**HOTEL ROOM BOOKING**

-- Drop and Recreate the Database

DROP DATABASE IF EXISTS HotelBookingDB;

CREATE DATABASE HotelBookingDB;

USE HotelBookingDB;

-- Guests Table

CREATE TABLE IF NOT EXISTS Guests (

guest\_id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Room Types Table

CREATE TABLE IF NOT EXISTS RoomTypes (

room\_type\_id INT AUTO\_INCREMENT PRIMARY KEY,

type\_name VARCHAR(50) NOT NULL UNIQUE,

room\_description TEXT,

capacity INT NOT NULL,

price\_per\_night DECIMAL(10,2) NOT NULL

);

-- Rooms Table

CREATE TABLE IF NOT EXISTS Rooms (

room\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_number VARCHAR(10) UNIQUE NOT NULL,

room\_type\_id INT NOT NULL,

is\_active BOOLEAN DEFAULT TRUE,

FOREIGN KEY (room\_type\_id) REFERENCES RoomTypes(room\_type\_id)

);

-- Bookings Table

CREATE TABLE IF NOT EXISTS Bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

guest\_id INT NOT NULL,

room\_id INT NOT NULL,

check\_in DATE NOT NULL,

check\_out DATE NOT NULL,

booking\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

booking\_status ENUM('PENDING', 'CONFIRMED', 'CANCELLED', 'COMPLETED') DEFAULT 'PENDING',

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

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

CHECK (check\_in < check\_out)

);

-- Indexes

CREATE INDEX idx\_room\_type ON Rooms(room\_type\_id);

CREATE INDEX idx\_checkin\_checkout ON Bookings(check\_in, check\_out);

CREATE INDEX idx\_booking\_status ON Bookings(booking\_status);

-- Sample Data

-- Room Types

INSERT IGNORE INTO RoomTypes (type\_name, room\_description, capacity, price\_per\_night)

VALUES

('Single', 'Single bed room', 1, 80.00),

('Double', 'Double bed room', 2, 120.00),

('Suite', 'Luxury suite with premium amenities', 4, 300.00);

-- Guests

INSERT IGNORE INTO Guests (full\_name, email, phone) VALUES

('Ali Khan', 'ali@example.com', '03001234567'),

('Sara Ahmed', 'sara@example.com', '03007654321'),

('John Doe', 'john@example.com', '1234567890');

-- Rooms

INSERT IGNORE INTO Rooms (room\_number, room\_type\_id, is\_active)

VALUES

('101', 1, TRUE),

('102', 2, TRUE),

('201', 3, TRUE);

-- Initial Bookings

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES

(1, 1, '2025-07-01', '2025-07-05', 'CONFIRMED'),

(2, 2, '2025-07-02', '2025-07-06', 'CONFIRMED'),

(3, 3, '2025-07-03', '2025-07-07', 'CANCELLED');

-- Fix for Query 8 and 10

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES (1, 1, CURDATE() + INTERVAL 2 DAY, CURDATE() + INTERVAL 4 DAY, 'CONFIRMED');

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES (2, 2, CURDATE(), CURDATE() + INTERVAL 2 DAY, 'CONFIRMED');

-- Room Availability View

CREATE OR REPLACE VIEW AvailableRooms AS

SELECT r.room\_id, r.room\_number, rt.type\_name, rt.price\_per\_night

FROM Rooms r

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE r.is\_active = TRUE

AND NOT EXISTS (

SELECT 1 FROM Bookings b

WHERE b.room\_id = r.room\_id

AND b.booking\_status = 'CONFIRMED'

AND (b.check\_in < CURDATE() + INTERVAL 30 DAY)

AND (b.check\_out > CURDATE())

);

-- Confirm a Pending Booking (Booking ID 4 example)

CREATE TEMPORARY TABLE confirmable\_bookings AS

SELECT booking\_id

FROM Bookings b1

WHERE b1.booking\_id = 4

AND b1.booking\_status = 'PENDING'

AND NOT EXISTS (

SELECT 1 FROM Bookings b2

WHERE b2.room\_id = b1.room\_id

AND b2.booking\_status = 'CONFIRMED'

AND b2.booking\_id != b1.booking\_id

AND (b2.check\_in < b1.check\_out AND b2.check\_out > b1.check\_in)

);

UPDATE Bookings b

JOIN confirmable\_bookings cb ON b.booking\_id = cb.booking\_id

SET b.booking\_status = 'CONFIRMED';

DROP TEMPORARY TABLE confirmable\_bookings;

-- Final Bookings Status

SELECT \* FROM Bookings;

-- Additional Functional Queries

-- 1. Room Types with Avg Pricing

SELECT type\_name, capacity, AVG(price\_per\_night) AS avg\_price

FROM RoomTypes

GROUP BY type\_name, capacity;

-- 2. Bookings for Specific Guest (ID: 1)

SELECT b.\*, r.room\_number, rt.type\_name

FROM Bookings b

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

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE b.guest\_id = 1;

-- 3. Guests with More Than One Booking

SELECT g.full\_name, g.email, COUNT(b.booking\_id) AS total\_bookings

FROM Guests g

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

GROUP BY g.guest\_id

HAVING total\_bookings > 1;

-- 5. Room Utilization (%)

SELECT

rt.type\_name,

COUNT(b.booking\_id) AS bookings\_count,

ROUND(

SUM(DATEDIFF(b.check\_out, b.check\_in)) / (30.0 \* COUNT(r.room\_id)) \* 100, 2

) AS utilization\_percentage

FROM Bookings b

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

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE b.booking\_status = 'CONFIRMED'

AND MONTH(b.check\_in) = MONTH(CURDATE())

GROUP BY rt.type\_name;

-- 7. Today's Check-ins

SELECT \*

FROM Bookings

WHERE check\_in = CURDATE()

AND booking\_status = 'CONFIRMED';

-- 9. Guest Booking Summary View

CREATE OR REPLACE VIEW GuestBookingSummary AS

SELECT g.full\_name, g.email, COUNT(b.booking\_id) AS total\_bookings,

SUM(DATEDIFF(b.check\_out, b.check\_in)) AS total\_nights

FROM Guests g

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

GROUP BY g.guest\_id;

-- View output for Query 9

SELECT \* FROM GuestBookingSummary;

-- 10. Room Current Status View

CREATE OR REPLACE VIEW RoomStatus AS

SELECT r.room\_number, rt.type\_name,

CASE

WHEN EXISTS (

SELECT 1 FROM Bookings b

WHERE b.room\_id = r.room\_id

AND b.booking\_status = 'CONFIRMED'

AND b.check\_in <= CURDATE() AND b.check\_out >= CURDATE()

) THEN 'Occupied'

ELSE 'Available'

END AS current\_status

FROM Rooms r

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id;

-- View output for Query 10

SELECT \* FROM RoomStatus;

-- 11. Daily Booking Count in Current Month

SELECT DATE(booking\_date) AS date, COUNT(\*) AS bookings

FROM Bookings

WHERE MONTH(booking\_date) = MONTH(CURDATE())

GROUP BY DATE(booking\_date);

-- 12. Guests with Overlapping Bookings (Date: 2025-07-04)

SELECT DISTINCT g.full\_name, b.check\_in, b.check\_out

FROM Guests g

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

WHERE '2025-07-04' BETWEEN b.check\_in AND b.check\_out;

-- Drop and Recreate the Database

DROP DATABASE IF EXISTS HotelBookingDB;

CREATE DATABASE HotelBookingDB;

USE HotelBookingDB;

-- Guests Table

CREATE TABLE IF NOT EXISTS Guests (

guest\_id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Room Types Table

CREATE TABLE IF NOT EXISTS RoomTypes (

room\_type\_id INT AUTO\_INCREMENT PRIMARY KEY,

type\_name VARCHAR(50) NOT NULL UNIQUE,

room\_description TEXT,

capacity INT NOT NULL,

price\_per\_night DECIMAL(10,2) NOT NULL

);

-- Rooms Table

CREATE TABLE IF NOT EXISTS Rooms (

room\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_number VARCHAR(10) UNIQUE NOT NULL,

room\_type\_id INT NOT NULL,

is\_active BOOLEAN DEFAULT TRUE,

FOREIGN KEY (room\_type\_id) REFERENCES RoomTypes(room\_type\_id)

);

-- Bookings Table

CREATE TABLE IF NOT EXISTS Bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

guest\_id INT NOT NULL,

room\_id INT NOT NULL,

check\_in DATE NOT NULL,

check\_out DATE NOT NULL,

booking\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

booking\_status ENUM('PENDING', 'CONFIRMED', 'CANCELLED', 'COMPLETED') DEFAULT 'PENDING',

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

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

CHECK (check\_in < check\_out)

);

-- Indexes

CREATE INDEX idx\_room\_type ON Rooms(room\_type\_id);

CREATE INDEX idx\_checkin\_checkout ON Bookings(check\_in, check\_out);

CREATE INDEX idx\_booking\_status ON Bookings(booking\_status);

-- Sample Data

-- Room Types

INSERT IGNORE INTO RoomTypes (type\_name, room\_description, capacity, price\_per\_night)

VALUES

('Single', 'Single bed room', 1, 80.00),

('Double', 'Double bed room', 2, 120.00),

('Suite', 'Luxury suite with premium amenities', 4, 300.00);

-- Guests

INSERT IGNORE INTO Guests (full\_name, email, phone) VALUES

('Ali Khan', 'ali@example.com', '03001234567'),

('Sara Ahmed', 'sara@example.com', '03007654321'),

('John Doe', 'john@example.com', '1234567890');

-- Rooms

INSERT IGNORE INTO Rooms (room\_number, room\_type\_id, is\_active)

VALUES

('101', 1, TRUE),

('102', 2, TRUE),

('201', 3, TRUE);

-- Initial Bookings

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES

(1, 1, '2025-07-01', '2025-07-05', 'CONFIRMED'),

(2, 2, '2025-07-02', '2025-07-06', 'CONFIRMED'),

(3, 3, '2025-07-03', '2025-07-07', 'CANCELLED');

-- Fix for Query 8 and 10

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES (1, 1, CURDATE() + INTERVAL 2 DAY, CURDATE() + INTERVAL 4 DAY, 'CONFIRMED');

INSERT INTO Bookings (guest\_id, room\_id, check\_in, check\_out, booking\_status)

VALUES (2, 2, CURDATE(), CURDATE() + INTERVAL 2 DAY, 'CONFIRMED');

-- Room Availability View

CREATE OR REPLACE VIEW AvailableRooms AS

SELECT r.room\_id, r.room\_number, rt.type\_name, rt.price\_per\_night

FROM Rooms r

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE r.is\_active = TRUE

AND NOT EXISTS (

SELECT 1 FROM Bookings b

WHERE b.room\_id = r.room\_id

AND b.booking\_status = 'CONFIRMED'

AND (b.check\_in < CURDATE() + INTERVAL 30 DAY)

AND (b.check\_out > CURDATE())

);

-- Confirm a Pending Booking (Booking ID 4 example)

CREATE TEMPORARY TABLE confirmable\_bookings AS

SELECT booking\_id

FROM Bookings b1

WHERE b1.booking\_id = 4

AND b1.booking\_status = 'PENDING'

AND NOT EXISTS (

SELECT 1 FROM Bookings b2

WHERE b2.room\_id = b1.room\_id

AND b2.booking\_status = 'CONFIRMED'

AND b2.booking\_id != b1.booking\_id

AND (b2.check\_in < b1.check\_out AND b2.check\_out > b1.check\_in)

);

UPDATE Bookings b

JOIN confirmable\_bookings cb ON b.booking\_id = cb.booking\_id

SET b.booking\_status = 'CONFIRMED';

DROP TEMPORARY TABLE confirmable\_bookings;

-- Final Bookings Status

SELECT \* FROM Bookings;

-- Additional Functional Queries

-- 1. Room Types with Avg Pricing

SELECT type\_name, capacity, AVG(price\_per\_night) AS avg\_price

FROM RoomTypes

GROUP BY type\_name, capacity;

-- 2. Bookings for Specific Guest (ID: 1)

SELECT b.\*, r.room\_number, rt.type\_name

FROM Bookings b

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

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE b.guest\_id = 1;

-- 3. Guests with More Than One Booking

SELECT g.full\_name, g.email, COUNT(b.booking\_id) AS total\_bookings

FROM Guests g

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

GROUP BY g.guest\_id

HAVING total\_bookings > 1;

-- 4. Revenue per Room Type

SELECT rt.type\_name,

SUM(rt.price\_per\_night \* DATEDIFF(b.check\_out, b.check\_in)) AS total\_revenue

FROM Bookings b

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

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE b.booking\_status = 'COMPLETED'

GROUP BY rt.type\_name;

SELECT \* FROM Bookings WHERE booking\_status = 'COMPLETED';

-- 5. Room Utilization (%)

SELECT

rt.type\_name,

COUNT(b.booking\_id) AS bookings\_count,

ROUND(

SUM(DATEDIFF(b.check\_out, b.check\_in)) / (30.0 \* COUNT(r.room\_id)) \* 100, 2

) AS utilization\_percentage

FROM Bookings b

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

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id

WHERE b.booking\_status = 'CONFIRMED'

AND MONTH(b.check\_in) = MONTH(CURDATE())

GROUP BY rt.type\_name;

-- 7. Today's Check-ins

SELECT \*

FROM Bookings

WHERE check\_in = CURDATE()

AND booking\_status = 'CONFIRMED';

-- 8. Cancel Future Bookings for Room (ID: 1)

SELECT \* FROM Bookings

WHERE room\_id = 1

AND check\_in > CURDATE()

AND booking\_status = 'CONFIRMED';

-- 9. Guest Booking Summary View

CREATE OR REPLACE VIEW GuestBookingSummary AS

SELECT g.full\_name, g.email, COUNT(b.booking\_id) AS total\_bookings,

SUM(DATEDIFF(b.check\_out, b.check\_in)) AS total\_nights

FROM Guests g

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

GROUP BY g.guest\_id;

-- View output for Query 9

SELECT \* FROM GuestBookingSummary;

-- 10. Room Current Status View

CREATE OR REPLACE VIEW RoomStatus AS

SELECT r.room\_number, rt.type\_name,

CASE

WHEN EXISTS (

SELECT 1 FROM Bookings b

WHERE b.room\_id = r.room\_id

AND b.booking\_status = 'CONFIRMED'

AND b.check\_in <= CURDATE() AND b.check\_out >= CURDATE()

) THEN 'Occupied'

ELSE 'Available'

END AS current\_status

FROM Rooms r

JOIN RoomTypes rt ON r.room\_type\_id = rt.room\_type\_id;

-- View output for Query 10

SELECT \* FROM RoomStatus;

-- 11. Daily Booking Count in Current Month

SELECT DATE(booking\_date) AS date, COUNT(\*) AS bookings

FROM Bookings

WHERE MONTH(booking\_date) = MONTH(CURDATE())

GROUP BY DATE(booking\_date);

-- 12. Guests with Overlapping Bookings (Date: 2025-07-04)

SELECT DISTINCT g.full\_name, b.check\_in, b.check\_out

FROM Guests g

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

WHERE '2025-07-04' BETWEEN b.check\_in AND b.check\_out;