Hotel Management System Database Project

🏨 Project Overview

This project implements a sophisticated relational database schema for a **Hotel Management System**. It is designed to mirror real-world hospitality workflows, managing the end-to-end lifecycle of a guest's stay—from the initial booking lead to room occupancy, auxiliary service billing, and final performance analytics.

The system is architected using **Third Normal Form (3NF)** principles to ensure high data integrity, eliminate redundancy, and support scalable operations.

📊 Entity-Relationship Diagram (ERD)

The database architecture is centered around the **Reservation** and **Invoice** entities, which act as the primary bridges between guest profiles and financial transactions.

```
erDiagram
    CUSTOMER ||--|| ADDRESS : "has profile"
    CUSTOMER ||--o{ BOOKING : "initiates"
    CUSTOMER ||--o{ RESERVATION : "holds"
    CUSTOMER ||--o{ INVOICE : "receives"

    BOOKING ||--o{ RESERVATION : "converts to"
    RESERVATION ||--|| ROOM : "occupies"
    RESERVATION ||--|| INVOICE : "triggers billing"

    INVOICE ||--o{ LINE : "details charges"
    SERVICES ||--o{ LINE : "categorizes"

    INVOICE ||--|| TRANSACTIONS : "settled by"
    TRANSACTIONS ||--o{ SATISFACTION : "collects feedback"
```

🔍 Detailed Table Analysis & Purpose

1. Identity & Demographics

- **CUSTOMER**: Acts as the central hub for all guest interactions. It stores unique identifiers and contact details necessary for communication and loyalty tracking.
- **ADDRESS**: A separate normalized entity to store physical location data. By decoupling this from the `CUSTOMER` table, the system can handle guest relocations or multiple addresses without corrupting historical records.

2. Operational Workflow

- **BOOKING**: Records the "Intent to Stay." Its primary purpose is to track the **Booking Source** (e.g., Direct, Expedia, Travel Agent), which allows management to analyze which marketing channels are most effective.
- **RESERVATION**: The "Active Record." This table links a specific customer to a timeframe (Check-in/Check-out). It serves as the master key that connects the booking intent to a physical room.
- **ROOM**: Manages physical inventory. It tracks fixed attributes like room type (Suite, Standard) and bed configuration, along with dynamic attributes like the base nightly price.

3. Financial Ecosystem

- **SERVICES**: A master price list or "Product Catalog" for non-room charges. It stores the standardized cost for items like Spa sessions, Laundry, or Dining.
- **INVOICE**: The financial header. It is triggered upon reservation and serves as the container for all costs associated with a single stay.
- **LINE (Invoice Line Items)**: A bridge table that allows for **Atomic Billing**. It maps specific services to an invoice with a quantity, allowing the system to handle multiple charges (e.g., 3 Breakfasts, 1 Spa session) under one bill.
- **TRANSACTIONS**: Records the actual movement of money. It tracks the payment method and timestamp, serving as the official audit trail for the accounting department.

4. Experience Analytics

- **SATISFACTION**: Captures post-stay feedback. By linking this to the `TRANSACTION_ID`, the system can correlate guest happiness with specific room types or services used, providing actionable insights for hotel management.

⚙️ Core Functionality & Relational Logic

🔄 The "Stay-to-Bill" Lifecycle

1. **Lead Generation**: A `BOOKING` source is identified.
2. **Occupancy**: A `RESERVATION` is created, which "locks" a `ROOM`.
3. **Consumption**: During the stay, any service used is added as a `LINE` item linked to the `INVOICE`.
4. **Settlement**: Upon check-out, a `TRANSACTION` is processed, closing the `INVOICE`.
5. **Quality Control**: The guest provides a `SATISFACTION` rating based on that transaction.

🔗 Key Relationships

- **One-to-Many (1:N)**: A single `CUSTOMER` can have many `RESERVATIONS` over time, but each reservation belongs to only one customer.
- **One-to-One (1:1)**: During a specific timeframe, one `RESERVATION` is assigned to one `ROOM` to prevent overbooking.
- **Many-to-Many (M:N) Resolved**: The `LINE` table resolves the relationship between `INVOICE` and `SERVICES` . One invoice can have many services, and one service type can appear on many invoices.

🚀 Technical Showcase

🛠️ Advanced Dynamic Billing

This logic aggregates data across five tables to calculate a guest's total financial liability.

```
SELECT
    C.first_name,
    C.last_name,
    I.invoice_id,
    (R.price + SUM(L.quantity * S.service_price)) AS total_due
FROM Customer C
JOIN Invoice I ON C.customer_id = I.customer_id
JOIN Reservation Res ON I.res_id = Res.res_id
JOIN Room R ON Res.res_id = R.res_id
JOIN Line L ON I.invoice_id = L.invoice_id
JOIN Services S ON L.service_id = S.service_id
GROUP BY I.invoice_id;
```

📈 Management Reporting View

Automated logic to monitor real-time business health.

```
CREATE VIEW View_Occupancy_Report AS
SELECT
    room_type,
    COUNT(room_num) AS capacity,
    SUM(CASE WHEN status = 'Occupied' THEN 1 ELSE 0 END) AS occupied,
    ROUND((SUM(CASE WHEN status = 'Occupied' THEN 1 ELSE 0 END) * 100.0 / COUNT(r
FROM Room
GROUP BY room_type;
```