'Okay, staff was slow.'),

.

.



**BASIC QUESTIONS**

1. **Add an extra column called features to the Hotels table.**

ALTER TABLE Hotels

ADD COLUMN features VARCHAR(255);

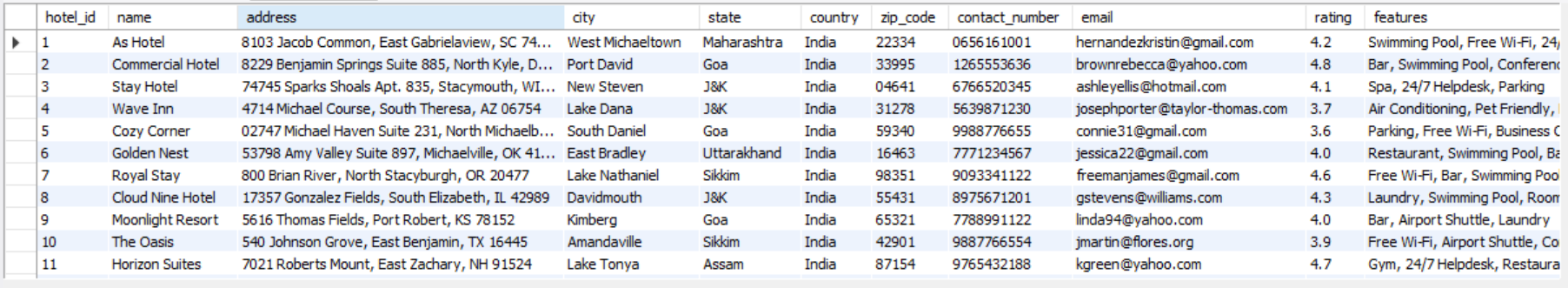
UPDATE Hotels SET features = 'Swimming Pool, Free Wi-Fi, 24/7 Helpdesk' WHERE hotel\_id = 1;

UPDATE Hotels SET features = 'Bar, Swimming Pool, Conference Room' WHERE hotel\_id = 2;

UPDATE Hotels SET features = 'Spa, 24/7 Helpdesk, Parking' WHERE hotel\_id = 3;

.

**OUTPUT:**



1. **How many rooms are available in each hotel?**

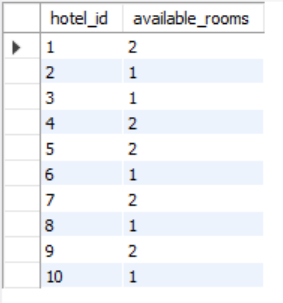
Select hotel\_id,count(\*) as available\_rooms

from rooms

where availability=true

group by hotel\_id;

**OUTPUT:**



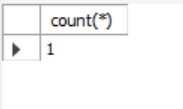
1. **Which rooms have a price per night greater than ₹2500 and are not available?**

select count(\*)

from rooms

where price\_per\_night>2500 and availability=false;

**OUTPUT:**

****

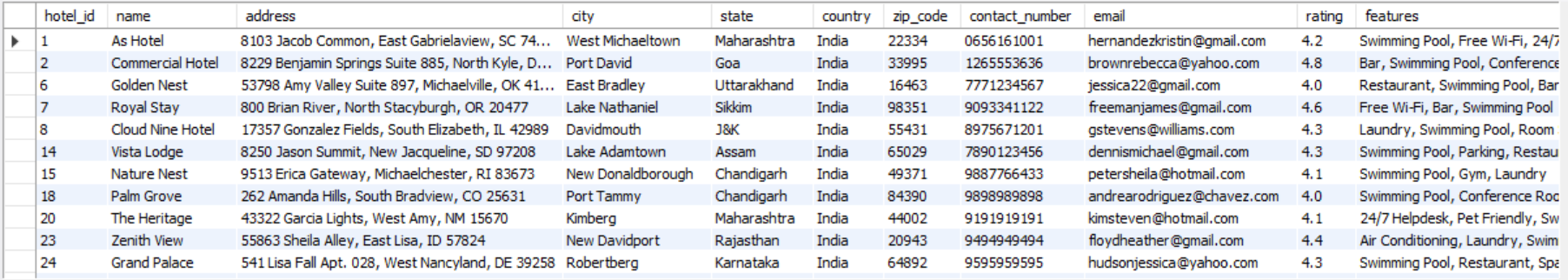
1. **Show hotels that have swimming pool in features?**

select\*

from hotels

where features like ('%swimming pool%');

**OUTPUT:**



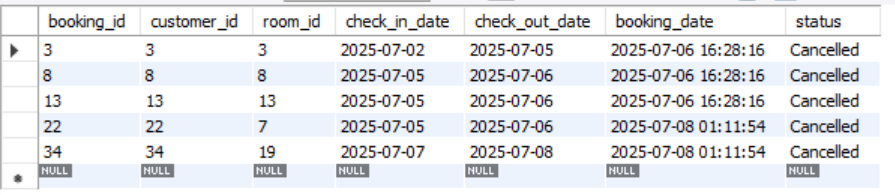
1. **Write a query to list all bookings that have cancelled between '2025-07-01' and '2025-07-10'?**

select \*

from bookings

where check\_in\_date between '2025-27-01'and '2025-07-07' and status='cancelled';

**OUTPUT:**

****

1. **Write a query to find the most commonly used payment method and average amount?**

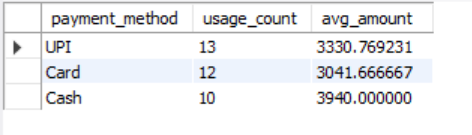
SELECT payment\_method, COUNT(\*) AS usage\_count,AVG(amount) AS avg\_amount

FROM Payments

GROUP BY payment\_method

ORDER BY usage\_count DESC;

**OUTPUT:**



1. **Write a query to find the top 5 hotels with the highest ratings?**

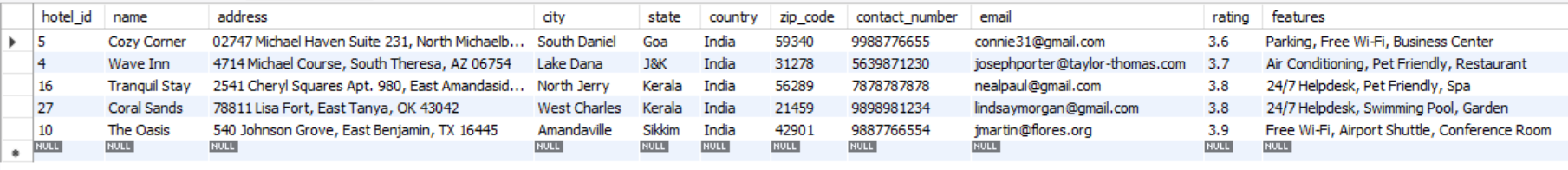
select \*

from hotels

order by rating

limit 5;

**OUTPUT:**



1. **Write a query to count how many hotels exist in each state, sorted by the highest count?**

select count(\*) as hotel\_count,state

from hotels

group by state

order by hotel\_count desc;

**OUTPUT:**



**SUB-QUERIES**

1. **List all hotels whose rating is higher than the average rating?**

SELECT name, rating

FROM Hotels

WHERE rating > (SELECT AVG(rating) FROM Hotels);

**OUTPUT:**



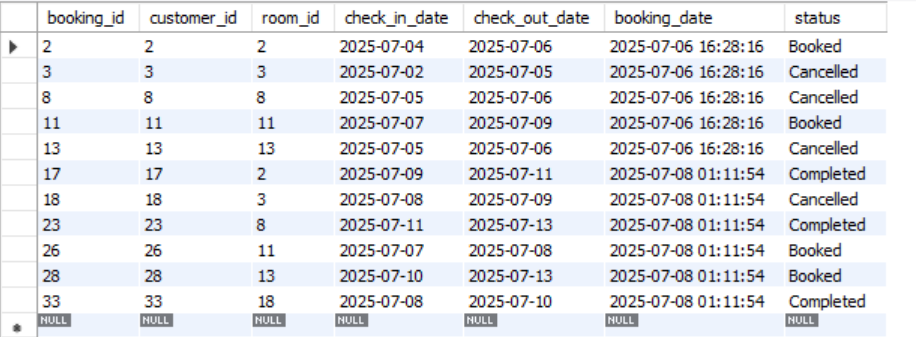
1. **List booking IDs that haven’t been paid for yet.?**

SELECT \*

FROM Bookings

WHERE room\_id NOT IN (SELECT booking\_id FROM Payments WHERE payment\_status = 'Paid');

**OUTPUT:**



1. **Get all room details from hotels rated 4.5 and above?**

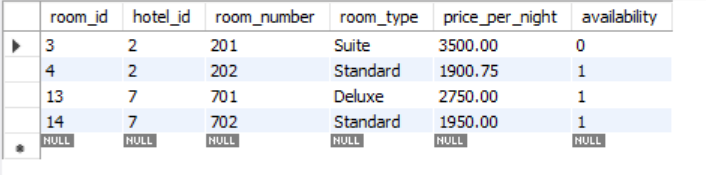
SELECT \*

FROM Rooms

WHERE hotel\_id IN (

SELECT hotel\_id FROM Hotels WHERE rating >= 4.5);

**OUTPUT:**



1. **Find staff members who work in hotels that have a gym?**

SELECT name, role

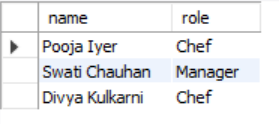
FROM Staff

WHERE hotel\_id IN (

SELECT hotel\_id FROM Hotels WHERE features LIKE '%Gym%'

);

**OUTPUT:**



1. **Show names of hotels that got 5-star feedback from any customer.**

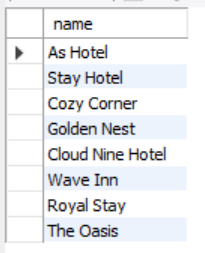
SELECT name

FROM Hotels

WHERE hotel\_id IN (

SELECT hotel\_id FROM Feedback WHERE rating = 5);

**OUTPUT:**



1. **Get all feedback comments for bookings paid via UPI.**

SELECT comments

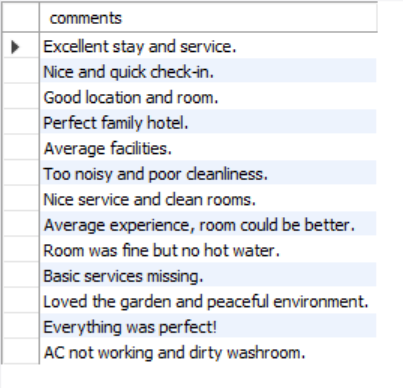
FROM Feedback

WHERE booking\_id IN (

SELECT booking\_id FROM Payments WHERE payment\_method = 'UPI'

);

**OUTPUT:**



**JOINS**

1. **List all bookings with customer name, hotel name, and check-in/check-out dates.**

SELECT c.name AS customer\_name, h.name AS hotel\_name, b.check\_in\_date, b.check\_out\_date

FROM Bookings b

JOIN Customers c ON b.customer\_id = c.customer\_id

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

JOIN Hotels h ON r.hotel\_id = h.hotel\_id;

**OUTPUT:**



1. **Show hotel names and average feedback ratings for hotels with more than 2 feedbacks.**

SELECT h.name AS hotel\_name, AVG(f.rating) AS avg\_rating

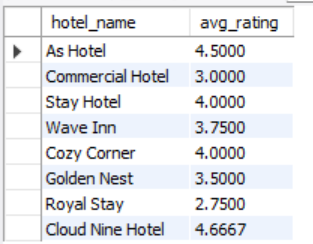
FROM Feedback f

JOIN Hotels h ON f.hotel\_id = h.hotel\_id

GROUP BY h.hotel\_id, h.name

HAVING COUNT(f.hotel\_id) > 2;

**OUTPUT:**



1. **Get the most expensive room booked by each customer.**

SELECT c.name AS customer\_name, MAX(price\_per\_night) AS max\_room\_price

FROM Bookings b

JOIN Customers c ON b.customer\_id = c.customer\_id

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

GROUP BY c.customer\_id, c.name;

**OUTPUT:**



1. **List hotel names and total revenue generated per hotel (from paid bookings only).**

SELECT h.name AS hotel\_name, SUM(p.amount) AS total\_revenue

FROM Payments p

JOIN Bookings b ON p.booking\_id = b.booking\_id

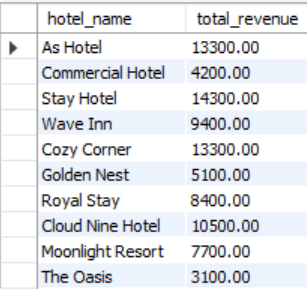
JOIN Rooms r ON b.room\_id = r.room\_id

JOIN Hotels h ON r.hotel\_id = h.hotel\_id

WHERE p.payment\_status = 'Paid'

GROUP BY h.hotel\_id, h.name;

**OUTPUT:**



1. **Show feedback for completed bookings only along with customer and hotel names.**

SELECT c.name AS customer\_name, h.name AS hotel\_name, f.rating, f.comments

FROM Feedback f

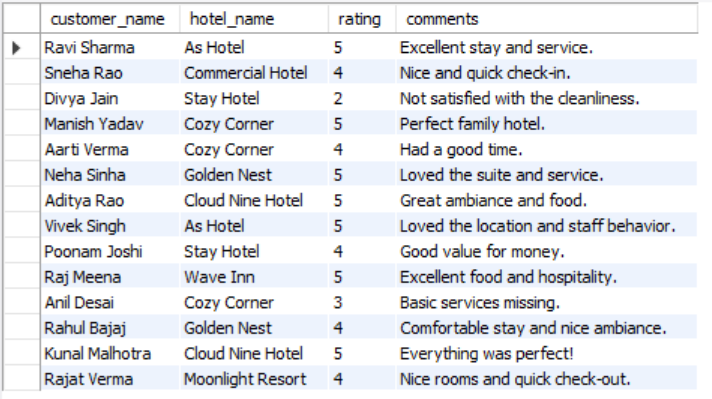
JOIN Bookings b ON f.booking\_id = b.booking\_id

JOIN Customers c ON f.customer\_id = c.customer\_id

JOIN Hotels h ON f.hotel\_id = h.hotel\_id

WHERE b.status = 'Completed';

**OUTPUT:**



**CONCLUSION**

This project focused on building a complete database system to manage hotel operations for the OYO platform. From hotel details and room bookings to customer feedback and features, the system was designed to organize everything in one place — clearly and efficiently.

The main goal was to make the system easy to use, accurate, and accessible. And we achieved that by using SQL to structure the database, insert real-world sample data, and run meaningful queries. Along the way, we applied practical skills like creating tables, setting relationships, removing duplicates, and analyzing the data with advanced SQL queries.

The final result is a solid, scalable system that can grow with OYO’s future needs. Overall, this project was a great hands-on experience in designing a real-world database and understanding how data supports daily business operations.