针对你选择的**酒店管理系统**三级项目报告，我将根据报告要求详细解答每一部分的内容。

一、需求描述（满分2分）

1.1 系统需求信息

功能需求描述：

酒店管理系统需要为酒店提供全面的管理服务，包括但不限于客户信息管理、房间管理、预订管理、入住管理、退房管理、账单管理和员工管理等功能。

数据处理需求描述：

- 用户类型：
    a. **普通用户（客户）**：客户可以注册、登录，查看酒店的房间类型、价格等信息，进行房间预订和查询预订状态。
    b. **管理员**：管理员可以查看、添加、编辑、删除房间信息，管理用户预订情况，处理客户入住、退房操作，查看账单及员工管理等。

实体及属性：

1. 用户（**User**）
    - 用户ID（user_id）
    - 姓名（name）
    - 手机号（phone）
    - 邮箱（email）
    - 密码（password）
2. 房间（**Room**）
    - 房间ID（room_id）
    - 房间类型（room_type）
    - 房间状态（room_status）
    - 价格（price）
3. 预订（**Booking**）
    - 预订ID（booking_id）
    - 用户ID（user_id）
    - 房间ID（room_id）

- 预订日期（booking_date）
- 入住日期（check_in_date）
- 退房日期（check_out_date）
- 总价（total_price）

4. 账单（**Invoice**）
    - 账单ID（invoice_id）
    - 预订ID（booking_id）
    - 总金额（total_amount）
    - 支付方式（payment_method）
    - 支付状态（payment_status）
5. 员工（**Staff**）
    - 员工ID（staff_id）
    - 姓名（staff_name）
    - 职位（staff_position）
    - 工资（salary）

1.2. 数据库功能需求描述

- 系统需要处理用户的注册、登录、查询、预订等操作。
- 管理员需要查看房间信息、编辑房间状态、管理用户预订情况、生成账单、员工管理等。

二、概念结构设计（满分3分）

1. E-R图

E-R图是数据库概念结构设计的重要组成部分。基于上述需求，可以绘制如下E-R图：

- **实体：** 用户（User）、房间（Room）、预订（Booking）、账单（Invoice）、员工（Staff）
- **联系：**
    - 用户与预订是1:n（一个用户可以有多个预订）。
    - 房间与预订是1:n（一个房间可以被多个用户预订，前提是房间未被占用）。

- 预订与账单是1:1（每个预订有一张账单）。
- 员工与预订无直接联系，但管理员可管理预订。

2. 联系类型

- 用户与预订是1:n。
- 房间与预订是1:n。
- 预订与账单是1:1。

三、逻辑结构设计（满分3分）

1. 关系模式（表结构）

根据E-R图将其转换为关系模式，以下是相应的关系模式：

用户表（**User**）

```
User(user_id, name, phone, email, password)
```

- 主键：user_id

房间表（**Room**）

```
Room(room_id, room_type, room_status, price)
```

- 主键：room_id

预订表（**Booking**）

```
Booking(booking_id, user_id, room_id, booking_date, check_in_date,
check_out_date, total_price)
```

- 主键：booking_id
- 外键：user_id引用User表，room_id引用Room表

账单表（**Invoice**）

```
Invoice(invoice_id, booking_id, total_amount, payment_method,
payment_status)
```

- 主键：invoice_id
- 外键：booking_id引用Booking表

员工表（**Staff**）

```
Staff(staff_id, staff_name, staff_position, salary)
```

- 主键：staff_id

2. 规范化

根据需求，可以进一步确保所有关系模式至少符合第三范式（3NF）。例如，`预订表`（Booking）中的房间ID和用户ID分别作为外键，避免了数据冗余。

四、物理结构设计（满分1分）

1. 表结构设计

以“用户表”为例，设计具体的物理结构：

用户表（**User**）

```
CREATE TABLE User (
    user_id INT PRIMARY KEY,          -- 用户ID，主键
    name VARCHAR(100) NOT NULL,       -- 用户姓名
    phone VARCHAR(20) NOT NULL,       -- 用户手机号
    email VARCHAR(100) NOT NULL,      -- 用户邮箱
    password VARCHAR(255) NOT NULL    -- 用户密码
);
```

房间表（**Room**）

```sql
CREATE TABLE Room (
    room_id INT PRIMARY KEY,          -- 房间ID，主键
    room_type VARCHAR(50) NOT NULL,   -- 房间类型（如单人间、双人间等）
    room_status VARCHAR(20) NOT NULL, -- 房间状态（如空闲、已预订、维修中等）
    price DECIMAL(10, 2) NOT NULL     -- 房间价格
);
```

预订表（**Booking**）

```sql
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,       -- 预订ID，主键
    user_id INT NOT NULL,             -- 用户ID，外键，引用User表
    room_id INT NOT NULL,             -- 房间ID，外键，引用Room表
    booking_date DATE NOT NULL,       -- 预订日期
    check_in_date DATE NOT NULL,      -- 入住日期
    check_out_date DATE NOT NULL,     -- 退房日期
    total_price DECIMAL(10, 2) NOT NULL, -- 总价
    FOREIGN KEY (user_id) REFERENCES User(user_id),  -- 外键约束，引用User表
    FOREIGN KEY (room_id) REFERENCES Room(room_id)   -- 外键约束，引用Room表
);
```

账单表（Invoice）

```sql
CREATE TABLE Invoice (
    invoice_id INT PRIMARY KEY,       -- 账单ID，主键
    booking_id INT NOT NULL,          -- 预订ID，外键，引用Booking表
    total_amount DECIMAL(10, 2) NOT NULL, -- 总金额
    payment_method VARCHAR(50) NOT NULL, -- 支付方式（如现金、信用卡等）
    payment_status VARCHAR(20) NOT NULL, -- 支付状态（如已支付、未支付等）
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)  -- 外键约束，引用Booking表
);
```

员工表（Staff）

```
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY,           -- 员工ID，主键
    staff_name VARCHAR(100) NOT NULL,    -- 员工姓名
    staff_position VARCHAR(50) NOT NULL, -- 员工职位
    salary DECIMAL(10, 2) NOT NULL       -- 员工工资
);
```

2. 视图和索引设计

创建视图以便于查询用户预订情况：

预订视图（**Booking_View**）

```
CREATE VIEW Booking_View AS
SELECT User.name, Room.room_type, Booking.booking_date,
Booking.check_in_date, Booking.check_out_date
FROM Booking
JOIN User ON Booking.user_id = User.user_id
JOIN Room ON Booking.room_id = Room.room_id;
```

创建索引以加速查询：

```
CREATE INDEX idx_user_id ON Booking(user_id);
CREATE INDEX idx_room_id ON Booking(room_id);
```

五、数据库实施（满分2分）

1. SQL命令实现

根据设计的表结构和关系，使用SQL创建数据库和表、视图、索引等。例如：

```sql
CREATE DATABASE HotelDB;

USE HotelDB;

CREATE TABLE User (
    user_id INT(11) PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    email VARCHAR(100),
    password VARCHAR(255) NOT NULL
);

-- 创建其他表、视图和索引
```

六、数据库运行（满分8分）

根据系统的功能需求，模拟用户操作，展示以下SQL语句：

1. 用户注册：

```sql
INSERT INTO User (name, phone, email, password)
VALUES ('张三', '1234567890', 'zhangsan@example.com',
'password123');
```

2. 查询房间：

```sql
SELECT * FROM Room WHERE room_status = 'Available';
```

3. 用户预订房间：

```sql
INSERT INTO Booking (user_id, room_id, booking_date, check_in_date,
check_out_date, total_price)
VALUES (1, 2, '2025-05-25', '2025-06-01', '2025-06-07', 1000.00);
```

4. 查看预订详情：

```sql
SELECT * FROM Booking_View WHERE user_id = 1;
```

5. 生成账单：

```sql
INSERT INTO Invoice (booking_id, total_amount, payment_method,
payment_status)
VALUES (1, 1000.00, 'Credit Card', 'Paid');
```

七、数据库维护（满分1分）

1. 创建数据库用户和角色、授权和收权

```sql
-- 创建用户
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'adminpass';

-- 授权
GRANT ALL PRIVILEGES ON HotelDB.* TO 'admin'@'localhost';

-- 收回权限
REVOKE ALL PRIVILEGES ON HotelDB.* FROM 'admin'@'localhost';
```

以上是针对**酒店管理系统**三级项目报告的详细设计，涵盖了需求描述、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、数据库运行和数据库维护的各个方面。

```sql
-- 创建数据库
CREATE DATABASE HotelDB;

-- 使用数据库
USE HotelDB;

-- 创建User表
CREATE TABLE User (
    user_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '用户ID，主键，自增
长',
    name VARCHAR(100) NOT NULL COMMENT '用户姓名',
    phone VARCHAR(15) NOT NULL COMMENT '用户手机号，保证唯一',
    email VARCHAR(100) COMMENT '用户邮箱',
    password VARCHAR(255) NOT NULL COMMENT '用户密码',
    UNIQUE (phone) COMMENT '保证手机号唯一',
    UNIQUE (email) COMMENT '保证邮箱唯一'
);
```

```sql
-- 创建Room表
CREATE TABLE Room (
    room_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '房间ID，主键，自增长',
    room_type VARCHAR(50) NOT NULL COMMENT '房间类型（如单人间、双人间等）',
    room_status VARCHAR(20) NOT NULL COMMENT '房间状态（如空闲、已预订、维修中等）',
    price DECIMAL(10, 2) NOT NULL COMMENT '房间价格'
    -- 可选：如果你需要房间类型和状态唯一，可以保留以下约束
    -- UNIQUE (room_type, room_status) COMMENT '保证同类型房间的多重状态唯一'
);

-- 创建Booking表
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '预订ID，主键，自增长',
    user_id INT NOT NULL COMMENT '用户ID，外键，引用User表',
    room_id INT NOT NULL COMMENT '房间ID，外键，引用Room表',
    booking_date DATE NOT NULL COMMENT '预订日期',
    check_in_date DATE NOT NULL COMMENT '入住日期',
    check_out_date DATE NOT NULL COMMENT '退房日期',
    total_price DECIMAL(10, 2) NOT NULL COMMENT '总价',
    CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES User(user_id),
    CONSTRAINT fk_room FOREIGN KEY (room_id) REFERENCES Room(room_id)
);

-- 创建Invoice表
CREATE TABLE Invoice (
    invoice_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '账单ID，主键，自增长',
    booking_id INT NOT NULL COMMENT '预订ID，外键，引用Booking表',
    total_amount DECIMAL(10, 2) NOT NULL COMMENT '总金额',
    payment_method VARCHAR(50) NOT NULL COMMENT '支付方式（如现金、信用卡等）',
    payment_status VARCHAR(20) NOT NULL COMMENT '支付状态（如已支付、未支付等）',
```

```
    CONSTRAINT fk_booking FOREIGN KEY (booking_id) REFERENCES
Booking(booking_id)
);

-- 创建Staff表
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY AUTO_INCREMENT COMMENT '员工ID，主键，自
增长',
    staff_name VARCHAR(100) NOT NULL COMMENT '员工姓名',
    staff_position VARCHAR(50) NOT NULL COMMENT '员工职位',
    salary DECIMAL(10, 2) NOT NULL COMMENT '员工工资'
);

-- 创建Booking_View视图
CREATE VIEW Booking_View AS
SELECT
    User.name AS user_name,
    Room.room_type,
    Booking.booking_date,
    Booking.check_in_date,
    Booking.check_out_date
FROM Booking
JOIN User ON Booking.user_id = User.user_id
JOIN Room ON Booking.room_id = Room.room_id;

-- 创建索引
CREATE INDEX idx_user_id ON Booking(user_id);
CREATE INDEX idx_room_id ON Booking(room_id);
```

以下是模拟用户对数据库进行操作的一些常见增删改查 SQL 语句，包含至少10个知识点，涵盖单表查询、分组查询、排序查询、条件查询、子查询、多表连接查询、添加数据、修改数据、删除数据以及查询视图等。

1. 单表查询：查询所有用户的基本信息

```
SELECT * FROM User;
```

这个查询展示了`User`表中的所有字段数据。

---

2. 条件查询：查询所有已支付的账单

```sql
SELECT * FROM Invoice WHERE payment_status = '已支付';
```

此查询返回所有 `Invoice` 表中支付状态为 `已支付` 的账单。

---

3. 排序查询：查询按房间价格升序排列的所有房间

```sql
SELECT * FROM Room ORDER BY price ASC;
```

查询结果将按 `price` 字段升序排列房间。

---

4. 分组查询：查询每个房间状态下的房间数量

```sql
SELECT room_status, COUNT(*) AS room_count
FROM Room
GROUP BY room_status;
```

此查询统计了 `Room` 表中每个房间状态的房间数量，使用了 `GROUP BY` 进行分组。

---

5. 条件查询和排序：查询入住日期为 2025 年 6 月的预订，并按退房日期降序排列

```sql
SELECT * FROM Booking
WHERE check_in_date BETWEEN '2025-06-01' AND '2025-06-30'
ORDER BY check_out_date DESC;
```

此查询查询了 2025 年 6 月内的所有预订，并按 `check_out_date` 降序排序。

---

6. 子查询：查询没有支付的账单金额

```sql
SELECT total_amount
FROM Invoice
WHERE payment_status = '未支付'
AND invoice_id NOT IN (
    SELECT invoice_id
    FROM Payment
    WHERE payment_status = '已支付'
);
```

通过子查询，查询 `Invoice` 表中状态为 `未支付` 的账单金额，且这些账单不在已经支付的账单中。

---

7. 多表连接查询：查询每个用户的预订信息

```sql
SELECT
    User.name AS user_name,
    Room.room_type,
    Booking.booking_date,
    Booking.check_in_date,
    Booking.check_out_date
FROM Booking
JOIN User ON Booking.user_id = User.user_id
JOIN Room ON Booking.room_id = Room.room_id;
```

通过多表连接查询（`JOIN`），我们获得了每个用户的预订信息，连接了 `Booking`、`User` 和 `Room` 表。

---

8. 添加数据：插入一个新的用户记录

```sql
INSERT INTO User (name, phone, email, password)
VALUES ('张三', '12345678901', 'zhangsan@example.com',
'password123');
```

这个 SQL 语句向 `User` 表插入了一条新的用户记录。

---

9. 修改数据：更新某个用户的邮箱

```sql
UPDATE User
SET email = 'newemail@example.com'
WHERE user_id = 1;
```

该语句更新了 `user_id` 为 1 的用户的邮箱地址。

---

10. 删除数据：删除某个预订记录

```sql
DELETE FROM Booking
WHERE booking_id = 10;
```

该语句删除了 `booking_id` 为 10 的预订记录。

---

11. 查询视图：查询所有用户的预订信息（使用视图）

```sql
SELECT * FROM Booking_View;
```

这里使用了先前创建的 `Booking_View` 视图，查询了所有用户的预订信息。视图简化了多表连接查询的操作。

---

12. 联合查询：查询所有用户的预订记录和对应的账单

```sql
SELECT User.name AS user_name, Booking.booking_date,
Invoice.total_amount
FROM User
JOIN Booking ON User.user_id = Booking.user_id
JOIN Invoice ON Booking.booking_id = Invoice.booking_id;
```

此查询通过联合 `User`、`Booking` 和 `Invoice` 三张表来获取每个用户的预订日期和账单金额。

---

13. 条件查询和计数：查询每个房间类型的预订数量

```sql
SELECT Room.room_type, COUNT(Booking.booking_id) AS booking_count
FROM Room
JOIN Booking ON Room.room_id = Booking.room_id
GROUP BY Room.room_type;
```

此查询显示每个房间类型的预订数量，使用了 `JOIN` 和 `GROUP BY`。

---

14. 修改数据：更改房间的价格

```sql
UPDATE Room
SET price = 350.00
WHERE room_id = 5;
```

这个查询更新了房间 ID 为 5 的房间价格。

这些 SQL 语句涵盖了多种常见的操作，包括数据查询、插入、更新和删除，以及多表连接、视图查询和子查询等。你可以根据实际需求进行调整。