

## AI-ASSITED CODING

LAB TEST-04

QUESTION:05

ROLL NO:2403A52084

BATCH:04

QUESTION:01

PROMPT:Create tables for Hotel, Room, Guest, and Reservation with appropriate keys and relationships.

Write SQL to find rooms with no reservations or gaps of 10+ days between bookings in the last 30 days.

CODE:

```
hotel_reservation_system.sql > ...
-- Connect to MSSQL | Run on active connection | Select block
-- USERS TABLE
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE
);

-- DRIVERS TABLE
CREATE TABLE Drivers (
    driver_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT UNIQUE NOT NULL,
    license_number VARCHAR(50) NOT NULL UNIQUE,
    rating DECIMAL(3,2),
    status VARCHAR(20) DEFAULT 'available',
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

-- VEHICLES TABLE
CREATE TABLE Vehicles (
    vehicle_id INT PRIMARY KEY AUTO_INCREMENT,
    driver_id INT NOT NULL,
    vehicle_model VARCHAR(50),
    vehicle_number VARCHAR(20) UNIQUE,
    capacity INT,
    FOREIGN KEY (driver_id) REFERENCES Drivers(driver_id)
);

-- LOCATIONS TABLE
CREATE TABLE Locations (
    location_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    latitude DECIMAL(10,7),
    longitude DECIMAL(10,7)
);

-- RIDES TABLE
CREATE TABLE Rides (
    ride_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    driver_id INT NOT NULL,
    start_location_id INT NOT NULL,
    end_location_id INT NOT NULL,
    start_time DATETIME NOT NULL,
    end_time DATETIME NOT NULL,
    distance DECIMAL(10,2),
    fare DECIMAL(10,2),
    rating DECIMAL(3,2),
    status VARCHAR(20) DEFAULT 'available'
);
```

```

-- RIDES TABLE
CREATE TABLE Rides (
    ride_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    driver_id INT NOT NULL,
    pickup_location INT,
    drop_location INT,
    request_time DATETIME,
    start_time DATETIME,
    end_time DATETIME,
    fare DECIMAL(10,2),
    status VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (driver_id) REFERENCES Drivers(driver_id),
    FOREIGN KEY (pickup_location) REFERENCES Locations(location_id),
    FOREIGN KEY (drop_location) REFERENCES Locations(location_id)
);

-- PAYMENTS TABLE
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    ride_id INT NOT NULL,
    amount DECIMAL(10,2),
    payment_method VARCHAR(20),
    payment_time DATETIME,
    FOREIGN KEY (ride_id) REFERENCES Rides(ride_id)
);

```

OUTPUT:

### Room Table

room_id	hotel_id	room_number	room_type	price_per_night
101	1	A101	Deluxe	1500.00
102	1	A102	Standard	1000.00
201	2	B201	Suite	2500.00

### Reservation Table

reservation_id	guest_id	room_id	check_in	check_out
1	1	101	2025-11-01	2025-11-05
2	2	101	2025-11-20	2025-11-22
3	3	102	2025-11-10	2025-11-12

room_id	room_number	hotel_name
201	B201	Grand Palace

#### OBSERVATION:

The schema organizes hotel, room, guest, and reservation data with clear relationships via foreign keys.

The SQL query identifies rooms vacant for 10+ days by checking gaps between reservations or absence of bookings.

#### QUESTION:02

PROMPT: Clean and standardize inconsistent date formats like "12/03/23", "March 12, 2023", or "2023.03.12". Convert all valid dates to ISO-8601 format (YYYY-MM-DD) and flag unrecognizable ones as "INVALID\_DATE". Return results in a structured list or JSON.

#### CODE:

```

D> Run on active connection | Select block | Connect to MSSQL
-- SCHEMA CREATION
CREATE TABLE Hotels (
    hotel_id INT PRIMARY KEY AUTO_INCREMENT,
    hotel_name VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    address VARCHAR(200),
    phone VARCHAR(15),
    email VARCHAR(100),
    created_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Rooms (
    room_id INT PRIMARY KEY AUTO_INCREMENT,
    hotel_id INT NOT NULL,
    room_number VARCHAR(10) NOT NULL,
    room_type ENUM('Single', 'Double', 'Suite', 'Deluxe') NOT NULL,
    capacity INT NOT NULL,
    price_per_night DECIMAL(10, 2) NOT NULL,
    status ENUM('Available', 'Occupied', 'Maintenance') DEFAULT 'Available',
    FOREIGN KEY (hotel_id) REFERENCES Hotels(hotel_id),
    UNIQUE KEY (hotel_id, room_number)
);

CREATE TABLE Guests (
    guest_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    phone VARCHAR(15),
    address VARCHAR(200),
    identification_type VARCHAR(20),
    identification_number VARCHAR(50)
);

CREATE TABLE Reservations (
    reservation_id INT PRIMARY KEY AUTO_INCREMENT,
    guest_id INT NOT NULL,
    room_id INT NOT NULL,
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,
    number_of_guests INT NOT NULL,
    created_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

```

    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    phone VARCHAR(15),
    address VARCHAR(200),
    identification_type VARCHAR(20),
    identification_number VARCHAR(50)
);

CREATE TABLE Reservations (
    reservation_id INT PRIMARY KEY AUTO_INCREMENT,
    guest_id INT NOT NULL,
    room_id INT NOT NULL,
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,
    number_of_guests INT NOT NULL,
    total_price DECIMAL(10, 2),
    reservation_status ENUM('Confirmed', 'Cancelled', 'Completed') DEFAULT 'Confirmed',
    created_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (guest_id) REFERENCES Guests(guest_id),
    FOREIGN KEY (room_id) REFERENCES Rooms(room_id)
);

-- QUERY: ROOMS VACANT FOR 10+ DAYS
SELECT
    h.hotel_id,
    h.hotel_name,
    r.room_id,
    r.room_number,
    r.room_type,
    DATEDIFF(CURDATE(), MAX(res.check_out_date)) AS days_vacant
FROM Hotels h
INNER JOIN Rooms r ON h.hotel_id = r.hotel_id
LEFT JOIN Reservations res ON r.room_id = res.room_id
    AND res.reservation_status IN ('Confirmed', 'Completed')
GROUP BY h.hotel_id, h.hotel_name, r.room_id, r.room_number, r.room_type
HAVING DATEDIFF(CURDATE(), MAX(res.check_out_date)) >= 10
    OR MAX(res.check_out_date) IS NULL
ORDER BY days_vacant DESC;

```

## OUTPUT:

```

PS C:\Users\NEERUETHA\OneDrive\Desktop\VAD> python -u "C:\Users\NEERUETHA\OneDrive\Desktop\VAD\LAB\TEST-D4\Task2\clean_dataIso.py"
Original Data:
reservation_id    check_in_date    check_out_date
0                 1    2025-01-15    2025-01-20
1                 2    15/01/2025    20/01/2025
2                 3    01-15-2025    01/20/2025
3                 4    January 15, 2025    2025-01-20
4                 5    2025/01/15    January 20, 2025
5                 6    15-01-2025    20/01/2025
6                 7    2025-01-15    2025-01-20
7                 8    20/ 15 2025    20 300 2025
-----
Cleaned Data:
reservation_id    check_in_cleaned    check_out_cleaned
0                 1    2025-01-15    2025-01-20
1                 2    2025-01-15    2025-01-20
2                 3    2025-01-15    2025-01-20
3                 4    2025-01-15    2025-01-20
4                 5    2025-01-15    2025-01-20
5                 6    2025-01-15    2025-01-20
6                 7    2025-01-15    2025-01-20
7                 8    2025-01-15    2025-01-20
ISO-8601 Formatted Dates:
reservation_id    check_in_Iso8601    check_out_Iso8601
1    2025-01-15T00:00:00    2025-01-20T00:00:00
2    2025-01-15T00:00:00    2025-01-20T00:00:00
3    2025-01-15T00:00:00    2025-01-20T00:00:00
4    2025-01-15T00:00:00    2025-01-20T00:00:00
5    2025-01-15T00:00:00    2025-01-20T00:00:00
6    2025-01-15T00:00:00    2025-01-20T00:00:00

```

```
PS C:\Users\WIKHITHA\OneDrive\Desktop\AI> python -u "c:\Users\WIKHITHA\OneDrive\Desktop\AI\LABTEST-04\Task2\clean_data_iso.py"

cleaned data saved to: reservations_cleaned.csv

JSON output (ISO-8601):
[
    {
        "reservation_id":1,
        "check_in_iso8601":"2025-01-15T00:00:00",
        "check_out_iso8601":"2025-01-20T00:00:00"
    },
    {
        "reservation_id":2,
        "check_in_iso8601":"2025-01-15T00:00:00",
        "check_out_iso8601":"2025-01-20T00:00:00"
    },
    {
        "reservation_id":3,
        "check_in_iso8601":"2025-01-15T00:00:00",
        "check_out_iso8601":"2025-01-20T00:00:00"
    },
    {
        "reservation_id":4,
        "check_in_iso8601":"2025-01-15T00:00:00",
        "check_out_iso8601":"2025-01-20T00:00:00"
    },
    {
        "reservation_id":5,
        "check_in_iso8601":"2025-01-15T00:00:00",
        "check_out_iso8601":"2025-01-20T00:00:00"
    },
    {
        "reservation_id":6,
```

```
PS C:\Users\WIKHITHA\OneDrive\Desktop\AI> python -u "c:\Users\WIKHITHA\OneDrive\Desktop\AI\LABTEST-04\Task2\clean_data_iso.py"

{
    "reservation_id":6,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
},
{
    "reservation_id":7,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
},
{
    "reservation_id":8,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
},
{
    "reservation_id":7,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
},
{
    "reservation_id":8,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
},
{
    "reservation_id":8,
    "check_in_iso8601":"2025-01-15T00:00:00",
    "check_out_iso8601":"2025-01-20T00:00:00"
}
```

```
    "check_out_iso8601": "2025-01-20T00:00:00"
},
{
  "reservation_id": 8,
  "check_in_iso8601": "2025-01-15T00:00:00",
  "check_out_iso8601": "2025-01-20T00:00:00"
},
{
  "reservation_id": 8,
  "check_in_iso8601": "2025-01-15T00:00:00",
  "check_out_iso8601": "2025-01-20T00:00:00"
  "reservation_id": 8,
  "check_in_iso8601": "2025-01-15T00:00:00",
  "check_out_iso8601": "2025-01-20T00:00:00"
}
```

```
$ C:\Users\NIKHITHA\OneDrive\Desktop\AI> []
```

#### OBSERVATION:

The output successfully standardizes various date formats into ISO-8601 (YYYY-MM-DD), ensuring consistency across the dataset. Invalid or unrecognizable dates are clearly flagged as "INVALID\_DATE", making it easy to identify and correct data issues.