hdjd

课程设计（论文）

**课程所属学院 信息与软件工程学院**

**所属课程名称 综合课程设计II**

**项目 名称 旅游预定平台**

**院 （系） 软件工程**

**班 级 软件工程2023-3班**

**学 生 张诗佳**

**指 导 教 师 王 珏**

**辅 导 教 师**

2024年 12月23日

hdjd课程设计(论文)任务书

**信息与软件工程**　学院 **软件工程** 专业**2023-3**班

一、课程设计(论文)题目 (按实际选题名称写)

二、课程设计(论文)工作自**2024**年 **12**月**23**日起至**2025**年**1**月**2**日止。

三、课程设计(论文) 地点: 软件测试室

四、课程设计(论文)内容要求：

**1．本课程设计的目的**

1）全面学习课程所学理论知识，巩固所学知识重点和难点，将理论和实践很好地结合起来。

2）提高综合运用所学知识，独立分析和解决问题的能力。

3）熟练使用一种高级语言进行编程实现。

**2．课程设计的任务及要求**

**1）基本要求：**

1、旅游景点信息管理，包括基本信息、门票价格等；

2、酒店信息管理，包括星级、地理位置、风格、价位等；

3、会员信息管理，包括基本信息、会员等级等；

4、旅游线路管理，整合旅游景点和酒店、交通线路等信息；

5、会员可以在线预定景点门票、旅游线路和酒店；

6、查询和统计，可以查询景点、酒店、会员和线路信息；

7、使用Java语言和JDBC实现。

**2）创新要求：**

1、能使用数据库的高级特性完成设计；

2、程序设计中采用设计模式。

**3）课程设计论文编写要求**

（1）要按照书稿的规格打印誊写课程设计论文；

（2）论文包括任务书、目录、绪论、正文、小结、参考文献等；

（3）课程设计论文装订按学校的统一要求完成。

**4）评分标准：**

（1）原理分析（30分）

（2）系统方案选择与实现（50分）

（3）论文写作（20分）

**5）参考文献：**

（1）http://www.docin.com/p-206666064.html

（2）丁振凡 范萍.《java语言程序设计》.清华大学出版社

**6）课程设计进度安排**

**内容 天数　　　　　　地点**

准备与动员 1　　　　　 　软件测试室

需求分析 1　　　　　 　分散

设计实现 2　　　　　 软件测试室，分散

文档撰写 1　　　　　　 分散

答辩 2　　　　　　 软件测试室

**7）诚信申明**

**本人郑重声明：所呈交的课程设计报告是我个人独立完成的，如在文中涉及抄袭或剽**

**窃行为，本人愿承担由此而造成的一切后果及责任。**

学生签名：

2024 年12月23 日

**课程设计(论文)评价标准**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | **考核内容** | **评分细则** | **课程目标** | **得分** | |
| 1 | **设计报告（20分）** | 报告格式规范，行距、字体、图、表等设置合理。使用材料翔实、恰当,掌握大量的背景资料和数据。能在问题研究中综合运用专业知识。理论分析准确，逻辑严密，层次清楚，结构合理，语言流畅。 | 1（20%） |  |  |
| 3（40%） |  |
| 5（40%） |  |
| 2 | **系统功能设计**  **（20分）** | 理论准备充分，系统功能完善，界面友好，操作简便  设计合理，系统运行很稳定，很少出现错误、死机现象。 | 2（50%） |  |  |
| 3（50%） |  |
| 3 | **编码质量（30分）** | 运行结果正确，逻辑清晰；有良好的编码风格与编程习惯，文档和注释合理，异常处理设计合理。模块划分层次清楚，可移植性好。 | 2（50%） |  |  |
| 4（50%） |  |
| 4 | **平时表现（20分）** | 其中考勤考核3次，缺勤一次扣5分，请假、迟到，早退一次扣2分。 | 1（50%） |  |  |
| 3（50%） |  |
| 4 | **答 辩（10分）** | 能切中要点，回答问题准确；条理清晰、重点突出、语言简练；数据科学、客观、详实。专业术语运用准确。 | 5（100%） |  | |
| **总 得 分** | | | |  | |



评阅人：

2025年1月5日

**目录**

[第一章 数据设计 - 1 -](#_Toc9010)

[1.1实体 - 1 -](#_Toc27143)

[1.2联系 - 3 -](#_Toc23333)

[第二章 数据库设计 - 4 -](#_Toc3301)

[2.1关系模式 - 4 -](#_Toc9864)

[2.2数据表 - 4 -](#_Toc21095)

[第三章 数据库实现 - 7 -](#_Toc5156)

[3.1数据库操作 - 7 -](#_Toc12246)

[3.2 数据操作 - 7 -](#_Toc26419)

[第四章 程序设计 - 10 -](#_Toc23088)

[4.1程序功能列表 - 10 -](#_Toc19623)

[4.2 程序设计概述 - 10 -](#_Toc3816)

[4.3类和接口 - 10 -](#_Toc18277)

[4.4类和接口设计 - 11 -](#_Toc31647)

[第五章 程序实现 - 17 -](#_Toc29585)

[5.1类和接口代码 - 17 -](#_Toc1224)

[5.2 界面代码 - 34 -](#_Toc299)

[第六章 运行结果 - 45 -](#_Toc27224)

[6.1（旅游景点信息管理） - 45 -](#_Toc2177)

[6.2（酒店信息管理） - 47 -](#_Toc742)

[6.3（会员信息管理） - 48 -](#_Toc12101)

[6.4（线路信息查询） - 51 -](#_Toc18672)

[6.5（会员预定查询） - 53 -](#_Toc2890)

[6.6（最受欢迎旅游线路统计） - 54 -](#_Toc4892)

[6.7（门票预定统计） - 54 -](#_Toc22581)

[第七章 设计小结 - 56 -](#_Toc22446)

[7.1数据库设计小结 - 56 -](#_Toc32676)

[7.2 Java程序设计小结 - 56 -](#_Toc7312)

[7.3 需完善之处 - 57 -](#_Toc12182)

# 第一章 数据设计

## 1.1实体

1. 旅游景点：用于存储旅游景点相关信息的实体。

* 景点id：一个自增的整数类型属性，从 1 开始，每次递增 1，作为该实体的主键，能够唯一地标识每一个景点
* 景点名称：字符串类型，最大长度为20个字符
* 门票价格：整数类型
* 门票数量：整数类型
* 开放时间：date类型
* 地理位置：字符串类型，最大长度为50个字符

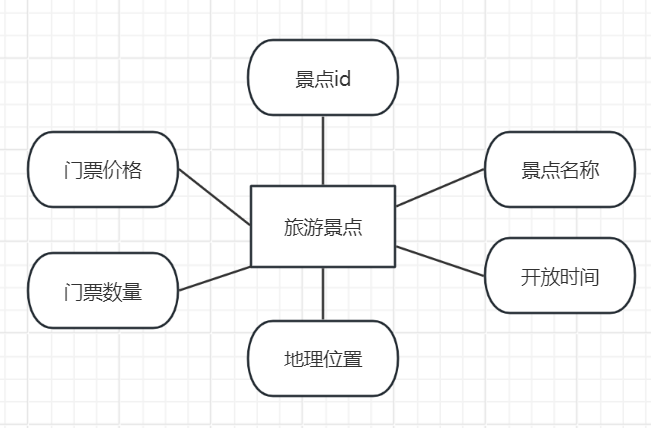


图1-1 旅游景点 E-R图

1. 酒店：用于存储酒店相关信息的实体。

* 酒店id：一个自增的整数类型属性，从 1 开始，每次递增 1，作为该实体的主键，能够唯一地标识每一个酒店
* 酒店名称：字符串类型，最大长度为20个字符
* 星级：整数类型
* 风格：字符串类型，最大长度为20个字符
* 价位：整数类型
* 地理位置：字符串类型，最大长度为50个字符

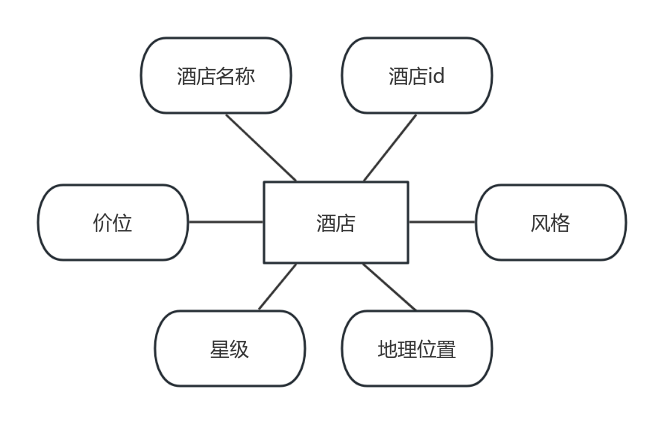


图1-2酒店 E-R图

3）会员：用于存储酒店相关信息的实体。

* 会员id：一个自增的整数类型属性，从 1 开始，每次递增 1，作为该实体的主键，能够唯一地标识每一个会员信息
* 会员姓名：字符串类型
* 会员等级：字符串类型
* 联系方式：字符串类型
* 注册时间：date类型

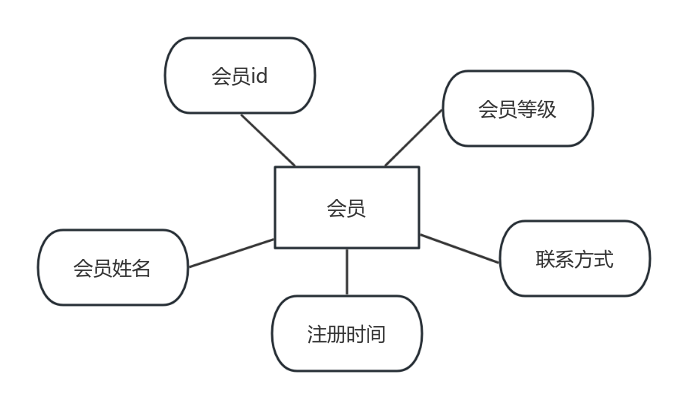


图1-3 会员 E-R图

4）旅游线路：用于存储旅游线路相关信息的实体。

* 线路名称：字符串类型
* 旅游线路编号：一个自增的整数类型属性，从 1 开始，每次递增 1，作为该实体的主键，能够唯一地标识每一条旅游线路



图1-4 旅游线路 E-R图

5）交通线路：用于存储交通线路相关信息的实体。

* 交通线路编号：一个自增的整数类型属性，从 1 开始，每次递增 1，作为该实体的主键，能够唯一地标识每一条交通线路
* 交通方式：字符串类型



图1-5 交通线路 E-R图

## 1.2联系

本设计中实体之间的联系如下：

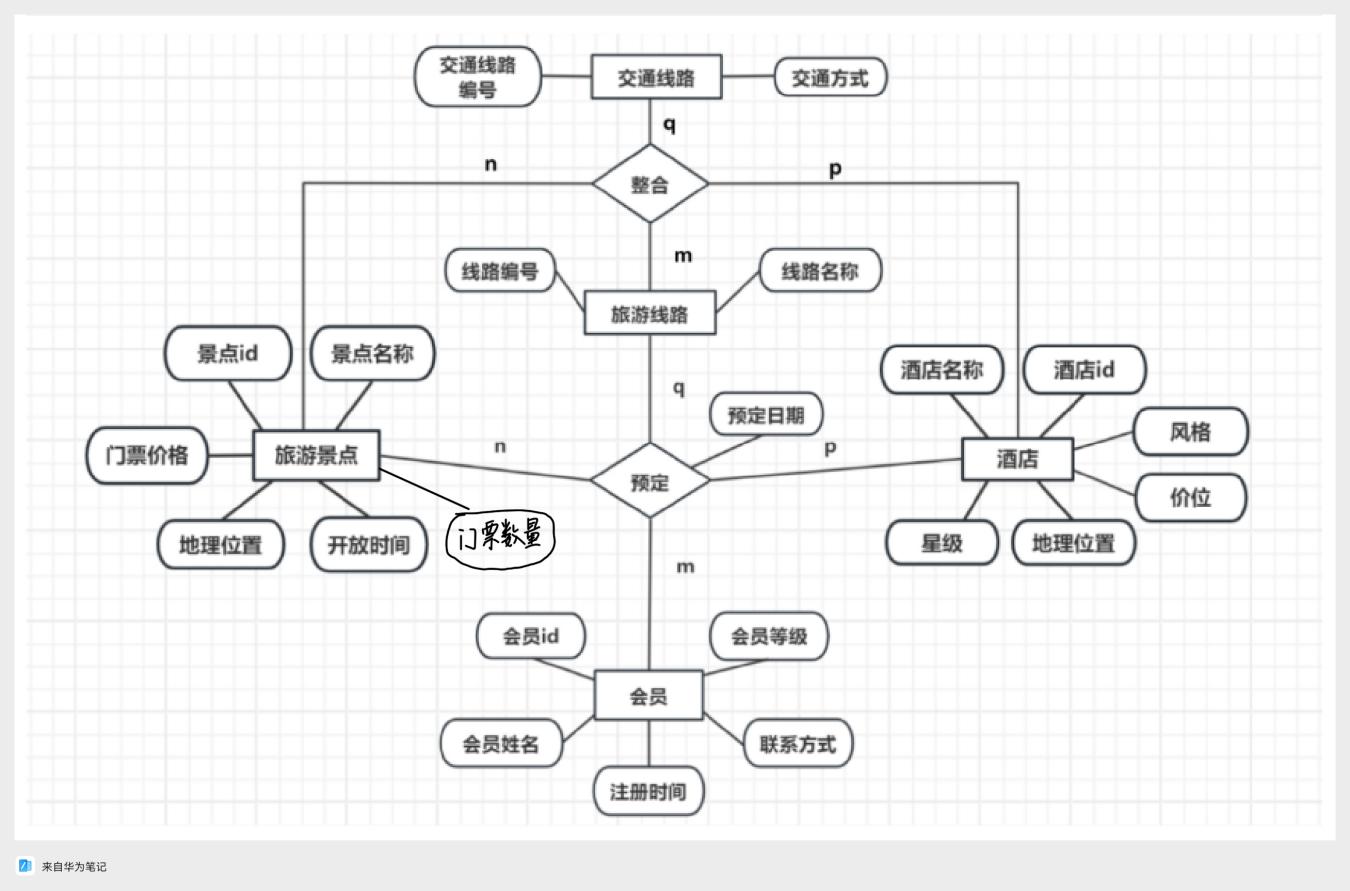


图1-6 总体联系E-R图

# 第二章 数据库设计

## 2.1关系模式

本设计中的关系模式如下：

**旅游景点** (景点id,景点名称,门票价格,门票数量，开放时间,地理位置)。关系的主键：景点id；外键：无。

**酒店**（酒店id，酒店名称，星级，风格，价位，地理位置）。关系的主键：酒店id；外键：无。

**会员**（会员id，会员姓名，会员等级，联系方式，注册时间）。关系的主键：会员id；外键：无。

**旅游线路**（线路名称，旅游线路编号）。关系的主键：旅游线路编号；外键：无。

**交通线路**（交通方式，交通线路编号）。关系的主键：交通线路编号；外键：无。

**预定**（会员id，景点id，酒店id，旅游线路编号，预定日期）。关系的主键：会员id，景点id，酒店id，旅游线路编号。外键：会员id，景点id，酒店id，旅游线路编号。

**旅游线路管理**（酒店id，景点id，旅游线路编号，交通线路编号）。关系的主键：酒店id，景点id，旅游线路编号，交通线路编号。外键：酒店id，景点id，旅游线路编号，交通线路编号。

## 2.2数据表

本设计中创建的数据库名为：**TravelBookingDB**，数据表分别为：

表2-1 TourSpot

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Tid | int | NO |
| Tname | varchar(20) | YES |
| Tprice | smallint | YES |
| Tlocation | varchar(50) | YES |
| Tdate | date | YES |
| Total\_tickets | int | YES |

表2-2 Hotel

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Hid | int | NO |
| Hname | Varchar(20) | YES |
| Hprice | smallint | YES |
| Hstyle | Varchar(20) | YES |
| Hstarrating | Smallint | YES |
| Hlocation | Varchar(50) | YES |

表2-3 Member

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Mid | int | NO |
| Mname | varchar(20) | YES |
| Mphone | Char(11) | YES |
| Mdate | date | YES |
| Mlevel | Char(10) | YES |

表2-4 Route

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Rid | int | NO |
| Rname | varchar(20) | YES |

表2-5 Rtraffic

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Rtrafficid | int | NO |
| Rtrafficway | varchar(20) | YES |

表2-6 Booking

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Mid | int | NO |
| Tid | int | NO |
| Rid | int | NO |
| Hid | int | NO |
| Bdate | date | YES |

表2-7 Routemanage

|  |  |  |
| --- | --- | --- |
| **字段名（列名）** | **数据类型** | **允许NULL** |
| Rtrafficid | int | NO |
| Tid | int | NO |
| Rid | int | NO |
| Hid | int | NO |

数据库关系图为：

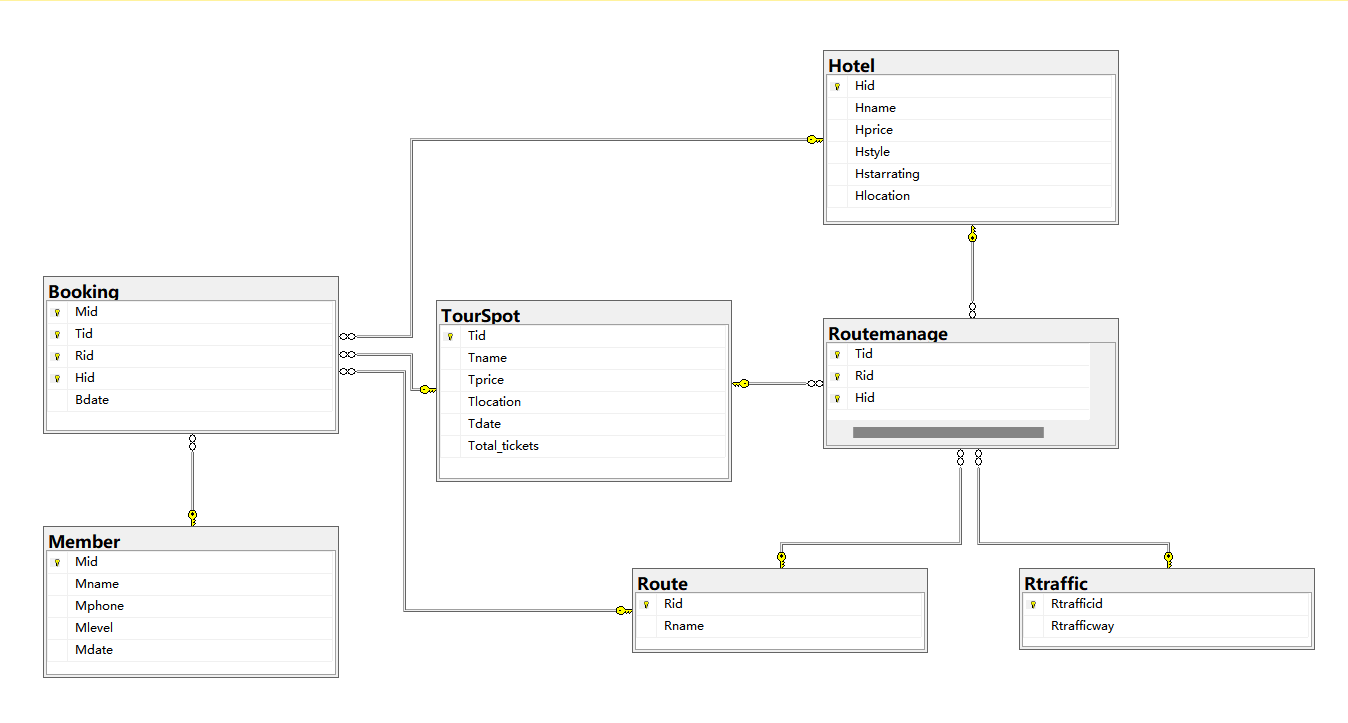


图2-8 数据库关系图

# 第三章 数据库实现

## 3.1数据库操作

系统中与数据库操作有关的功能如下：

1. 新建旅游景点表：新建一个包含景点id,景点名称,门票价格,开放时间,地理位置的旅游景点表。关系的主键为景点id，有关的数据表为：TourSpot
2. 旅游景点信息管理：对旅游景点表中的数据进行增删改查操作，有关的数据表为：TourSpot
3. 酒店信息管理：对酒店表中的数据进行增删改查操作，有关的数据表为：Hotel
4. 会员信息管理：对酒店表中的数据进行增删改查操作，有关的数据表为：Member
5. 线路信息查询：按照旅游线路编号从旅游线路管理表中查询基本线路信息，有关的数据表为：Routemanage,TourSpot,Hotel,Rtraffic
6. 会员预定查询：按照会员id查询预定的酒店，旅游景点和旅游线路信息，有关的数据表为：Booking,TourSpot,Hotel,Route
7. 旅游线路统计：统计最受欢迎的旅游线路，有关的数据表为：Booking
8. 门票预定统计：景点门票被预定后，自动更新门票数量，有关的数据表为：Booking，TourSpot

## 3.2 数据操作

上述功能对应操作的SQL语句为：

1. 新建旅游景点表：

create table TourSpot(

Tid int identity(1,1) primary key,

Tname varchar(20),

Tprice smallint,

Tlocation varchar(50),

Tdate date)

1. 旅游景点信息管理：

增加数据：INSERT INTO TourSpot (Tname, Tprice, Tlocation, Tdate)

VALUES

('Eiffel Tower', 30, 'Paris, France', '2024-01-01'),

('Great Wall of China', 50, 'Beijing, China', '2024-02-15'),

('Colosseum', 25, 'Rome, Italy', '2024-03-10'),

('Machu Picchu', 75, 'Cusco, Peru', '2024-04-05'),

('Mount Fuji', 40, 'Honshu, Japan', '2024-05-20');

删除数据：delete from TourSpot where Tid=1;

修改数据：update TourSpot Set Tprice=60 Where Tid=2

查询数据：select \* from TourSpot

1. 酒店信息管理：

增加数据：INSERT INTO Hotel (Hname, Hprice, Hstyle, Hstarrating, Hlocation)

VALUES

('Hotel Sunshine', 120, 'Luxury', 5, 'Downtown'),

('Hotel Ocean', 80, 'Budget', 3, 'Beachside'),

('Hotel Mountain View', 150, 'Resort', 4, 'Mountain Range');

('Hotel City Center', 100, 'Business', 4, 'City Center'),

('Hotel Riverside', 130, 'Boutique', 5, 'Riverside');

删除数据：delete from Hotel where Hid=1;

修改数据：update Hotel Set Hprice=280 Where Hid=2

查询数据：select \* from TourSpot

1. 会员信息管理：

增加数据：INSERT INTO Member (Mname, Mphone, Mlevel, Mdate)

VALUES

('张三', '13812345678', '普通', '2024-01-01'),

('李四', '13987654321', 'SVIP', '2023-12-15'),

('王五', '13612340987', '普通', '2024-05-10'),

('赵六', '13523456789', '高级', '2024-07-20'),

('孙七', '13434567890', 'SVIP', '2023-11-30');

删除数据：delete from Member where Mid=5;

修改数据：update Member Set Mlevel=’SVIP’ Where Mid=2

查询数据：select \* from Member

5)线路信息查询：

SELECT Routemanage.Rid, TourSpot.Tname, Hotel.Hname, Rtraffic.Rtrafficway

FROM Routemanage

JOIN TourSpot ON TourSpot.Tid = Routemanage.Tid

JOIN Hotel ON Hotel.Hid = Routemanage.Hid

JOIN Rtraffic ON Rtraffic.Rtrafficid = Routemanage.Rtrafficid

WHERE Routemanage.Rid = 1;

1. 会员预定查询：

SELECT \* FROM Booking

JOIN TourSpot ON TourSpot.Tid = Booking.Tid

JOIN Member ON Member.Mid = Booking.Mid

JOIN Hotel ON Hotel.Hid = Booking.Hid

WHERE Booking.Mid = 1;

1. 最受欢迎旅游线路统计：

SELECT TOP 1 Rid, COUNT(\*) AS booking\_count

FROM Booking

GROUP BY Rid

ORDER BY booking\_count DESC

1. 门票预定统计：

CREATE TRIGGER UpdateTicketQuantity

ON Booking

AFTER INSERT

AS

BEGIN

UPDATE ta

SET ta.Total\_tickets = ta.Total\_tickets - 1

FROM TourSpot ta

INNER JOIN inserted i

ON ta.Tid= i.Tid

WHERE ta.Total\_tickets > 0;

IF EXISTS (SELECT 1

FROM TourSpot ta

INNER JOIN inserted i

ON ta.Tid= i.Tid

WHERE ta.Total\_tickets < 0)

BEGIN

ROLLBACK TRANSACTION;

PRINT '门票不足，预定失败';

END

END;

# 第四章 程序设计

## 4.1程序功能列表

表4-1 系统功能列表

|  |  |
| --- | --- |
| 功能名称 | 功能描述 |
| 旅游景点信息管理： | 对旅游景点表中的数据进行增删改查操作。 |
| 酒店信息管理： | 对酒店表中的数据进行增删改查操作。 |
| 会员信息管理： | 对会员表中的数据进行增删改查操作。 |
| 线路信息查询： | 查询旅游线路管理表的基本信息。 |
| 会员预定查询： | 根据会员id，酒店id，旅游景点id或者旅游线路id查询预定信息。 |
| 最受欢迎旅游线路统计： | 统计最受欢迎的旅游线路。 |
| 门票预定统计： | 统计旅游景点被预定后的门票数量。 |

## 4.2 程序设计概述

本程序结构是通过控制台界面和功能模块的划分来实现各项功能。程序中使用了 命令模式来处理用户输入的操作，使用了数据访问对象模式（DAO）来进行数据库操作。

## 4.3类和接口

程序中定义的Java类和接口如下：

**1.实体类**

1. TourSpot类：旅游景点实体类，包含旅游景点相关的属性；
2. Hotel类：酒店实体类，包含了酒店相关的属性；
3. Member类: 会员实体类，包含了会员相关的属性；
4. Booking类: 预订实体类，包含了预订相关的属性；
5. Route类: 旅游线路实体类，包含了旅游线路的属性；
6. Rtraffic类: 交通线路实体类，包含了交通线路的属性；
7. Routemanage类：线路管理实体类，包含与线路相关的属性；
8. **功能类**
9. BaseDAO接口，定义了对数据库实体的基本操作；
10. BookingDAO接口: 定义了多种预订查询方法；
11. TourSpotDAOImpl: 这是BaseDAO接口的实现类，实现对旅游景点信息进行增删改查的基本功能。
12. HotelDAOImpl类: BaseDAO接口的实现类，实现对酒店信息进行增删改查的基本功能。
13. MemberDAOImpl类: BaseDAO接口的实现类，实现对会员信息进行增删改查的基本功能。
14. BookingDAOImpl类: BookingDAO接口的实现类，实现多种预定查询方式的功能；
15. RoutemanageDAOImpl类: 线路管理实现类，实现线路信息查询的功能；
16. MostPopularRoute类: 实现类，实现查询最受欢迎的旅游线路的功能；
17. TicketBooking: 实现类，实现门票的预定功能，管理景点门票数量。

## 4.4类和接口设计

**1.实体类**

1) TourSpot类

public class TourSpot {  
 private int Tid;  
 private String Tname;  
 private short Tprice;  
 private String Tlocation;  
 private Date Tdate;  
 private int Total\_tickets;  
  
 // 无参构造器  
 public TourSpot();

//有参构造器

public TourSpot(int Tid, String Tname, short Tprice, String Tlocation, Date Tdate); @Override  
 public String toString() ;

}

1. Hotel类

public class Hotel {  
 private int Hid;  
 private String Hname;  
 private short Hprice;  
 private String Hstyle;  
 private short Hstarrating;  
 private String Hlocation;  
  
 // 无参构造器  
 public Hotel() ;

//有参构造器  
 public Hotel(int Hid, String Hname, short Hprice, String Hstyle, short Hstarrating, String Hlocation);

@Override

public String toString();

}

1. Member类

public class Member {  
 private int Mid;  
 private String Mname;  
 private String Mphone;  
 private String Mlevel;  
 private Date Mdate;  
  
 public Member(int Mid, String Mname, String Mphone, String Mlevel, Date Mdate);

@Override

public String toString();

}

1. Booking类

public class Booking {  
 private int Mid;  
 private int Tid;  
 private int Rid;  
 private int Hid;  
 private Date Bdate;  
  
 public Booking(int Mid, int Tid, int Rid, int Hid, Date Bdate);

@Override  
 public String toString();

}

1. Route类

public class Route {  
 private int Rid;  
 private String Rname;  
 @Override  
 public String toString() ;  
 }  
}

1. Rtraffic类

public class Rtraffic {  
 private int Rtrafficid;  
 private String Rtrafficway;  
 @Override  
 public String toString();  
}

1. Routemanage类

public class Routemanage {  
 private int Rtrafficid;  
 private int Tid;  
 private int Rid;  
 private int Hid;  
 private Rtraffic rtraffic;  
 private TourSpot tourSpot;  
 private Route route;  
 private Hotel hotel;  
 @Override  
 public String toString();

}

**2.功能类**

1. BaseDAO接口

public interface BaseDAO<T> {  
 void add(T entity);  
 void update(T entity);  
 void delete(int id);  
 List<T> findById(int id);  
}

1. BookingDAO接口

public interface BookingDAO {  
 List<Booking> findBookingsByMemberId(int memberId); // 根据会员 ID 查询预定  
 List<Booking> findBookingsByHotelId(int hotelId); // 根据酒店 ID 查询预定  
 List<Booking> findBookingsByTourSpotId(int tourSpotId); // 根据旅游景点 ID 查询预定  
 List<Booking> findBookingsByRouteId(int tourSpotId);  
}

1. TourSpotDAOImpl类

public class TourSpotDAOImpl implements BaseDAO<TourSpot> {  
//增加数据  
 @Override  
 public void add(TourSpot tourSpot);

//修改数据  
 @Override  
 public void update(TourSpot tourSpot);

//删除数据  
 @Override  
 public void delete(int id);

//查询ID  
 @Override  
 public List<TourSpot> findById(int id);

return tourSpots;  
   
}

1. HotelDAOImpl类

public class HotelDAOImpl implements BaseDAO<Hotel> {  
//增加数据  
 @Override  
 public void add(Hotel hotel);  
//修改数据  
 @Override  
 public void update(Hotel hotel);  
//删除数据  
 @Override  
 public void delete(int id);  
//查询ID  
 @Override  
 public List<Hotel> findById(int id);  
 return hotels;  
}

1. MemberDAOImpl类

public class MemberDAOImpl implements BaseDAO<Member> {  
//增加数据

@Override  
 public void add(Member member);

//修改数据

@Override  
 public void update(Member member);

//删除数据  
 @Override  
 public void delete(int id);

//查询ID  
 @Override  
 public List<Member> findById(int id);  
 return members;  
}

1. BookingDAOImpl类

public class BookingDAOImpl implements BookingDAO {   
 // 根据会员 ID 查询预定  
 @Override  
 public List<Booking> findBookingsByMemberId(int memberId);  
 return bookings;  
   
 // 根据酒店 ID 查询预定  
 @Override  
 public List<Booking> findBookingsByHotelId(int hotelId);  
 return bookings;  
  
// 根据旅游景点ID 查询预定  
 @Override  
 public List<Booking> findBookingsByTourSpotId(int tourSpotId); return bookings;

// 根据旅游线路ID 查询预定  
 public List<Booking> findBookingsByRouteId(int routeId);  
 return bookings;  
}

1. RoutemanageDAOImpl类

public class RoutemanageDAOImpl {  
 //查询旅游线路  
 public List<Routemanage> findAllRoutemanageWithDetails(); return routemanages;  
}

1. MostPopularRoute类

public class MostPopularRoute {  
 //统计最受欢迎路线  
 public static void getMostPopularRoute(); }

1. TicketBooking类

public class TicketBooking {  
 //统计预定门票数量  
 public static void bookTicket(int tid, int quantity); }

# 第五章 程序实现

## 5.1类和接口代码

**1.实体类**

1. TourSpot类

public class TourSpot {  
 private int Tid;  
 private String Tname;  
 private short Tprice;  
 private String Tlocation;  
 private Date Tdate;  
 private int Total\_tickets;  
  
 // 无参构造器  
 public TourSpot() {  
 this.Tid = Tid;  
 this.Tname = Tname;  
 this.Tprice = Tprice;  
 this.Tlocation = Tlocation;  
 this.Tdate = Tdate;  
 }

//有参构造器

public TourSpot(int Tid, String Tname, short Tprice, String Tlocation, Date Tdate) {

this.Tid = Tid;

this.Tname = Tname;

this.Tprice = Tprice;

this.Tlocation = Tlocation;

this.Tdate = Tdate;

}  
 @Override  
 public String toString() {  
 return "TourSpot{" +  
 "Tid=" + Tid +  
 ", Tname='" + Tname +   
 ", Tprice=" + Tprice +  
 ", Tlocation='" + Tlocation +   
 ", Tdate=" + Tdate +  
 '}';  
 }  
}

1. Hotel类

public class Hotel {  
 private int Hid;  
 private String Hname;  
 private short Hprice;  
 private String Hstyle;  
 private short Hstarrating;  
 private String Hlocation;  
  
 // 无参构造器  
 public Hotel() {  
 this.Hid = Hid;  
 this.Hname = Hname;  
 this.Hprice = Hprice;  
 this.Hstyle = Hstyle;  
 this.Hstarrating = Hstarrating;  
 this.Hlocation = Hlocation;  
 }

//有参构造器  
 public Hotel(int Hid, String Hname, short Hprice, String Hstyle, short Hstarrating, String Hlocation) {

this.Hid = Hid;

this.Hname = Hname;

this.Hprice = Hprice;

this.Hstyle = Hstyle;

this.Hstarrating = Hstarrating;

this.Hlocation = Hlocation;

}

@Override  
 public String toString() {  
 return "Hotel{" +  
 "Hid=" + Hid +  
 ", Hname='" + Hname +  
 ", Hprice=" + Hprice +  
 ", Hstyle='" + Hstyle + ", Hstarrating=" + Hstarrating +  
 ", Hlocation=" + Hlocation +  
 '}';  
 }  
}

1. Member类

public class Member {  
 private int Mid;  
 private String Mname;  
 private String Mphone;  
 private String Mlevel;  
 private Date Mdate;  
  
 public Member(int Mid, String Mname, String Mphone, String Mlevel, Date Mdate) {  
 this.Mid = Mid;  
 this.Mname = Mname;  
 this.Mphone = Mphone;  
 this.Mlevel = Mlevel;  
 this.Mdate = Mdate;  
 }  
 @Override  
 public String toString() {  
 return "Member{" +  
 "Mid=" + Mid +  
 ", Mname='" + Mname + '\'' +  
 ", Mphone=" + Mphone +  
 ", Mlevel='" + Mlevel + '\'' +  
 ", Mdate=" + Mdate +  
 '}';  
 }  
}

1. Booking类

public class Booking {  
 private int Mid;  
 private int Tid;  
 private int Rid;  
 private int Hid;  
 private Date Bdate;  
  
 public Booking(int Mid, int Tid, int Rid, int Hid, Date Bdate) {  
 this.Mid = Mid;  
 this.Tid = Tid;  
 this.Rid = Rid;  
 this.Hid = Hid;  
 this.Bdate = Bdate;  
 }  
 @Override  
 public String toString() {  
 return "Booking{" +  
 "Mid=" + Mid +  
 ", Tid=" + Tid +  
 ", Rid=" + Rid +  
 ", Hid=" + Hid +  
 ", Bdate=" + Bdate +  
 '}';  
 }  
}

1. Route类

public class Route {

private int Rid;

private String Rname;

@Override

public String toString() {

return "Route{"+"Rid: " + Rid + ", Rname: " + Rname+"}";

}

}

1. Rtraffic类

public class Rtraffic {  
 private int Rtrafficid;  
 private String Rtrafficway;  
 @Override  
 public String toString() {  
 return "Rtraffic{"+"Rtrafficid: "+Rtrafficid + ", Rtrafficway: " + Rtrafficway+"}";  
 }  
}

1. Routemanage类

public class Routemanage {  
 private int Rtrafficid;  
 private int Tid;  
 private int Rid;  
 private int Hid;  
 private Rtraffic rtraffic;  
 private TourSpot tourSpot;  
 private Route route;  
 private Hotel hotel;  
 @Override  
 public String toString() {  
 return "Routemanage{" +  
 "Rtrafficid=" + Rtrafficid +  
 ", Tid=" + Tid +  
 ", Rid=" + Rid +  
 ", Hid=" + Hid +'\n'+  
 ", " + rtraffic +'\n'+  
 ", " + tourSpot +'\n'+  
 ", " + route +'\n'+  
 ", " + hotel +  
 '}';  
 }  
}

**2.功能类**

1. TourSpotDAOImpl类

public class TourSpotDAOImpl implements BaseDAO<TourSpot> {  
  
 @Override  
 public void add(TourSpot tourSpot) {  
 String query = "INSERT INTO TourSpot (Tname, Tprice, Tlocation, Tdate) VALUES (?, ?, ?, ?)";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, tourSpot.getTname());  
 stmt.setShort(2, tourSpot.getTprice());  
 stmt.setString(3, tourSpot.getTlocation());  
 stmt.setDate(4, tourSpot.getTdate());  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void update(TourSpot tourSpot) {  
 String query = "UPDATE TourSpot SET Tname = ?, Tprice = ?, Tlocation = ?, Tdate = ? WHERE Tid = ?";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, tourSpot.getTname());  
 stmt.setShort(2, tourSpot.getTprice());  
 stmt.setString(3, tourSpot.getTlocation());  
 stmt.setDate(4, tourSpot.getTdate());  
 stmt.setInt(5, tourSpot.getTid());  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void delete(int id) {  
 String query = "DELETE FROM TourSpot WHERE Tid = ?";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id);  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public List<TourSpot> findById(int id) {  
 List<TourSpot> tourSpots = new ArrayList<>();  
 String query = "SELECT \* FROM TourSpot WHERE Tid = ?";  
 TourSpot tourSpot = null;  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id);  
 ResultSet rs = stmt.executeQuery();  
 while (rs.next()) {  
 id = rs.getInt("Tid");  
 String name = rs.getString("Tname");  
 short price = rs.getShort("Tprice");  
 String location = rs.getString("Tlocation");  
 Date date = rs.getDate("Tdate");  
 tourSpots.add(new TourSpot());  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return tourSpots;  
 }  
}

1. HotelDAOImpl类

public class HotelDAOImpl implements BaseDAO<Hotel> {  
  
 @Override  
 public void add(Hotel hotel) {  
 String query = "INSERT INTO Hotel (Hname, Hprice, Hstyle, Hstarrating, Hlocation) VALUES (?, ?, ?, ?, ?)";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, hotel.getHname());  
 stmt.setShort(2, hotel.getHprice());  
 stmt.setString(3, hotel.getHstyle());  
 stmt.setShort(4, hotel.getHstarrating());  
 stmt.setString(5, hotel.getHlocation());  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void update(Hotel hotel) {  
 String query = "UPDATE Hotel SET Hname = ?, Hprice = ?, Hstyle = ?, Hstarrating = ?, Hlocation = ? WHERE Hid = ?";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, hotel.getHname());  
 stmt.setShort(2, hotel.getHprice());  
 stmt.setString(3, hotel.getHstyle());  
 stmt.setShort(4, hotel.getHstarrating());  
 stmt.setString(5, hotel.getHlocation());  
 stmt.setInt(6, hotel.getHid());  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void delete(int id) {  
 String query = "DELETE FROM Hotel WHERE Hid = ?";  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id);  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public List<Hotel> findById(int id) {  
 List<Hotel> hotels= new ArrayList<>();  
 String query = "SELECT \* FROM Hotel WHERE Hid = ?";  
 Hotel hotel = null;  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id);  
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 hotels.add(new Hotel(  
 ));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return hotels;  
 }  
}

1. MemberDAOImpl类

public class MemberDAOImpl implements BaseDAO<Member> {  
  
 private static final String DB\_URL = "jdbc:sqlserver://localhost:1433;databaseName=TravelBookingDB;trustServerCertificate=true";  
 private static final String DB\_USER = "sa";  
 private static final String DB\_PASSWORD = "123456";  
  
 @Override  
 public void add(Member member) {  
 String query = "INSERT INTO Member (Mname, Mphone, Mlevel, Mdate) VALUES (?, ?, ?, ?)";  
 try (Connection conn = getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 // 从 Member 对象中获取数据并绑定到 PreparedStatement  
 stmt.setString(1, member.getMname()); // 获取 Mname  
 stmt.setString(2, member.getMphone()); // 获取 Mphone  
 stmt.setString(3, member.getMlevel()); // 获取 Mlevel  
 stmt.setDate(4, (Date) member.getMdate()); // 获取 Mate  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void update(Member member) {  
 String query = "UPDATE Member SET Mname = ?, Mphone = ?, Mlevel = ?, Mdate = ? WHERE Mid = ?";  
 try (Connection conn = getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 // 从 Member 对象中获取数据并绑定到 PreparedStatement  
 stmt.setString(1, member.getMname()); // 获取 Mname  
 stmt.setString(2, member.getMphone()); // 获取 Mphone  
 stmt.setString(3, member.getMlevel()); // 获取 Mlevel  
 stmt.setDate(4, (Date) member.getMdate()); // 获取 Mdate  
 stmt.setInt(5, member.getMid()); // 获取 Mid (用于更新指定的会员)  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 System.err.println("Error updating member: " + e.getMessage());  
 }  
 }  
  
 @Override  
 public void delete(int id) {  
 String query = "DELETE FROM Member WHERE Mid = ?";  
 try (Connection conn = getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id); // 通过 Mid 删除  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 System.err.println("Error deleting member: " + e.getMessage());  
 }  
 }  
  
 @Override  
 public List<Member> findById(int id) {  
 List<Member> members = new ArrayList<>();  
 String query = "SELECT \* FROM Member WHERE Mid = ?";  
 Member member = null;  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, id); // 设置 Mid 进行查询  
 try (ResultSet rs = stmt.executeQuery()) {  
 if (rs.next()) {  
 // 从 ResultSet 中获取数据并创建 Member 对象  
 members.add (new Member(  
 rs.getInt("Mid"), // 获取 Mid  
 rs.getString("Mname"), // 获取 Mname  
 rs.getString("Mphone"), // 获取 Mphone  
 rs.getString("Mlevel"), // 获取 Mlevel  
 rs.getDate("Mdate") // 获取 Mdate  
 ));  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return members;  
 }  
}

1. BookingDAOImpl类

public class BookingDAOImpl implements BookingDAO {  
 private static final String DB\_URL = "jdbc:sqlserver://localhost:1433;databaseName=TravelBookingDB;trustServerCertificate=true";  
 private static final String DB\_USER = "sa";  
 private static final String DB\_PASSWORD = "123456";   
 // 根据会员 ID 查询预定  
 @Override  
 public List<Booking> findBookingsByMemberId(int memberId) {  
 List<Booking> bookings = new ArrayList<>();  
 String query = "SELECT \* FROM Booking WHERE Mid = ?";   
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, memberId);  
 try (ResultSet rs = stmt.executeQuery()) {  
 while (rs.next()) {  
 bookings.add(new Booking(  
 rs.getInt("Mid"),  
 rs.getInt("Tid"),  
 rs.getInt("Rid"),  
 rs.getInt("Hid"),  
 rs.getDate("Bdate")  
 ));  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return bookings;  
 }  
  
 // 根据酒店 ID 查询预定  
 @Override  
 public List<Booking> findBookingsByHotelId(int hotelId) {  
 List<Booking> bookings = new ArrayList<>();  
 String query = "SELECT \* FROM Booking WHERE Hid = ?";   
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, hotelId);  
 try (ResultSet rs = stmt.executeQuery()) {  
 while (rs.next()) {  
 bookings.add(new Booking(  
 rs.getInt("Mid"),  
 rs.getInt("Tid"),  
 rs.getInt("Rid"),  
 rs.getInt("Hid"),  
 rs.getDate("Bdate")  
 ));  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return bookings;  
 }  
  
 // 根据旅游景点ID 查询预定  
 @Override  
 public List<Booking> findBookingsByTourSpotId(int tourSpotId) {  
 List<Booking> bookings = new ArrayList<>();  
 String query = "SELECT \* FROM Booking WHERE Tid = ?";   
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, tourSpotId);  
 try (ResultSet rs = stmt.executeQuery()) {  
 while (rs.next()) {  
 bookings.add(new Booking(  
 rs.getInt("Mid"),  
 rs.getInt("Tid"),  
 rs.getInt("Rid"),  
 rs.getInt("Hid"),  
 rs.getDate("Bdate")  
 ));  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return bookings;  
 }

// 根据旅游线路ID 查询预定  
 public List<Booking> findBookingsByRouteId(int routeId) {  
 List<Booking> bookings = new ArrayList<>();  
 String query = "SELECT \* FROM Booking WHERE Rid = ?";   
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setInt(1, routeId);  
 try (ResultSet rs = stmt.executeQuery()) {  
 while (rs.next()) {  
 bookings.add(new Booking(  
 rs.getInt("Mid"),  
 rs.getInt("Tid"),  
 rs.getInt("Rid"),  
 rs.getInt("Hid"),  
 rs.getDate("Bdate")  
 ));  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return bookings;  
 }  
}

1. RoutemanageDAOImpl类

public class RoutemanageDAOImpl {  
 private static final String DB\_URL = "jdbc:sqlserver://localhost:1433;databaseName=TravelBookingDB;trustServerCertificate=true";  
 private static final String DB\_USER = "sa";  
 private static final String DB\_PASSWORD = "123456";  
  
 public List<Routemanage> findAllRoutemanageWithDetails() {  
 List<Routemanage> routemanages = new ArrayList<>();  
 String query = "SELECT rm.\*, rt.\*, ts.\*, r.\*, h.\* " +  
 "FROM Routemanage rm " +  
 "JOIN Rtraffic rt ON rm.Rtrafficid = rt.Rtrafficid " +  
 "JOIN TourSpot ts ON rm.Tid = ts.Tid " +  
 "JOIN Route r ON rm.Rid = r.Rid " +  
 "JOIN Hotel h ON rm.Hid = h.Hid";  
  
 try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 PreparedStatement stmt = conn.prepareStatement(query);  
 ResultSet rs = stmt.executeQuery()) {  
  
 while (rs.next()) {  
 Routemanage routemanage = new Routemanage();  
 routemanage.setRtrafficid(rs.getInt("Rtrafficid"));  
 routemanage.setTid(rs.getInt("Tid"));  
 routemanage.setRid(rs.getInt("Rid"));  
 routemanage.setHid(rs.getInt("Hid"));  
 routemanages.add(routemanage);  
 // 创建并设置 Rtraffic 对象  
 Rtraffic rtraffic = new Rtraffic();  
 rtraffic.setRtrafficid(rs.getInt("Rtrafficid"));  
 rtraffic.setRtrafficway(rs.getString("Rtrafficway"));  
 routemanage.setRtraffic(rtraffic);  
 // 创建并设置 TourSpot 对象  
 TourSpot tourSpot = new TourSpot();  
 tourSpot.setTid(rs.getInt("Tid"));  
 tourSpot.setTname(rs.getString("Tname"));  
 tourSpot.setTprice(rs.getShort("Tprice"));  
 tourSpot.setTlocation(rs.getString("Tlocation"));  
 tourSpot.setTdate(rs.getDate("Tdate"));  
 routemanage.setTourSpot(tourSpot);  
  
 // 创建并设置 Route 对象  
 Route route = new Route();  
 route.setRid(rs.getInt("Rid"));  
 route.setRname(rs.getString("Rname"));  
 routemanage.setRoute(route);  
  
 // 创建并设置 Hotel 对象  
 Hotel hotel = new Hotel();  
 hotel.setHid(rs.getInt("Hid"));  
 hotel.setHname(rs.getString("Hname"));  
 hotel.setHprice(rs.getShort("Hprice"));  
 hotel.setHstyle(rs.getString("Hstyle"));  
 hotel.setHstarrating(rs.getShort("Hstarrating"));  
 hotel.setHlocation(rs.getString("Hlocation"));  
 routemanage.setHotel(hotel);  
  
 routemanages.add(routemanage);  
  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return routemanages;  
 }  
}

1. MostPopularRoute类

public class MostPopularRoute {  
 private static final String DB\_URL = "jdbc:sqlserver://localhost:1433;databaseName=TravelBookingDB;trustServerCertificate=true";  
 private static final String DB\_USER = "sa";  
 private static final String DB\_PASSWORD = "123456";  
  
 public static void getMostPopularRoute() {  
 Connection connection = null;  
 Statement statement = null;  
 ResultSet resultSet = null;  
  
 try {  
 connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
 statement = connection.createStatement();  
 String sql = "SELECT TOP 1 Rid, COUNT(\*) AS booking\_count " +  
 "FROM Booking " +  
 "GROUP BY Rid " +  
 "ORDER BY booking\_count DESC " ;  
 resultSet = statement.executeQuery(sql);  
  
 if (resultSet.next()) {  
 int rid = resultSet.getInt("Rid"); // 获取线路ID  
 int bookingCount = resultSet.getInt("booking\_count"); // 获取预订数量  
 System.out.println("Most Popular Route ID: " + rid);  
 System.out.println("Booking Count: " + bookingCount);  
 } else {  
 System.out.println("No data found.");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 // 7. 关闭资源  
 try {  
 if (resultSet != null) {  
 resultSet.close();  
 }  
 if (statement != null) {  
 statement.close();  
 }  
 if (connection != null) {  
 connection.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

1. TicketBooking类

public class TicketBooking {  
 private static final String DB\_URL = "jdbc:sqlserver://localhost:1433;databaseName=TravelBookingDB;trustServerCertificate=true";  
 private static final String DB\_USER = "sa";  
 private static final String DB\_PASSWORD = "123456";  
  
 public static void bookTicket(int tid, int quantity) throws SQLException {  
 Connection connection = null;  
 PreparedStatement preparedStatement = null;  
 ResultSet resultSet = null;  
  
 try {  
 connection = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);  
  
 connection.setAutoCommit(false);  
  
 String query = "SELECT Total\_tickets FROM TourSpot WHERE Tid = ?";  
 preparedStatement = connection.prepareStatement(query);  
 preparedStatement.setInt(1, tid);  
 resultSet = preparedStatement.executeQuery();  
  
 if (resultSet.next()) {  
 int currentTickets = resultSet.getInt("Total\_tickets");  
  
 // 5. 判断是否有足够的门票  
 if (currentTickets >= quantity) {  
 String updateQuery = "UPDATE TourSpot SET Total\_tickets = Total\_tickets - ? WHERE Tid = ?";  
 preparedStatement = connection.prepareStatement(updateQuery);  
 preparedStatement.setInt(1, quantity);  
 preparedStatement.setInt(2, tid);  
 int rowsAffected = preparedStatement.executeUpdate();  
  
 // 判断更新是否成功  
 if (rowsAffected > 0) {  
 connection.commit();  
 System.out.println("预定成功，门票数量已更新!");  
 } else {  
 connection.rollback();  
 System.out.println("更新失败，预定未成功!");  
 }  
 } else {  
 connection.rollback();  
 System.out.println("门票不足，预定失败!");  
 }  
 } else {  
 System.out.println("线路ID不存在!");  
 }  
 } catch (SQLException e) {  
 if (connection != null) {  
 connection.rollback();  
 }  
 e.printStackTrace();  
 } finally {  
 // 关闭资源  
 try {  
 if (resultSet != null) {  
 resultSet.close();  
 }  
 if (preparedStatement != null) {  
 preparedStatement.close();  
 }  
 if (connection != null) {  
 connection.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

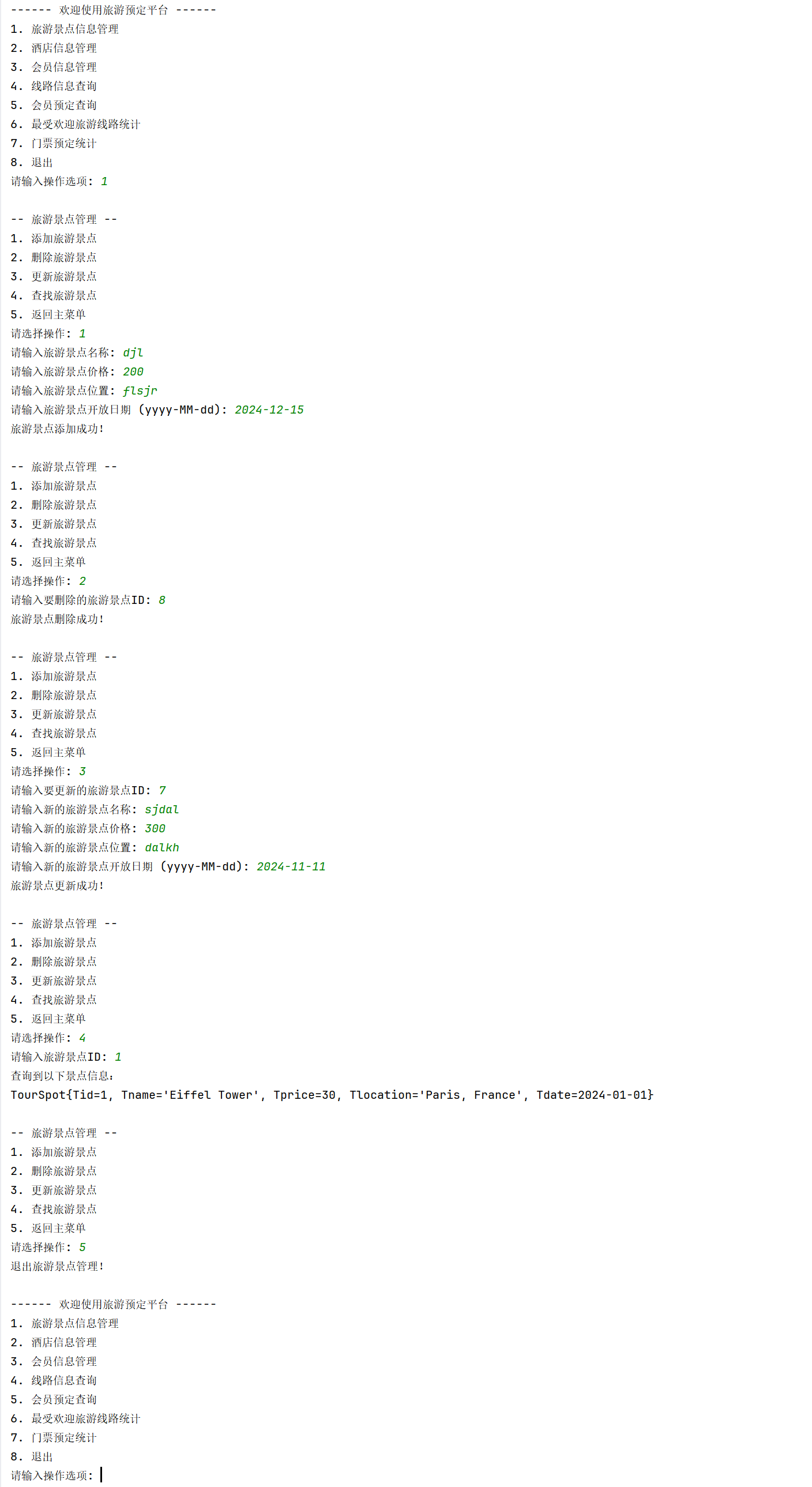
## 5.2 界面代码

界面为命令行（或者图形），代码如下：

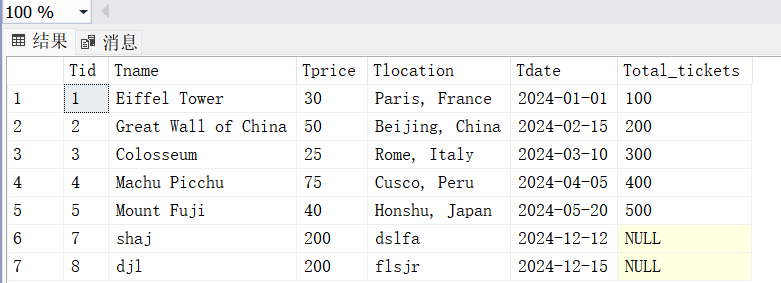
public class Main {  
  
 private static Scanner scanner = new Scanner(System.in);  
  
 public static void main(String[] args) {  
 while (true) {  
 System.out.println("\n------ 欢迎使用旅游预定平台 ------");  
 System.out.println("1. 旅游景点信息管理");  
 System.out.println("2. 酒店信息管理");  
 System.out.println("3. 会员信息管理");  
 System.out.println("4. 线路信息查询");  
 System.out.println("5. 会员预定查询");  
 System.out.println("6. 最受欢迎旅游线路统计");  
 System.out.println("7. 门票预定统计");  
 System.out.println("8. 退出");  
 System.out.print("请输入操作选项: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
  
 switch (choice) {  
 case 1:  
 touristSpotManagement();  
 break;  
 case 2:  
 hotelManagement();  
 break;  
 case 3:  
 memberManagement();  
 break;  
 case 4:  
 findAllRouteInfo();  
 break;  
 case 5:  
 BookingService();  
 break;  
 case 6:  
 MostPopularRoute.getMostPopularRoute();  
 break;  
 case 7:  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("请输入预定的线路ID:");  
 int tid = scanner.nextInt();  
 System.out.println("请输入预定的数量:");  
 int quantity = scanner.nextInt();  
 try {  
 TicketBooking.bookTicket(tid, quantity);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 case 8:  
 System.out.println("感谢使用！");  
 return;  
 default:  
 System.out.println("无效的选项，请重新输入！");  
 }  
 }  
 }  
  
  
  
 // 会员管理  
 private static void memberManagement() {  
 while (true) {  
 System.out.println("\n------ 会员管理 ------");  
 System.out.println("1. 添加会员信息");  
 System.out.println("2. 删除会员信息");  
 System.out.println("3. 更新会员信息");  
 System.out.println("4. 查找会员信息");  
 System.out.println("5. 返回主菜单");  
 System.out.print("请输入操作选项: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
  
 switch (choice) {  
 case 1:  
 addMember();  
 break;  
 case 2:  
 deleteMember();  
 break;  
 case 3:  
 updateMember();  
 break;  
 case 4:  
 findMember();  
 break;  
 case 5:  
 return;  
 default:  
 System.out.println("无效的选项，请重新输入！");  
 }  
 }  
 }  
  
 private static void addMember() {  
 System.out.print("请输入会员姓名: ");  
 String mname = scanner.nextLine();  
 System.out.print("请输入会员电话: ");  
 String mphone = scanner.nextLine();  
 System.out.print("请输入会员等级: ");  
 String mlevel = scanner.nextLine();  
 System.out.print("请输入会员注册日期 (yyyy-MM-dd): ");  
 String mdateStr = scanner.nextLine();  
 java.sql.Date mdate = java.sql.Date.valueOf(mdateStr);  
 Member member = new Member(0, mname, mphone, mlevel, mdate);  
 MemberDAOImpl memberDAOImpl = new MemberDAOImpl();  
 memberDAOImpl.add(member);  
 System.out.println("会员添加成功！");  
 }  
  
 private static void deleteMember() {  
 System.out.print("请输入要删除的会员 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 MemberDAOImpl memberDAOImpl = new MemberDAOImpl();  
 memberDAOImpl.delete(id);  
 System.out.println("会员删除成功！");  
 }  
  
 private static void updateMember() {  
 System.out.print("请输入要更新的会员 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 System.out.print("请输入新的会员姓名: ");  
 String mname = scanner.nextLine();  
 System.out.print("请输入新的会员电话: ");  
 String mphone = scanner.nextLine();  
 System.out.print("请输入新的会员等级: ");  
 String mlevel = scanner.nextLine();  
 System.out.print("请输入新的会员注册日期 (yyyy-MM-dd): ");  
 String mdateStr = scanner.nextLine();  
 java.sql.Date mdate = java.sql.Date.valueOf(mdateStr);  
 Member member = new Member(id, mname, mphone, mlevel, mdate);  
 MemberDAOImpl memberDAOImpl = new MemberDAOImpl();  
 memberDAOImpl.update(member);  
 System.out.println("会员信息更新成功！");  
 }  
  
 private static void findMember() {  
 System.out.print("请输入要查找的会员 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 MemberDAOImpl memberDAOImpl = new MemberDAOImpl();  
 List<Member> members = memberDAOImpl.findById(id); // 返回多个匹配的会员  
  
 if (members != null && !members.isEmpty()) {  
 System.out.println("查询到以下会员信息: ");  
 for (Member member : members) {  
 System.out.println(member);  
 }  
 } else {  
 System.out.println("未找到匹配的会员！");  
 }  
 }  
  
 // 酒店管理  
 private static void hotelManagement() {  
 HotelDAOImpl hotelDAOImpl = new HotelDAOImpl();  
  
 while (true) {  
 System.out.println("\n------ 酒店管理 ------");  
 System.out.println("1. 添加酒店信息");  
 System.out.println("2. 删除酒店信息");  
 System.out.println("3. 更新酒店信息");  
 System.out.println("4. 查找酒店信息");  
 System.out.println("5. 返回主菜单");  
 System.out.print("请输入操作选项: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
  
 switch (choice) {  
 case 1:  
 addHotel();  
 break;  
 case 2:  
 deleteHotel();  
 break;  
 case 3:  
 updateHotel();  
 break;  
 case 4:  
 findHotel();  
 break;  
 case 5:  
 return;  
 default:  
 System.out.println("无效的选项，请重新输入！");  
 }  
 }  
 }  
  
 private static void addHotel() {  
 System.out.print("请输入酒店名称: ");  
 String hname = scanner.nextLine();  
 System.out.print("请输入酒店价格: ");  
 short hprice = scanner.nextShort();  
 System.out.print("请输入酒店风格: ");  
 String hstyle = scanner.nextLine();  
 System.out.print("请输入酒店星级: ");  
 short hstarrating = scanner.nextShort();  
 System.out.print("请输入酒店位置: ");  
 String hlocation = scanner.nextLine();  
 scanner.nextLine(); // 清除换行符  
 Hotel hotel = new Hotel(0, hname, hprice, hstyle, hstarrating, hlocation);  
 HotelDAOImpl hotelDAOImpl = new HotelDAOImpl();  
 hotelDAOImpl.add(hotel);  
 System.out.println("酒店添加成功！");  
 }  
  
 private static void deleteHotel() {  
 System.out.print("请输入要删除的酒店 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 HotelDAOImpl hotelDAOImpl = new HotelDAOImpl();  
 hotelDAOImpl.delete(id);  
 System.out.println("酒店删除成功！");  
 }  
  
 private static void updateHotel() {  
 System.out.print("请输入要更新的酒店 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 System.out.print("请输入新的酒店名称: ");  
 String hname = scanner.nextLine();  
 System.out.print("请输入新的酒店价格: ");  
 short hprice = scanner.nextShort();  
 System.out.print("请输入新的酒店风格: ");  
 String hstyle = scanner.nextLine();  
 System.out.print("请输入新的酒店星级: ");  
 short hstarrating = scanner.nextShort();  
 System.out.print("请输入新的酒店位置: ");  
 String hlocation = scanner.nextLine();  
 scanner.nextLine(); // 清除换行符  
 Hotel hotel = new Hotel(id, hname, hprice, hstyle, hstarrating, hlocation);  
 HotelDAOImpl hotelDAOImpl = new HotelDAOImpl();  
 hotelDAOImpl.update(hotel);  
 System.out.println("酒店信息更新成功！");  
  
 }  
  
 private static void findHotel() {  
 System.out.print("请输入要查找的酒店 ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 HotelDAOImpl hotelDAOImpl = new HotelDAOImpl();  
 List<Hotel> hotels = hotelDAOImpl.findById(id);  
 if (hotels != null && !hotels.isEmpty()) {  
 System.out.println("查询到以下会员信息: ");  
 for (Hotel hotel : hotels) {  
 System.out.println(hotel);  
 }  
 } else {  
 System.out.println("未找到该酒店！");  
 }  
 }  
  
  
 // 旅游景点管理  
 private static void touristSpotManagement() {  
 TourSpotDAOImpl tourSpotDAOImpl = new TourSpotDAOImpl();  
  
 while (true) {  
 System.out.println("\n-- 旅游景点管理 --");  
 System.out.println("1. 添加旅游景点");  
 System.out.println("2. 删除旅游景点");  
 System.out.println("3. 更新旅游景点");  
 System.out.println("4. 查找旅游景点");  
 System.out.println("5. 返回主菜单");  
 System.out.print("请选择操作: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // 清空换行符  
  
 switch (choice) {  
 case 1:  
 addTouristSpot();  
 break;  
 case 2:  
 deleteTouristSpot();  
 break;  
 case 3:  
 updateTouristSpot();  
 break;  
 case 4:  
 findTouristSpot();  
 break;  
 case 5:  
 System.out.println("退出旅游景点管理！");  
 return;  
 default:  
 System.out.println("无效的选项，请重新选择！");  
 }  
 }  
 }  
  
 // 添加旅游景点  
 private static void addTouristSpot() {  
 System.out.print("请输入旅游景点名称: ");  
 String Tname = scanner.nextLine();  
 System.out.print("请输入旅游景点价格: ");  
 short Tprice = scanner.nextShort();  
 scanner.nextLine(); // 清空换行符  
 System.out.print("请输入旅游景点位置: ");  
 String Tlocation = scanner.nextLine();  
 System.out.print("请输入旅游景点开放日期 (yyyy-MM-dd): ");  
 String TdateStr = scanner.nextLine();  
 java.sql.Date Tdate = java.sql.Date.valueOf(TdateStr);  
 TourSpot spot = new TourSpot(0,Tname,Tprice,Tlocation,Tdate);  
 TourSpotDAOImpl tourSpotDAOImpl = new TourSpotDAOImpl();  
 tourSpotDAOImpl.add(spot);  
 System.out.println("旅游景点添加成功！");  
 }  
  
 // 删除旅游景点  
 private static void deleteTouristSpot() {  
 System.out.print("请输入要删除的旅游景点ID: ");  
 int Tid = scanner.nextInt();  
 scanner.nextLine();  
 TourSpotDAOImpl tourSpotDAOImpl = new TourSpotDAOImpl();  
 tourSpotDAOImpl.delete(Tid);  
 System.out.println("旅游景点删除成功！");  
 }  
  
 // 更新旅游景点  
 private static void updateTouristSpot() {  
 System.out.print("请输入要更新的旅游景点ID: ");  
 int Tid = scanner.nextInt();  
 scanner.nextLine(); // 清空换行符  
 System.out.print("请输入新的旅游景点名称: ");  
 String Tname = scanner.nextLine();  
 System.out.print("请输入新的旅游景点价格: ");  
 short Tprice = scanner.nextShort();  
 scanner.nextLine(); // 清空换行符  
 System.out.print("请输入新的旅游景点位置: ");  
 String Tlocation = scanner.nextLine();  
 System.out.print("请输入新的旅游景点开放日期 (yyyy-MM-dd): ");  
 String TdateStr = scanner.nextLine();  
 java.sql.Date Tdate = java.sql.Date.valueOf(TdateStr);  
 TourSpotDAOImpl tourSpotDAOImpl = new TourSpotDAOImpl();  
 TourSpot spot = new TourSpot(Tid,Tname,Tprice,Tlocation,Tdate);  
 tourSpotDAOImpl.update(spot);  
 System.out.println("旅游景点更新成功！");  
 }  
  
 // 查找旅游景点  
 private static void findTouristSpot() {  
 System.out.print("请输入旅游景点ID: ");  
 int Tid = scanner.nextInt();  
 scanner.nextLine();  
 TourSpotDAOImpl tourSpotDAOImpl = new TourSpotDAOImpl();  
 List<TourSpot> spots = tourSpotDAOImpl.findById(Tid);  
 if (spots != null && !spots.isEmpty()) {  
 System.out.println("查询到以下景点信息：");  
 for (TourSpot spot : spots) {  
 System.out.println(spot);  
 }  
 } else {  
 System.out.println("未找到该旅游景点！");  
 }  
 }  
  
 //线路信息查询  
 private static void findAllRouteInfo() {  
  
 RoutemanageDAO routemanageDAO = new RoutemanageDAO();  
  
 // 查询所有线路信息及其相关详细信息  
 List<Routemanage> routemanages = routemanageDAO.findAllRoutemanageWithDetails();  
  
 // 输出查询到的线路信息  
 if (routemanages.isEmpty()) {  
 System.out.println("未找到任何线路信息。");  
 } else {  
 // 遍历并打印每个 Routemanage 对象的详细信息  
 for (Routemanage routemanage : routemanages) {  
 // 打印出 Routemanage 对象的字段（假设你已定义合适的 toString() 方法）  
 System.out.println(routemanage);  
 System.out.print("\n");  
 }  
 }  
 }  
 //会员预定查询  
 private static void BookingService() {  
  
 while (true) {  
 System.out.println("\n------ 预定管理 ------");  
 System.out.println("1. 根据会员ID查询");  
 System.out.println("2. 根据酒店ID查询");  
 System.out.println("3. 根据旅游景点ID查询");  
 System.out.println("4. 根据旅游线路ID查询");  
 System.out.println("5. 返回主菜单");  
 System.out.print("请输入操作选项: ");  
 int choice = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
  
 switch (choice) {  
 case 1:  
 findBookingsByMemberId();  
 break;  
 case 2:  
 findBookingsByHotelId();  
 break;  
 case 3:  
 findBookingsByTourSpotId();  
 break;  
 case 4:  
 findBookingsByRouteId();  
 break;  
 case 5:  
 return; // 返回主菜单  
 default:  
 System.out.println("无效的选项，请重新输入！");  
 }  
 }  
 }  
  
 // 根据会员ID查询  
 private static void findBookingsByMemberId() {  
 System.out.print("请输入会员ID: ");  
 int memberId = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 BookingDAOImpl bookingDAOImpl = new BookingDAOImpl();  
 List<Booking> bookings = bookingDAOImpl.findBookingsByMemberId(memberId);  
 displayBookings(bookings);  
 }  
  
 // 根据酒店ID查询  
 private static void findBookingsByHotelId() {  
 System.out.print("请输入酒店ID: ");  
 int hotelId = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 BookingDAOImpl bookingDAOImpl = new BookingDAOImpl();  
 List<Booking> bookings = bookingDAOImpl.findBookingsByHotelId(hotelId);  
 displayBookings(bookings);  
 }  
  
 // 根据旅游景点ID查询  
 private static void findBookingsByTourSpotId() {  
 System.out.print("请输入旅游景点ID: ");  
 int tourSpotId = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 BookingDAOImpl bookingDAOImpl = new BookingDAOImpl();  
 List<Booking> bookings = bookingDAOImpl.findBookingsByTourSpotId(tourSpotId);  
 displayBookings(bookings);  
 }  
  
 // 根据旅游线路ID查询  
 private static void findBookingsByRouteId() {  
 System.out.print("请输入旅游景点ID: ");  
 int routeId = scanner.nextInt();  
 scanner.nextLine(); // 清除换行符  
 BookingDAOImpl bookingDAOImpl = new BookingDAOImpl();  
 List<Booking> bookings = bookingDAOImpl.findBookingsByRouteId(routeId);  
 displayBookings(bookings);  
 }  
  
 // 显示预定信息  
 private static void displayBookings(List<Booking> bookings) {  
 if (bookings.isEmpty()) {  
 System.out.println("没有找到符合条件的预定信息。");  
 } else {  
 System.out.println("\n------ 查询结果 ------");  
 for (Booking booking : bookings) {  
 System.out.println(booking);  
 }  
 }  
 }  
 }

# 第六章 运行结果

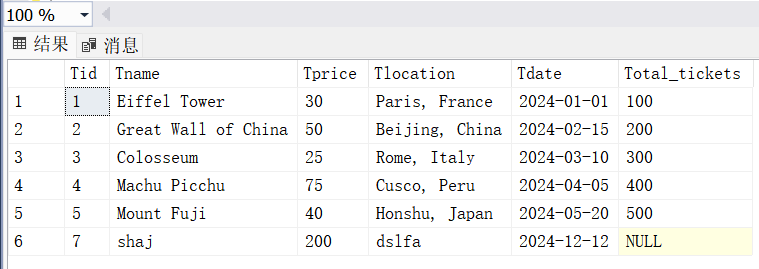
## 6.1（旅游景点信息管理）



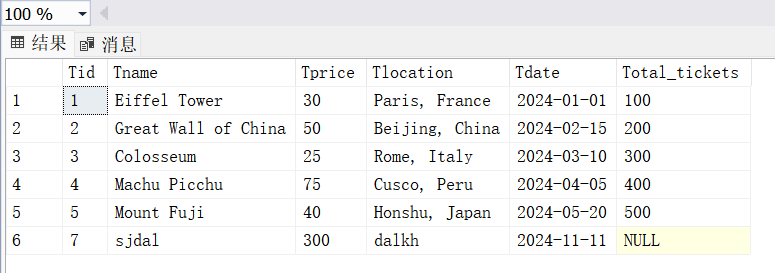
1.添加景点信息



1. 删除景点信息

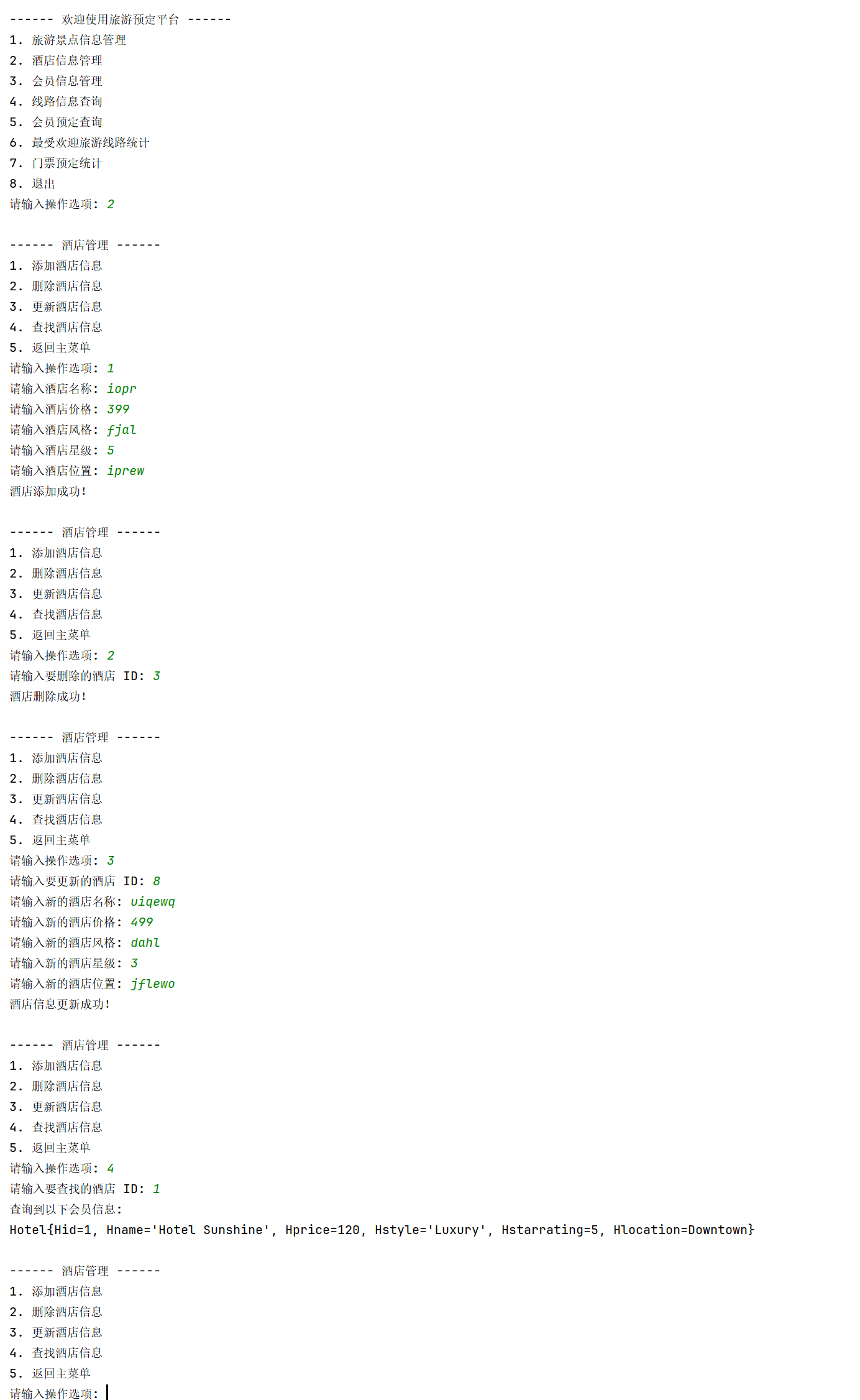


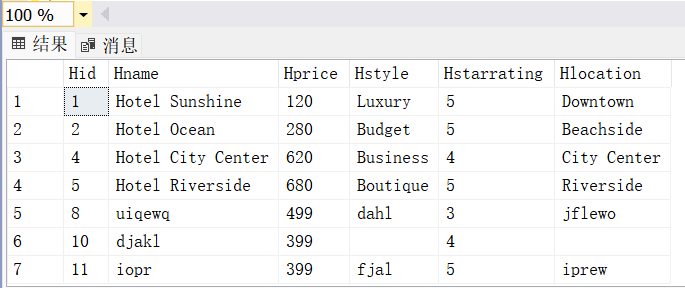
1. 更新景点信息



**图6-1旅游景点信息管理结果截图**

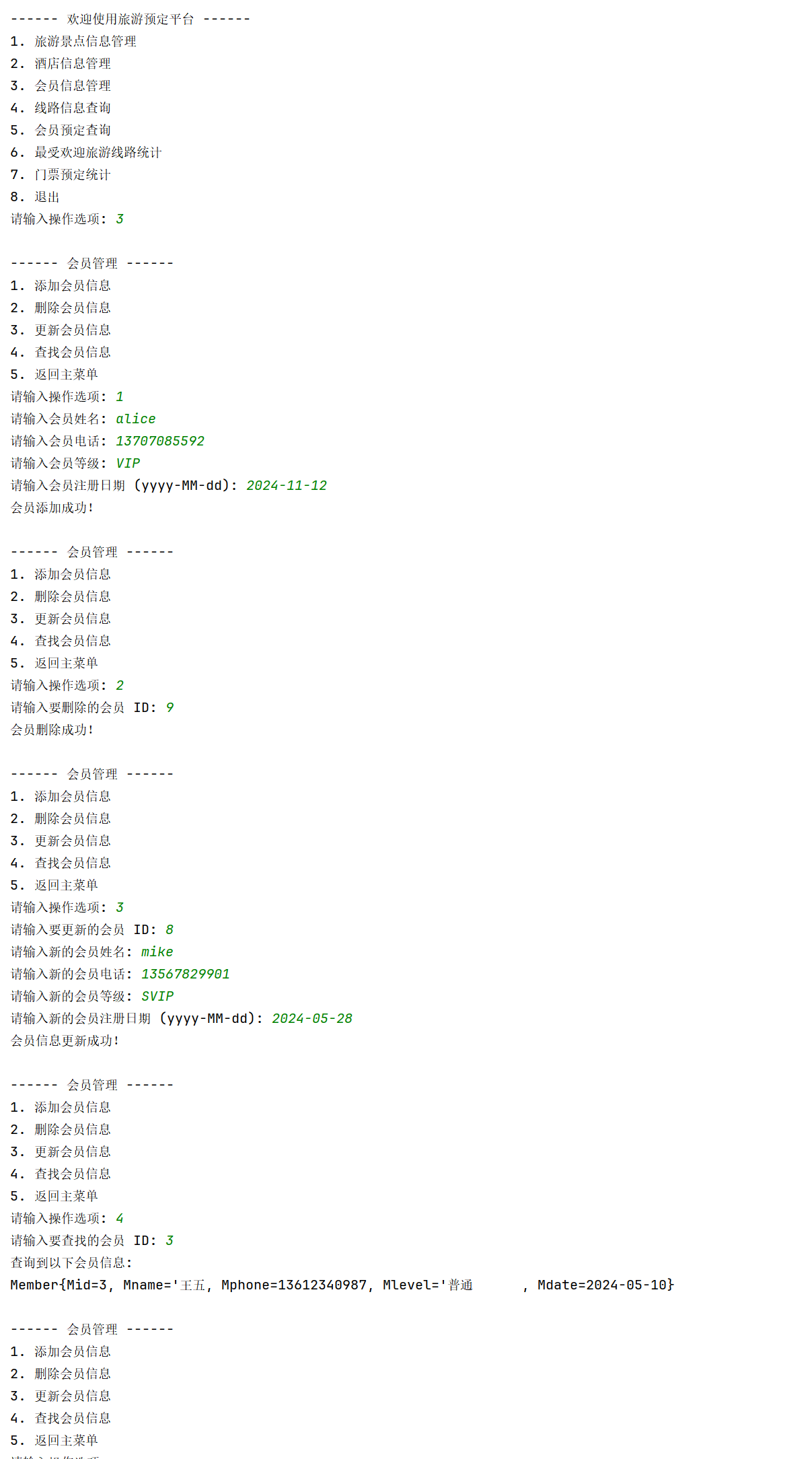
## 6.2（酒店信息管理）



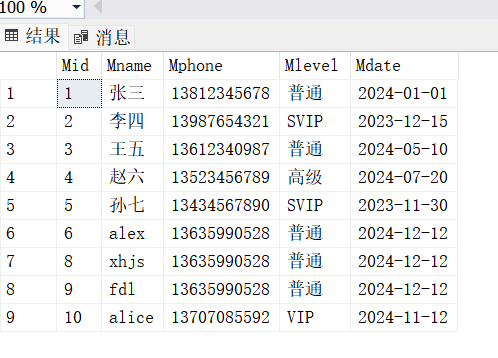


**图6-2酒店信息管理结果截图**

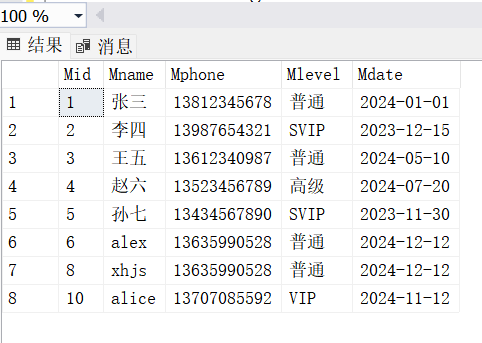
## 6.3（会员信息管理）



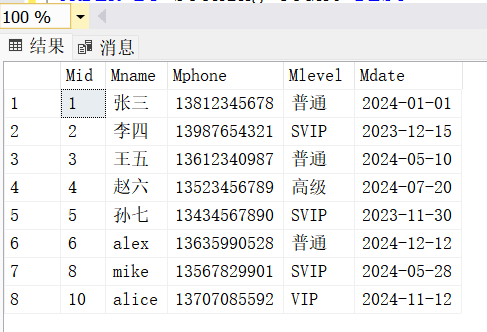
1.添加会员信息



2.删除会员信息

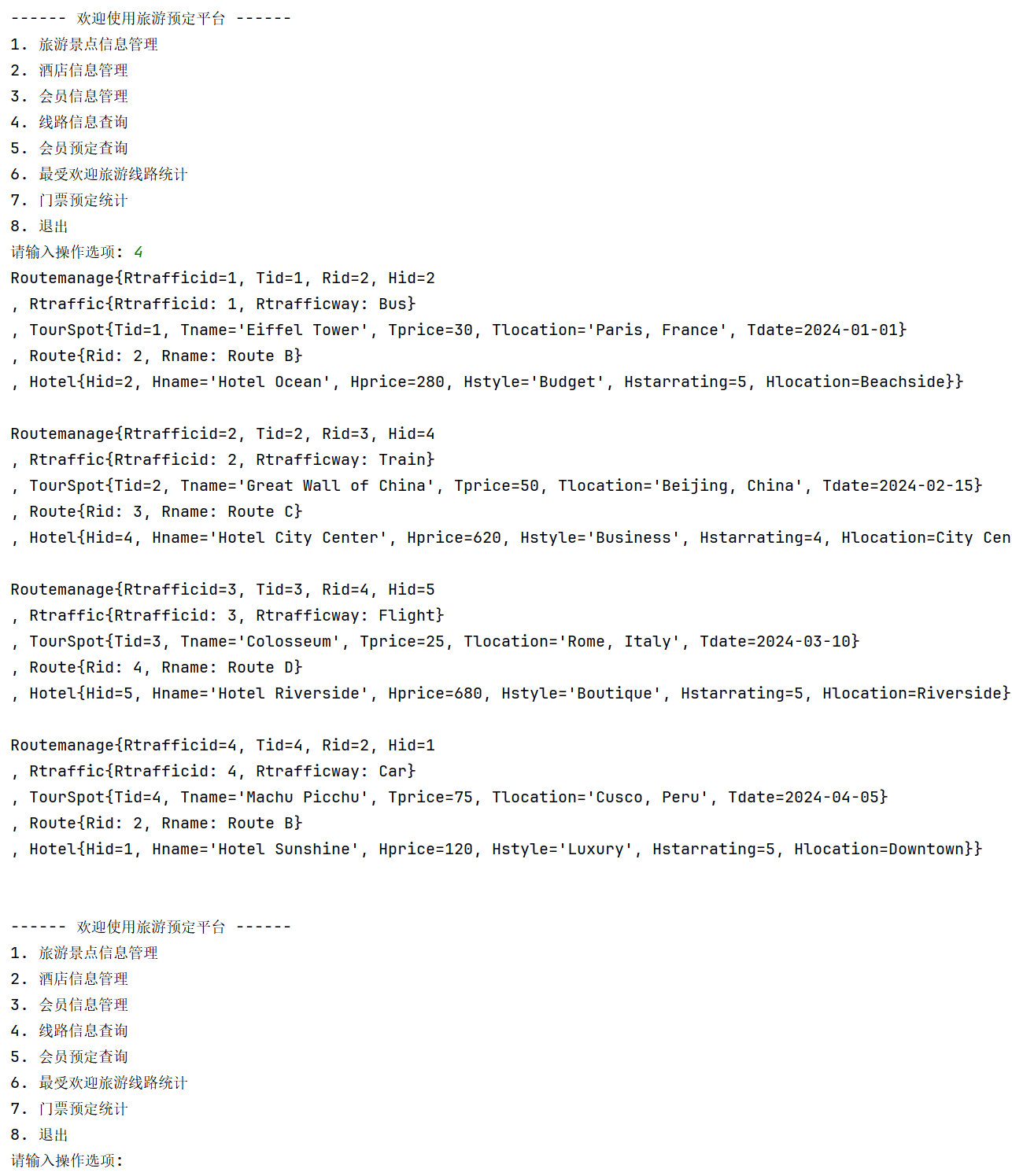


3.更新会员信息



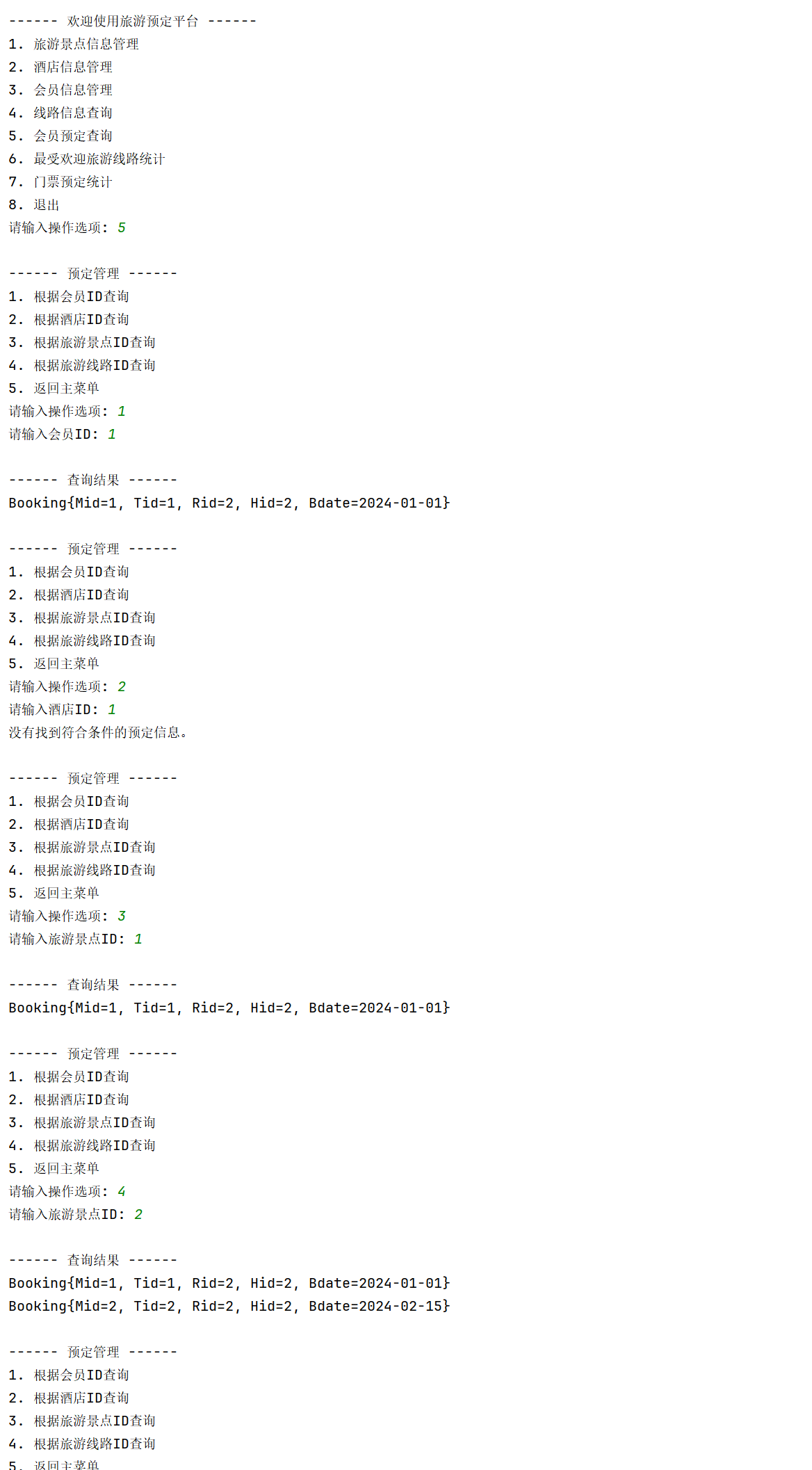
**图6-3 会员信息管理结果截图**

## 6.4（线路信息查询）



**图6-4 线路信息查询结果截图**

## 6.5（会员预定查询）



**图6-5 会员预定查询结果截图**

## 6.6（最受欢迎旅游线路统计）



**图6-6 最受欢迎旅游线路统计结果截图**

## 6.7（门票预定统计）



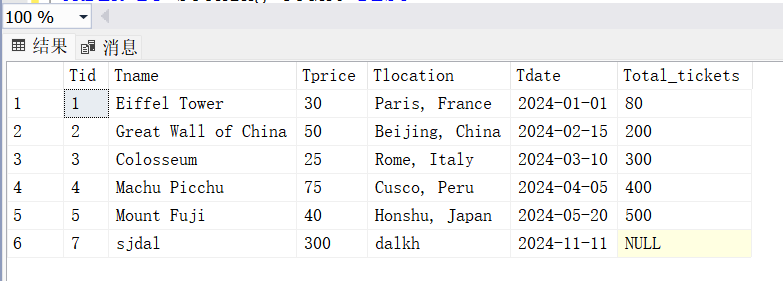
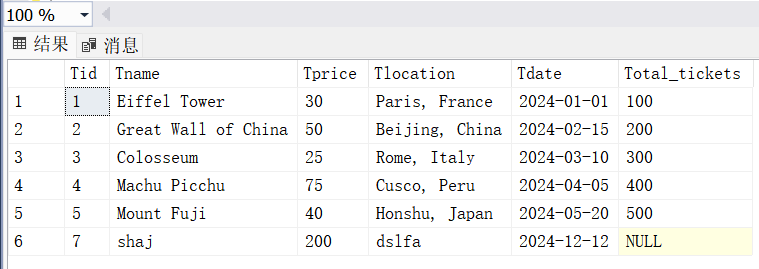


图6-7 门票预定统计结果截图

# 第七章 设计小结

## 7.1数据库设计小结

数据库设计的流程通常包括以下几个阶段：

**1.概念结构设计：**  
在这个阶段，首先需要识别数据库中的实体（例如旅游景点，酒店，会员等）以及它们的属性（例如会员id，会员等级等）。接着，分析实体之间的联系（例如旅游线路与旅游景点之间的“多对多”关系）。通过这些信息，构建出E-R 图。

**2.逻辑结构设计：**在概念模型的基础上，进行逻辑结构设计，即将E-R图转换为关系模型。在此过程中，实体转化为关系（表），实体的属性转化为列（字段），实体之间的关系通过外键来约束。此时，我们明确每个关系（表）中具体的字段、数据类型及其约束（如主键、外键、唯一性、非空等）。

**3.数据库实现：**  
接下来，基于已定义的关系模型，我们使用数据定义语言（DDL）来实际创建数据库结构。常用的DDL语句包括：CREATE：用于创建数据库、表、索引等对象。DROP：用于删除数据库或表等对象。ALTER：用于修改数据库或表的结构，例如添加、删除列，修改字段类型等。通过执行这些语句，数据库的结构得以在物理层面实现。

**4.数据库操作：**  
设计完成后，使用数据操纵语言（DML）进行数据的增删改查操作。DML语句包括：INSERT：插入数据。UPDATE：更新现有数据。DELETE：删除数据。SELECT：查询数据。

**5.高级特性：**数据库设计完成后，利用了一些高级特性来增强数据库的功能性与灵活性，比如触发器（Triggers）， 它是一种自动执行的数据库对象，我利用它实现门票被预定后自动更新门票数量的操作，通过这些高级特性，能够使得数据库在执行复杂操作时更加高效和智能。

## 7.2 Java程序设计小结

**1. 程序结构**

本程序采用了面向对象的设计方法，将不同的功能模块分解为独立的方法，便于管理和维护。整个程序包括以下几个主要部分：

（1）主菜单： 提供用户可操作的选项，包括旅游景点信息管理、酒店信息管理、会员信息管理等功能。程序通过一个循环不断提示用户输入选择，直到用户选择退出。

（2）模块化管理： 每个功能模块（如会员管理、旅游景点管理、酒店信息管理等）通过独立的方法实现，确保各个功能之间的高内聚和低耦合。

（3）输入与验证： 每次用户输入时，程序都会通过 scanner.nextInt() 获取输入并进行处理，同时用 scanner.nextLine() 清除换行符，防止输入错误或遗漏。

（4）异常处理： 通过 try-catch 语句捕捉 SQL 异常，保证系统的稳定性和健壮性。

**2. 功能实现**

（1）管理功能：

用户可以通过该功能模块添加、删除、更新和查找对应的信息。

（2）线路信息查询与会员预定查询功能：

用户可以查询线路信息，也可以通过各种id查询会员预定信息。

（3）门票预定统计：用户也通过输入想要预定的门票数量来预定门票，而门票数量也会自动更新。

（4）最受欢迎旅游线路统计：通过 MostPopularRoute.getMostPopularRoute() 方法，用户可以查看最受欢迎的旅游线路。这可以帮助平台运营方分析旅游市场的趋势，做出相应的调整。

**3. 数据库交互**

程序涉及与数据库的交互操作，每个数据库操作都使用 SQL 语句进行查询或更新，并通过 JDBC 进行连接。

## 7.3 需完善之处

**1. 用户输入与验证**

目前的程序在用户输入时只通过 scanner.nextInt() 等方法获取用户输入，但没有对输入的有效性进行严格验证。在接受用户输入时，应该对用户输入的合法性进行校验。确保数字输入符合预期范围（如预定票数、会员等级等），避免用户输入非法数据导致程序异常。

**2. 异常处理的优化**

虽然程序已经包含了基本的异常处理，但对于一些常见错误（如数据库连接失败、SQL语句错误等），可以考虑更精细的异常处理机制。

**3. 数据库优化**

目前的程序通过 JDBC 进行数据库操作，但是没有实现数据库连接池管理。每次执行数据库操作时都要重新建立数据库连接，这会影响程序性能，尤其在并发访问量较大时。