Project Report: HotelSphere

An Integrated SQL Database Solution for Hotel Chain Management

1. Problem Statement

Modern hotel chains operate in a highly competitive and complex environment. Managing a multi-location hotel business involves overseeing a wide array of interconnected operations, including guest reservations, customer relationships, human resources, payroll, procurement, and inventory management. Many hotel chains suffer from data fragmentation, where critical information is stored in disparate, non-integrated systems. This leads to several significant challenges:

- **Data Redundancy and Inconsistency:** The same information (e.g., customer details, supplier information) is often stored in multiple places, leading to inconsistencies and making updates cumbersome.
- **Operational Inefficiency:** Lack of a centralized system makes it difficult to get a holistic view of the business. Simple tasks, like tracking a guest's history across different hotel branches or managing inventory levels chain-wide, become complex and time-consuming.
- **Impaired Decision-Making:** Without a unified data source, management cannot perform effective business analytics. It is challenging to identify key trends like room occupancy rates, determine top-performing customers, or spot gaps in the supply chain.
- **Scalability Issues:** As the hotel chain grows, decentralized systems become increasingly difficult and expensive to manage, hindering scalability.

The **HotelSphere** project aims to solve these problems by designing and implementing a centralized, end-to-end, enterprise-grade SQL database solution. This system will integrate Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Supply Chain Management (SCM) modules into a single, cohesive, and normalized relational database. The goal is to ensure data integrity, eliminate redundancy, streamline operations, and provide powerful business intelligence capabilities to support strategic decision-making.

2. Entity Information

To model the entire hotel chain operation, the following key entities were identified. Each entity represents a fundamental component of the business.

Core Operations & CRM Entities:

- **Hotel:** Represents a single hotel branch in the chain.
    - Attributes: HotelID (Primary Key), Name, Address, City, Country, PhoneNumber, StarRating.
- **RoomType:** Describes a category of rooms available (e.g., Standard, Deluxe, Suite).
    - Attributes: RoomTypeID (Primary Key), TypeName, Description, BasePrice.
- **Room:** Represents a specific physical room in a hotel.
    - Attributes: RoomID (Primary Key), RoomNumber, HotelID (Foreign Key),

RoomTypeID (Foreign Key), Status.

- **Customer:** Stores information about guests.
  - Attributes: CustomerID (Primary Key), FirstName, LastName, Email, PhoneNumber, Address.
- **Booking:** Represents a reservation made by a customer.
  - Attributes: BookingID (Primary Key), CustomerID (Foreign Key), HotelID (Foreign Key), RoomID (Foreign Key), CheckInDate, CheckOutDate, TotalAmount.
- **Payment:** Records payments made for bookings.
  - Attributes: PaymentID (Primary Key), BookingID (Foreign Key), PaymentDate, Amount, PaymentMethod.
- **LoyaltyProgram:** Manages the customer loyalty program.
  - Attributes: LoyaltyID (Primary Key), CustomerID (Foreign Key), Points, TierLevel.

ERP (HR & Payroll) Entities:

- **Department:** Represents a department within a hotel (e.g., Front Desk, Housekeeping, HR).
  - Attributes: DeptID (Primary Key), DeptName.
- **Role:** Defines job roles for employees.
  - Attributes: RoleID (Primary Key), RoleName, JobDescription.
- **Employee:** Stores information about all staff members.
  - Attributes: EmployeeID (Primary Key), FirstName, LastName, SSN, HireDate, HotelID (Foreign Key), DeptID (Foreign Key), RoleID (Foreign Key).
- **Payroll:** Manages salary payments to employees.
  - Attributes: PayrollID (Primary Key), EmployeeID (Foreign Key), PayPeriodStartDate, PayPeriodEndDate, BaseSalary, Bonuses, Deductions, FinalSalary.

SCM (Procurement & Inventory) Entities:

- **Supplier:** Information about vendors who supply goods to the hotels.
  - Attributes: SupplierID (Primary Key), SupplierName, ContactPerson, PhoneNumber, Address.
- **Product:** Represents an item that can be procured (e.g., bed linens, toiletries, food items).
  - Attributes: ProductID (Primary Key), ProductName, Description, Category.
- **Warehouse:** Represents a storage facility, typically one per hotel.
  - Attributes: WarehouseID (Primary Key), HotelID (Foreign Key), Location.
- **Inventory:** Tracks the quantity of products in a warehouse.
  - Attributes: InventoryID (Primary Key), WarehouseID (Foreign Key), ProductID (Foreign Key), Quantity, ReorderLevel.
- **PurchaseOrder:** A request sent to a supplier to purchase products.
  - Attributes: PO_ID (Primary Key), HotelID (Foreign Key), SupplierID (Foreign Key), OrderDate, Status.
- **OrderDetail:** A weak entity that links products to a purchase order.
  - Attributes: PO_ID (Foreign Key), ProductID (Foreign Key), Quantity, UnitPrice.

3. Assumptions in ER Model

The design of the Entity-Relationship (ER) model is based on the following key assumptions about the business rules of the HotelSphere chain:

1. **Hotel and Room Structure:**

    - The chain consists of multiple hotels, each uniquely identifiable.
    - Each hotel has multiple rooms, but a room belongs to only one hotel.
    - Rooms are categorized by RoomType (e.g., 'Standard', 'Deluxe'). The price is primarily determined by RoomType, but can be adjusted in the final booking.
2. **Customer and Bookings:**
    - A customer can exist in the system without making a booking.
    - A single customer can make multiple bookings.
    - Each booking is made by exactly one customer for one specific room in a particular hotel.
    - A booking has a defined check-in and check-out date.
3. **Employees and Departments:**
    - An employee works at exactly one hotel branch.
    - Each employee is assigned to one primary department and has one specific role.
    - Departments are standardized across the hotel chain (e.g., 'HR' exists as a concept, but each hotel has its own HR staff).
4. **Suppliers and Procurement:**
    - A single purchase order is issued by one hotel to one supplier.
    - A purchase order can contain multiple products, and a single product can be part of multiple orders (M:N relationship, requiring a linking table OrderDetail).
    - Each hotel manages its own procurement and has its own warehouse/inventory.
5. **Weak Entities:**
    - OrderDetail is a weak entity as it cannot exist without a PurchaseOrder. Its primary key is a composite key formed from PO_ID and ProductID.
6. **Derived Attributes:**
    - In the Booking entity, the TotalAmount could be a derived attribute calculated from the room price and the duration of stay (CheckOutDate - CheckInDate).
    - In the Payroll entity, FinalSalary is a derived attribute calculated as BaseSalary + Bonuses - Deductions.

4. Relationships Information

The relationships between the entities define the logical structure of the database. The cardinalities (1:1, 1:N, M:N) are crucial for establishing foreign key constraints.

- **Hotel and Room (1:N):**
    - One Hotel can have many Rooms.
    - Each Room belongs to exactly one Hotel.
- **RoomType and Room (1:N):**
    - One RoomType can apply to many Rooms.
    - Each Room is of exactly one RoomType.
- **Customer and Booking (1:N):**
    - One Customer can make many Bookings.
    - Each Booking is made by exactly one Customer.
- **Hotel and Booking (1:N):**
    - One Hotel can have many Bookings.
    - Each Booking is associated with exactly one Hotel.
- **Room and Booking (1:N):**
    - One Room can be booked many times (over different periods).
    - A Booking is for exactly one Room.

- **Booking and Payment (1:N):**
  - One Booking can have multiple partial Payments.
  - Each Payment is associated with exactly one Booking.
- **Customer and LoyaltyProgram (1:1):**
  - One Customer has one LoyaltyProgram account.
  - Each LoyaltyProgram account belongs to exactly one Customer.
- **Hotel and Employee (1:N):**
  - One Hotel employs many Employees.
  - Each Employee works at exactly one Hotel.
- **Department and Employee (1:N):**
  - One Department can have many Employees.
  - Each Employee is assigned to exactly one Department.
- **Role and Employee (1:N):**
  - One Role can be held by many Employees.
  - Each Employee has exactly one Role.
- **Employee and Payroll (1:N):**
  - One Employee can have many Payroll records (one for each pay period).
  - Each Payroll record is for exactly one Employee.
- **Hotel and PurchaseOrder (1:N):**
  - One Hotel can place many PurchaseOrders.
  - Each PurchaseOrder is placed by exactly one Hotel.
- **Supplier and PurchaseOrder (1:N):**
  - One Supplier can receive many PurchaseOrders.
  - Each PurchaseOrder is sent to exactly one Supplier.
- **PurchaseOrder and Product (M:N):**
  - A PurchaseOrder can contain many Products.
  - A Product can be included in many PurchaseOrders.
  - This is resolved by a linking table: OrderDetail.
    - PurchaseOrder to OrderDetail is 1:N.
    - Product to OrderDetail is 1:N.
- **Hotel and Warehouse (1:1):**
  - We assume each Hotel has exactly one dedicated Warehouse.
- **Warehouse and Inventory (1:N):**
  - One Warehouse stores many types of products in its Inventory.
  - Each Inventory record (for a specific product) belongs to one Warehouse.
- **Product and Inventory (1:N):**
  - One Product can be in the Inventory of many warehouses.
  - An Inventory record pertains to exactly one Product.

5. ER Diagram

The ER Diagram visually represents the entities, their attributes, and the relationships between them. A comprehensive diagram would be created using a tool like Draw.io. Below is a Mermaid script which can be rendered to visualize the diagram.

```
erDiagram
    CUSTOMER {
        int CustomerID PK
        varchar FirstName
```

```
    varchar LastName
    varchar Email
    varchar PhoneNumber
}
BOOKING {
    int BookingID PK
    int CustomerID FK
    int HotelID FK
    int RoomID FK
    date CheckInDate
    date CheckOutDate
    decimal TotalAmount
}
PAYMENT {
    int PaymentID PK
    int BookingID FK
    date PaymentDate
    decimal Amount
}
HOTEL {
    int HotelID PK
    varchar Name
    varchar Address
    varchar City
}
ROOM {
    int RoomID PK
    varchar RoomNumber
    int HotelID FK
    int RoomTypeID FK
    varchar Status
}
ROOM_TYPE {
    int RoomTypeID PK
    varchar TypeName
    decimal BasePrice
}
EMPLOYEE {
    int EmployeeID PK
    varchar FirstName
    varchar LastName
    int HotelID FK
    int DeptID FK
    int RoleID FK
}
DEPARTMENT {
    int DeptID PK
    varchar DeptName
```

```
}
ROLE {
    int RoleID PK
    varchar RoleName
}
PURCHASE_ORDER {
    int PO_ID PK
    int HotelID FK
    int SupplierID FK
    date OrderDate
}
ORDER_DETAIL {
    int PO_ID PK, FK
    int ProductID PK, FK
    int Quantity
    decimal UnitPrice
}
PRODUCT {
    int ProductID PK
    varchar ProductName
    varchar Category
}
SUPPLIER {
    int SupplierID PK
    varchar SupplierName
    varchar ContactPerson
}

CUSTOMER ||--o{ BOOKING : makes
BOOKING ||--|{ PAYMENT : has
HOTEL ||--o{ ROOM : has
HOTEL ||--o{ BOOKING : receives
HOTEL ||--o{ EMPLOYEE : employs
HOTEL ||--o{ PURCHASE_ORDER : places
ROOM_TYPE ||--o{ ROOM : categorizes
ROOM ||--o{ BOOKING : is_for
DEPARTMENT ||--o{ EMPLOYEE : has
ROLE ||--o{ EMPLOYEE : has
SUPPLIER ||--o{ PURCHASE_ORDER : receives
PURCHASE_ORDER ||--|{ ORDER_DETAIL : contains
PRODUCT ||--|{ ORDER_DETAIL : details
```

6. ER Schema before Normalization

Before applying normalization, one might design tables that group related information conveniently, but this leads to data redundancy and modification anomalies.
Example 1: Denormalized Bookings Table

A single table could be created to hold all booking information, including customer and hotel details. BookingsMaster (BookingID, CheckInDate, CheckOutDate, CustomerID, CustomerFirstName, CustomerLastName, CustomerEmail, HotelID, HotelName, HotelCity, RoomNumber, RoomType, BasePrice)

- **Issues:**
    - Customer details (CustomerFirstName, CustomerLastName, CustomerEmail) are repeated for every booking made by the same customer.
    - Hotel details (HotelName, HotelCity) are repeated for every booking at that hotel.
    - Room details (RoomType, BasePrice) are repeated.
    - This structure violates 2NF and 3NF.

Example 2: Denormalized Employee Table

A table for employees might include full departmental information. EmployeeMaster (EmployeeID, EmpFirstName, EmpLastName, HotelID, HotelName, DeptID, DeptName, Role)

- **Issues:**
    - HotelName is repeated for every employee at the same hotel. This is a transitive dependency (EmployeeID -> HotelID -> HotelName).
    - DeptName is repeated for every employee in the same department. This is a transitive dependency (EmployeeID -> DeptID -> DeptName).
    - This structure violates 3NF.

7. Normalization

Normalization is the process of organizing columns and tables in a relational database to minimize data redundancy. We will normalize the denormalized schemas up to Boyce-Codd Normal Form (BCNF).

Let's normalize the BookingsMaster table.

Initial Denormalized Table: BookingsMaster (BookingID, CheckInDate, CheckOutDate, CustomerID, CustomerFirstName, CustomerLastName, CustomerEmail, HotelID, HotelName, HotelCity, RoomNumber, RoomType, BasePrice)

Primary Key: BookingID

First Normal Form (1NF) The table is already in 1NF as it contains no repeating groups or multi-valued attributes. Each cell holds a single, atomic value.

Second Normal Form (2NF) 2NF addresses partial dependencies on a composite primary key. Since our primary key BookingID is a single attribute, there can be no partial dependencies. Therefore, the table is automatically in 2NF.

Third Normal Form (3NF) 3NF addresses transitive dependencies. A transitive dependency exists when a non-key attribute depends on another non-key attribute.

Functional Dependencies in BookingsMaster:

1. BookingID -> CheckInDate, CheckOutDate, CustomerID, HotelID, RoomNumber (This is the primary dependency)
2. CustomerID -> CustomerFirstName, CustomerLastName, CustomerEmail (Transitive Dependency)
3. HotelID -> HotelName, HotelCity (Transitive Dependency)
4. RoomNumber + HotelID -> RoomType, BasePrice (This is tricky; RoomNumber is only unique within a hotel. This points to a need for a composite key in a Room table).

Decomposition to achieve 3NF:

We break the BookingsMaster table down to eliminate these transitive dependencies.

- **Customer Table:** Create a separate table for customer information. Customer

(CustomerID, CustomerFirstName, CustomerLastName, CustomerEmail)
    - CustomerID is the Primary Key.
- **Hotel Table:** Create a table for hotel information. Hotel (HotelID, HotelName, HotelCity)
    - HotelID is the Primary Key.
- **Room Table:** Create a table for room information. Room (RoomID, RoomNumber, RoomType, BasePrice, HotelID)
    - The PK should be RoomID.
- **Bookings Table (Revised):** The original table is now lean. Booking (BookingID, CheckInDate, CheckOutDate, CustomerID, RoomID)
    - BookingID is PK.
    - CustomerID is FK referencing Customer.
    - RoomID is a FK referencing Room.

This decomposition results in a schema that is in 3NF. Data for customers and hotels is stored only once.

Boyce-Codd Normal Form (BCNF) BCNF is a stricter version of 3NF. A table is in BCNF if, for every one of its non-trivial functional dependencies X -> Y, X is a superkey of that table. The final HotelSphere schema is designed to meet BCNF by ensuring all determinants of functional dependencies are candidate keys for their respective tables.

8. ER Schema after Normalization

The final, normalized schema (in BCNF) consists of the following relations. Primary Keys are marked with (PK), and Foreign Keys with (FK).

- **Hotel**(HotelID (PK), Name, Address, City, Country, PhoneNumber, StarRating)
- **RoomType**(RoomTypeID (PK), TypeName, Description, BasePrice)
- **Room**(RoomID (PK), RoomNumber, HotelID (FK), RoomTypeID (FK), Status)
- **Customer**(CustomerID (PK), FirstName, LastName, Email, PhoneNumber, Address)
- **Booking**(BookingID (PK), CustomerID (FK), RoomID (FK), CheckInDate, CheckOutDate, TotalAmount)
- **Payment**(PaymentID (PK), BookingID (FK), PaymentDate, Amount, PaymentMethod)
- **Department**(DeptID (PK), DeptName)
- **Role**(RoleID (PK), RoleName, JobDescription)
- **Employee**(EmployeeID (PK), FirstName, LastName, SSN, HireDate, HotelID (FK), DeptID (FK), RoleID (FK))
- **Payroll**(PayrollID (PK), EmployeeID (FK), PayPeriodStartDate, PayPeriodEndDate, BaseSalary, Bonuses, Deductions)
- **Supplier**(SupplierID (PK), SupplierName, ContactPerson, PhoneNumber, Address)
- **Product**(ProductID (PK), ProductName, Description, Category)
- **PurchaseOrder**(PO_ID (PK), HotelID (FK), SupplierID (FK), OrderDate, Status)
- **OrderDetail**(PO_ID (FK), ProductID (FK), Quantity, UnitPrice)
    - Primary Key: (PO_ID, ProductID)
- **Warehouse**(WarehouseID (PK), HotelID (FK), Location)
- **Inventory**(WarehouseID (FK), ProductID (FK), Quantity, ReorderLevel)
    - Primary Key: (WarehouseID, ProductID)

9. Creation of table (DDL Scripts)

Below are the SQL Data Definition Language (DDL) scripts for creating the normalized tables in

MySQL.

```
-- Core Operations & CRM
CREATE TABLE Hotel (
    HotelID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(100),
    Country VARCHAR(100),
    PhoneNumber VARCHAR(20),
    StarRating INT CHECK (StarRating BETWEEN 1 AND 5)
);

CREATE TABLE RoomType (
    RoomTypeID INT PRIMARY KEY AUTO_INCREMENT,
    TypeName VARCHAR(50) NOT NULL,
    Description TEXT,
    BasePrice DECIMAL(10, 2) NOT NULL
);

CREATE TABLE Room (
    RoomID INT PRIMARY KEY AUTO_INCREMENT,
    RoomNumber VARCHAR(10) NOT NULL,
    HotelID INT NOT NULL,
    RoomTypeID INT NOT NULL,
    Status VARCHAR(20) DEFAULT 'Available',
    FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),
    FOREIGN KEY (RoomTypeID) REFERENCES RoomType(RoomTypeID)
);

CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    PhoneNumber VARCHAR(20),
    Address VARCHAR(255)
);

CREATE TABLE Booking (
    BookingID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT NOT NULL,
    RoomID INT NOT NULL,
    CheckInDate DATE NOT NULL,
    CheckOutDate DATE NOT NULL,
    TotalAmount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)
);
```

```
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    BookingID INT NOT NULL,
    PaymentDate DATETIME NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    PaymentMethod VARCHAR(50),
    FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)
);

-- ERP (HR & Payroll)
CREATE TABLE Department (
    DeptID INT PRIMARY KEY AUTO_INCREMENT,
    DeptName VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE Role (
    RoleID INT PRIMARY KEY AUTO_INCREMENT,
    RoleName VARCHAR(100) NOT NULL UNIQUE,
    JobDescription TEXT
);

CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    SSN VARCHAR(20) UNIQUE NOT NULL,
    HireDate DATE NOT NULL,
    HotelID INT NOT NULL,
    DeptID INT NOT NULL,
    RoleID INT NOT NULL,
    FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID),
    FOREIGN KEY (RoleID) REFERENCES Role(RoleID)
);

CREATE TABLE Payroll (
    PayrollID INT PRIMARY KEY AUTO_INCREMENT,
    EmployeeID INT NOT NULL,
    PayPeriodStartDate DATE NOT NULL,
    PayPeriodEndDate DATE NOT NULL,
    BaseSalary DECIMAL(10, 2) NOT NULL,
    Bonuses DECIMAL(10, 2),
    Deductions DECIMAL(10, 2),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

-- SCM (Procurement & Inventory)
```

```
CREATE TABLE Supplier (
    SupplierID INT PRIMARY KEY AUTO_INCREMENT,
    SupplierName VARCHAR(150) NOT NULL,
    ContactPerson VARCHAR(100),
    PhoneNumber VARCHAR(20),
    Address VARCHAR(255)
);

CREATE TABLE Product (
    ProductID INT PRIMARY KEY AUTO_INCREMENT,
    ProductName VARCHAR(100) NOT NULL,
    Description TEXT,
    Category VARCHAR(50)
);

CREATE TABLE PurchaseOrder (
    PO_ID INT PRIMARY KEY AUTO_INCREMENT,
    HotelID INT NOT NULL,
    SupplierID INT NOT NULL,
    OrderDate DATE NOT NULL,
    Status VARCHAR(50) DEFAULT 'Pending',
    FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);

CREATE TABLE OrderDetail (
    PO_ID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity > 0),
    UnitPrice DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (PO_ID, ProductID),
    FOREIGN KEY (PO_ID) REFERENCES PurchaseOrder(PO_ID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Warehouse (
    WarehouseID INT PRIMARY KEY AUTO_INCREMENT,
    HotelID INT UNIQUE NOT NULL,
    Location VARCHAR(100),
    FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID)
);

CREATE TABLE Inventory (
    WarehouseID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    ReorderLevel INT,
    PRIMARY KEY (WarehouseID, ProductID),
```

```
    FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

10. Insertion of values (DML Scripts)

Here are sample INSERT statements to populate the database with meaningful data for testing.

```
-- Insert into Hotel
INSERT INTO Hotel (Name, Address, City, Country, PhoneNumber,
StarRating) VALUES
('HotelSphere Grand', '123 Luxury Ave', 'New York', 'USA',
'212-555-0199', 5),
('HotelSphere Bay', '456 Ocean Dr', 'Miami', 'USA', '305-555-0122',
4),
('HotelSphere Central', '789 Central Park', 'London', 'UK',
'+44-20-7946-0999', 5);

-- Insert into RoomType
INSERT INTO RoomType (TypeName, Description, BasePrice) VALUES
('Standard', 'A comfortable room with standard amenities.', 150.00),
('Deluxe', 'A spacious room with premium amenities and a great view.',
250.00),
('Suite', 'A luxurious suite with a separate living area.', 400.00);

-- Insert into Room
INSERT INTO Room (RoomNumber, HotelID, RoomTypeID, Status) VALUES
('101', 1, 1, 'Available'), ('102', 1, 2, 'Available'),
('201', 2, 1, 'Occupied'), ('202', 2, 3, 'Available'),
('301', 3, 2, 'Available'), ('302', 3, 2, 'Maintenance');

-- Insert into Customer
INSERT INTO Customer (FirstName, LastName, Email, PhoneNumber,
Address) VALUES
('John', 'Smith', 'john.smith@email.com', '555-0101', '11 Acorn St'),
('Jane', 'Doe', 'jane.doe@email.com', '555-0102', '22 Birch Rd'),
('Peter', 'Jones', 'peter.jones@email.com', '555-0103', '33 Cedar
Ln');

-- Insert into Booking
INSERT INTO Booking (CustomerID, RoomID, CheckInDate, CheckOutDate,
TotalAmount) VALUES
(1, 1, '2025-09-10', '2025-09-15', 750.00),
(2, 3, '2025-09-12', '2025-09-14', 500.00),
(1, 5, '2025-10-01', '2025-10-05', 1000.00);

-- Insert into Payment
INSERT INTO Payment (BookingID, PaymentDate, Amount, PaymentMethod)
```

```
VALUES
(1, '2025-09-01', 750.00, 'Credit Card'),
(2, '2025-09-05', 500.00, 'PayPal'),
(3, '2025-09-20', 1000.00, 'Credit Card');

-- Insert into Department
INSERT INTO Department (DeptName) VALUES ('Management'), ('Front
Desk'), ('Housekeeping'), ('HR');

-- Insert into Role
INSERT INTO Role (RoleName, JobDescription) VALUES
('General Manager', 'Oversees all hotel operations.'),
('Receptionist', 'Manages guest check-ins and inquiries.'),
('Cleaner', 'Maintains room and hotel cleanliness.'),
('HR Manager', 'Manages employee relations and recruitment.');

-- Insert into Employee
INSERT INTO Employee (FirstName, LastName, SSN, HireDate, HotelID,
DeptID, RoleID) VALUES
('Alice', 'Williams', '111-22-333', '2022-01-15', 1, 1, 1),
('Bob', 'Brown', '222-33-444', '2023-03-20', 1, 2, 2),
('Charlie', 'Davis', '333-44-555', '2021-06-10', 2, 2, 2),
('Diana', 'Miller', '444-55-666', '2023-08-01', 3, 3, 3);

-- Insert into Payroll
INSERT INTO Payroll (EmployeeID, PayPeriodStartDate, PayPeriodEndDate,
BaseSalary, Bonuses, Deductions) VALUES
(1, '2025-07-01', '2025-07-31', 6000.00, 500.00, 1200.00),
(2, '2025-07-01', '2025-07-31', 3000.00, 150.00, 600.00);

-- Insert into Supplier
INSERT INTO Supplier (SupplierName, ContactPerson, PhoneNumber,
Address) VALUES
('Fine Linens Inc.', 'Sarah Chen', '800-555-LINEN', '1 Textile Way'),
('Gourmet Supplies Co.', 'Mark Rivera', '800-555-FOOD', '2 Food
Plaza');

-- Insert into Product
INSERT INTO Product (ProductName, Description, Category) VALUES
('Queen Size Cotton Sheets', '300 Thread Count', 'Linens'),
('Luxury Bath Towel', '100% Egyptian Cotton', 'Toiletries'),
('Organic Coffee Beans', 'Fair trade, medium roast', 'Food &
Beverage');

-- Insert into PurchaseOrder
INSERT INTO PurchaseOrder (HotelID, SupplierID, OrderDate, Status)
VALUES
(1, 1, '2025-08-01', 'Completed'),
```

```
(2, 2, '2025-08-03', 'Pending');

-- Insert into OrderDetail
INSERT INTO OrderDetail (PO_ID, ProductID, Quantity, UnitPrice) VALUES
(1, 1, 50, 45.00),
(1, 2, 100, 15.00),
(2, 3, 20, 25.00);

-- Insert into Warehouse
INSERT INTO Warehouse (HotelID, Location) VALUES
(1, 'Basement Level A'),
(2, 'Ground Floor Storage'),
(3, 'Annex Building 1');

-- Insert into Inventory
INSERT INTO Inventory (WarehouseID, ProductID, Quantity, ReorderLevel)
VALUES
(1, 1, 150, 50),
(1, 2, 200, 100),
(2, 3, 40, 30);
```

11. Testing Queries

These SQL queries demonstrate how the HotelSphere database can be used to extract valuable business intelligence.

Query 1: Get Current Occupancy Rate for a Specific Hotel This query calculates the percentage of rooms currently occupied in a specific hotel.

```
SELECT
    h.Name AS HotelName,
    (COUNT(b.BookingID) / (SELECT COUNT(*) FROM Room WHERE HotelID =
h.HotelID)) * 100 AS OccupancyRatePercentage
FROM Hotel h
LEFT JOIN Room r ON h.HotelID = r.HotelID
LEFT JOIN Booking b ON r.RoomID = b.RoomID
WHERE
    h.HotelID = 1 -- Change HotelID to check other hotels
    AND CURDATE() BETWEEN b.CheckInDate AND b.CheckOutDate
GROUP BY h.Name, h.HotelID;
```

Query 2: Find Top 5 Customers by Total Spending This query identifies the most valuable customers based on their total payment amount.

```
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    SUM(p.Amount) AS TotalSpent
FROM Customer c
```

```
JOIN Booking b ON c.CustomerID = b.CustomerID
JOIN Payment p ON b.BookingID = p.BookingID
GROUP BY c.CustomerID, c.FirstName, c.LastName
ORDER BY TotalSpent DESC
LIMIT 5;
```

Query 3: Identify Supply Chain Gaps (Products Below Reorder Level) This query helps managers identify which products need to be reordered across all hotels.

```
SELECT
    h.Name AS HotelName,
    p.ProductName,
    i.Quantity AS CurrentQuantity,
    i.ReorderLevel
FROM Inventory i
JOIN Product p ON i.ProductID = p.ProductID
JOIN Warehouse w ON i.WarehouseID = w.WarehouseID
JOIN Hotel h ON w.HotelID = h.HotelID
WHERE i.Quantity < i.ReorderLevel;
```

Query 4: Monthly Booking Trends for the Entire Chain This query provides a report on the number of bookings per month, which is crucial for analyzing seasonal trends.

```
SELECT
    DATE_FORMAT(CheckInDate, '%Y-%m') AS BookingMonth,
    COUNT(BookingID) AS NumberOfBookings
FROM Booking
GROUP BY BookingMonth
ORDER BY BookingMonth;
```

Query 5: Employee List by Department for a Specific Hotel

```
SELECT
    e.FirstName,
    e.LastName,
    r.RoleName,
    d.DeptName
FROM Employee e
JOIN Department d ON e.DeptID = d.DeptID
JOIN Role r ON e.RoleID = r.RoleID
WHERE e.HotelID = 1 -- Change HotelID for different hotel branches
ORDER BY d.DeptName, e.LastName;
```

12. Conclusion

The HotelSphere project successfully addressed the critical need for a unified and efficient data management system for a hotel chain. By designing and implementing a centralized SQL database, we have created a single source of truth that integrates CRM, ERP, and SCM functionalities. The core of the solution is a robust relational model, normalized to BCNF, which guarantees data integrity, minimizes redundancy, and prevents modification anomalies that

plague denormalized systems.
The structured approach, from problem statement analysis to entity identification, normalization, and final implementation with DDL/DML scripts, has resulted in a database that is both powerful and scalable. The system is capable of handling complex operations, from guest bookings and employee payroll to supply chain management and inventory control.
The development of analytical queries demonstrates the significant business value of this integrated approach. Management can now easily access key performance indicators, such as occupancy trends, top customer metrics, and supply chain status, enabling data-driven strategic decisions. In conclusion, HotelSphere provides a solid technological foundation that can streamline operations, enhance customer service, and support the future growth of the hotel chain.

13. Outcome

The successful completion of the HotelSphere project has yielded the following key outcomes and deliverables:

- **A Fully Normalized Relational Database:** An enterprise-grade database schema normalized up to BCNF, ensuring high data integrity and minimal redundancy.
- **Integrated Management System:** A single, cohesive system that seamlessly combines modules for Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), and Supply Chain Management (SCM).
- **Complete DDL and DML Scripts:** A full set of CREATE TABLE scripts for easy deployment and INSERT scripts with sample data for testing and demonstration purposes.
- **Business Intelligence Capabilities:** A suite of powerful SQL queries designed to perform business analytics, providing insights into occupancy trends, top customers, payroll expenses, and supply chain efficiency.
- **Scalable Architecture:** A robust database architecture that can easily scale to accommodate an increasing number of hotels, customers, and transactions as the business grows.
- **Improved Operational Efficiency:** The centralized system eliminates data silos, automates data retrieval, and provides a holistic view of the business, significantly improving the efficiency of day-to-day operations.