

**学生课程设计报告**

实验课名称：数据库原理

课程设计名称：Python实现连接数据库，查询教师信息

专业名称：人工智能

班级： 2022240401

学号： 2022905226

学生姓名：

教师姓名：

**\_\_**2024**\_\_**年**\_\_**11**\_\_**月**\_\_**20**\_\_**日

目录

[**一、课设名称** 3](#_Toc183160580)

[**二、课设内容** 3](#_Toc183160581)

[**三、课设要求** 3](#_Toc183160582)

[**四、设计过程和结果** 4](#_Toc183160583)

[**4.1** **需求分析和设计思路** 4](#_Toc183160584)

[**4.1.1**  **需求分析** 4](#_Toc183160585)

[**4.1.2 ER图设计** 5](#_Toc183160586)

[**4.1.3**  **实体关系说明** 5](#_Toc183160587)

[**4.1.4 设计说明** 6](#_Toc183160588)

[**4.1.5**  **命令行实现** 8](#_Toc183160589)

[**4.1.6 Web界面实现** 8](#_Toc183160590)

[**4.2** **查询操作具体实现** 10](#_Toc183160591)

[**4.2.1**  **数据库连接** 10](#_Toc183160592)

[**4.2.2**  **教师查询** 11](#_Toc183160593)

[**4.2.3**  **教学记录查询** 11](#_Toc183160594)

[**4.2.3.1 教学记录查询sql语句分析：** 12](#_Toc183160595)

[**4.3** **Web端设计** 13](#_Toc183160596)

[**4.3.1**  **Web界面布局** 13](#_Toc183160597)

[**4.3.2**  **路由实现** 14](#_Toc183160598)

[**4.3.3**  **会话管理** 15](#_Toc183160599)

[**4.4** **异常处理** 15](#_Toc183160600)

[**4.4.1**  **数据库连接异常处理** 15](#_Toc183160601)

[**4.4.2**  **用户输入验证** 16](#_Toc183160602)

[**4.4.3**  **会话状态异常处理** 16](#_Toc183160603)

[**4.4.4**  **资源释放处理** 16](#_Toc183160604)

[**4.4.5**  **Flask错误提示** 16](#_Toc183160605)

[**4.4.6**  **异常恢复** 17](#_Toc183160606)

[**五、课设设计总结和心得体会** 17](#_Toc183160607)

[**六、附录** 18](#_Toc183160608)

[**6.1** **数据库代码** 18](#_Toc183160609)

[**6.2** **系统设计核心代码** 20](#_Toc183160610)

# **一、课设名称**

Python实现连接数据库，查询教师信息

# **二、课设内容**

请编写一段python程序，它允许大学管理员来打印一位教师的教学记录。

1. 首先让用户输入登录ID和密码，然后打开正确的连接。
2. 接下来要求用户输入一个搜索子字符串，并且系统返回姓名与该子字符串匹配的教师的(ID,name)对。使用SQL中的like(‘%子字符串%’)结构来执行此操作。如果搜索返回为空，则允许继续搜索，直到出现非空结果为止。
3. 然后要求用户输人一个ID编号，它是介于0和99999之间的一个数字。一旦用户输入了一个有效数字，检查是否存在具有该ID的教师。如果没有具有给定ID的教师，则打印合理的消息并退出。
4. 如果该教师没有讲授任何课程，请打印一条消息说明这一点。否则打印该教师的教学记录，显示系的名称、课程标识、课程名称、课程段号、学期、年份和总注册人数(并按dept\_name、course\_id、year、semester对那些记录进行排序)。

请仔细检测错误的输人。确保你的SQL查询不会引发异常。在登录时可能会发生异常，因为用户可能键入了错误的密码，但请捕获这些异常并允许用户重试。

# **三、课设要求**

使用Python连接数据库，按照题目要求，让用户完成对数据库的查询操作，并具有异常处理机制。

# **四、设计过程和结果**

## **4.1 需求分析和设计思路**

### **4.1.1 需求分析**

**业务需求**

根据题目要求，系统需要实现以下功能：

- 教师信息查询（按姓名和ID）

- 教师教学记录查询

- 课程选课人数统计

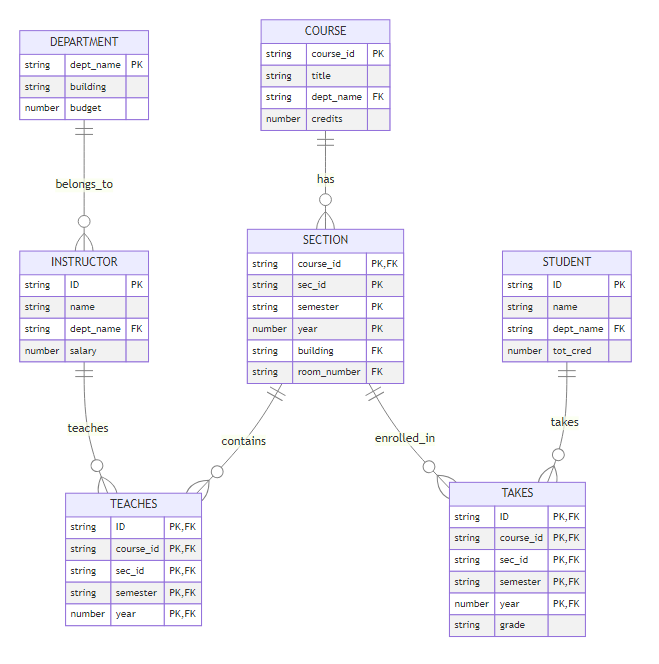
**数据需求**

1. **教师信息**
   * 教师ID（唯一标识）
   * 教师姓名
   * 所属系名
   * 工资信息
2. **课程信息**
   * 课程ID（唯一标识）
   * 课程名称
   * 所属系名
   * 学分
3. **课程段信息**
   * 课程ID
   * 段号
   * 学期
   * 年份
   * 教室信息
4. **教学关系**
   * 教师与课程段的对应关系
   * 一个教师可以教授多个课程段
   * 一个课程段可以由一个教师教授
5. **选课信息**
   * 学生选课记录
   * 用于统计课程注册人数

**数据约束**

* 教师ID范围：0-99999
* 学期限制：Fall、Winter、Spring、Summer
* 年份范围：1701-2100

### **4.1.2 ER图设计**



### **4.1.3 实体关系说明**

1. **INSTRUCTOR-TEACHES关系**
   * 一个教师可以教授多个课程段
   * 关系类型：一对多
2. **DEPARTMENT-INSTRUCTOR关系**
   * 一个系可以有多个教师
   * 每个教师属于一个系
   * 关系类型：一对多
3. **COURSE-SECTION关系**
   * 一个课程可以有多个课程段
   * 关系类型：一对多
4. **SECTION-TEACHES关系**
   * 一个课程段由一个教师教授
   * 关系类型：一对一
5. **SECTION-TAKES关系**
   * 一个课程段可以有多个学生选修
   * 一个学生可以选修多个课程段
   * 关系类型：多对多

### **4.1.4 设计说明**

1. **主键设计**
   * INSTRUCTOR: ID作为主键
   * DEPARTMENT: dept\_name作为主键
   * COURSE: course\_id作为主键
   * SECTION: (course\_id, sec\_id, semester, year)作为复合主键
2. **外键设计**
   * INSTRUCTOR.dept\_name -> DEPARTMENT.dept\_name
   * COURSE.dept\_name -> DEPARTMENT.dept\_name
   * TEACHES包含来自INSTRUCTOR和SECTION的外键
3. **实体属性**
   * 选择了必要的属性来支持查询需求
   * 包含了足够的信息来统计选课人数

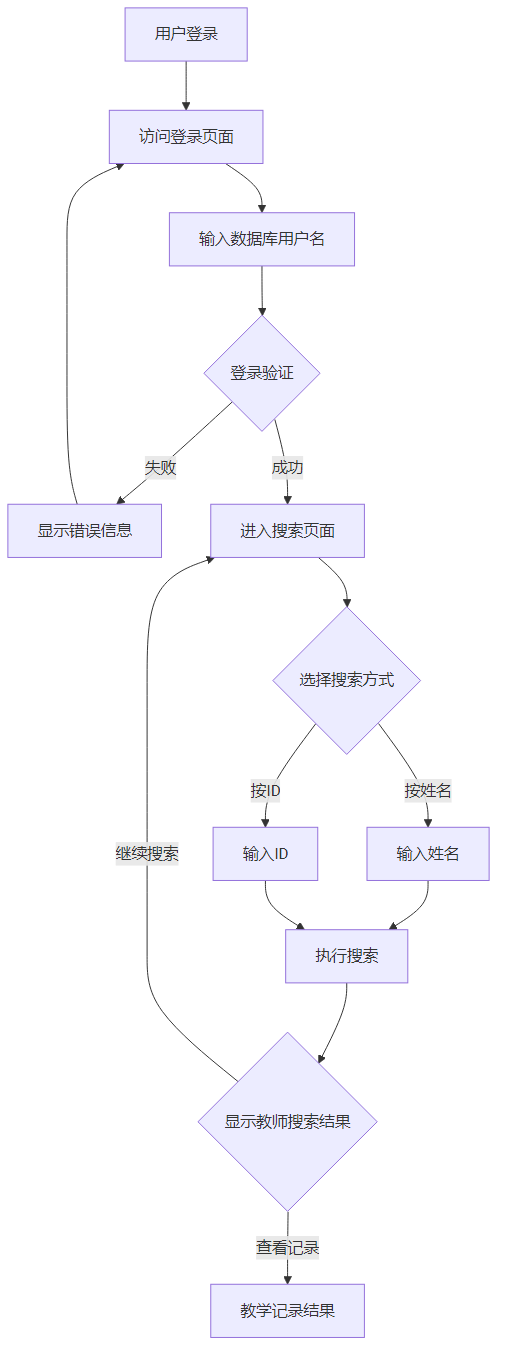
采用了两种实现方式：**命令行基础界面**和**Web界面**。两种方式都实现了相同的核心功能。

文件结构如下：

project/  
├── app.py # Flask Web应用主程序  
│ # 包含路由和后端逻辑  
│  
├── instructor\_teaching\_record.py # 命令行版本的实现程序   
│  
├── static/ # 静态文件目录  
│ └── style.css # 网页样式表  
│ # 定义页面布局和样式  
│  
└── templates/ # HTML模板目录  
 ├── login.html # 登录页面模板  
 ├── search.html # 搜索页面模板  
 └── record.html # 教学记录显示页面模板

核心处理数据库内容和查询操作的程序逻辑基本相同。

程序流程如下：



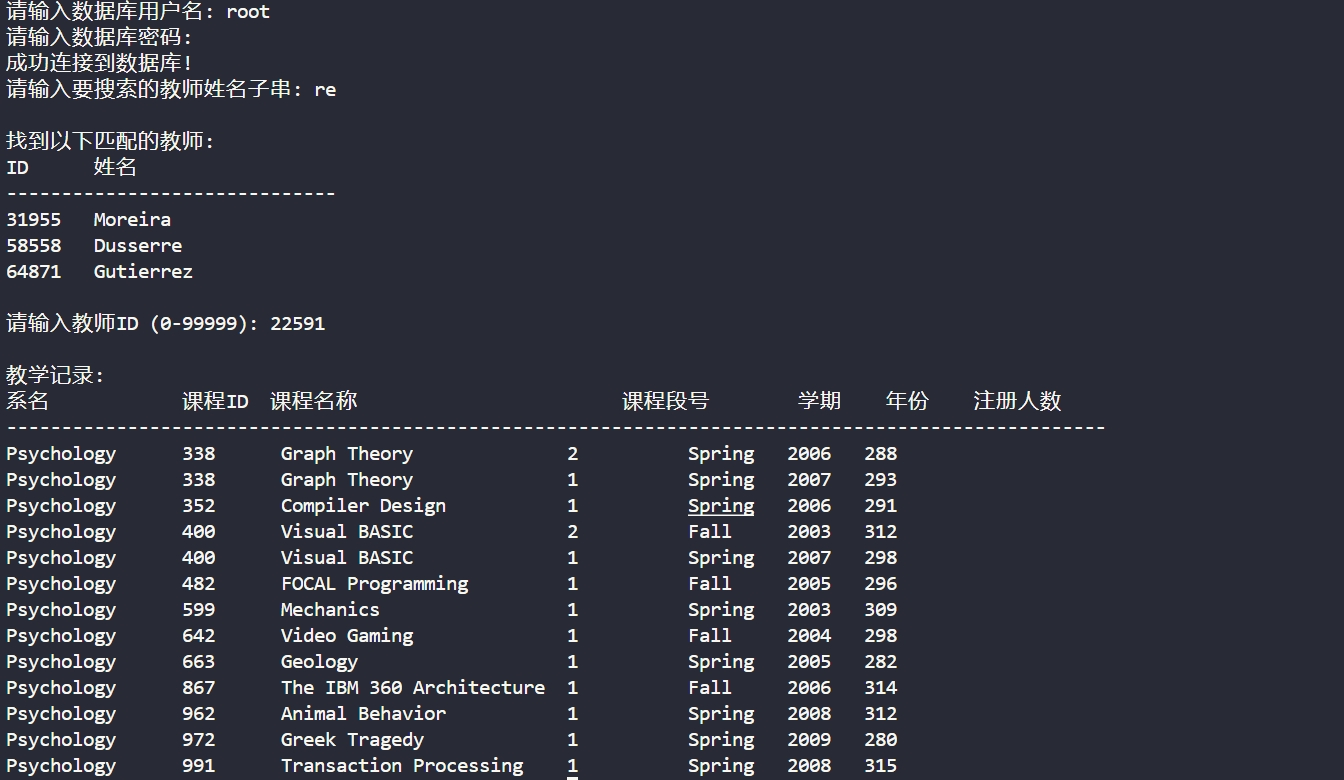
**图1：程序流程**

### **4.1.5 命令行实现**

首先实现了基础的命令行版本（instructor\_teaching\_record.py），主要包含以下功能模块：

* 数据库连接模块：处理用户登录和数据库连接，直接用pymysql库实现连接
* 教师搜索模块：支持按姓名模糊搜索
* ID验证模块：处理教师ID的输入和验证
* 教学记录查询模块：获取并展示教师的教学记录并按要求排序

命令行实现结果：



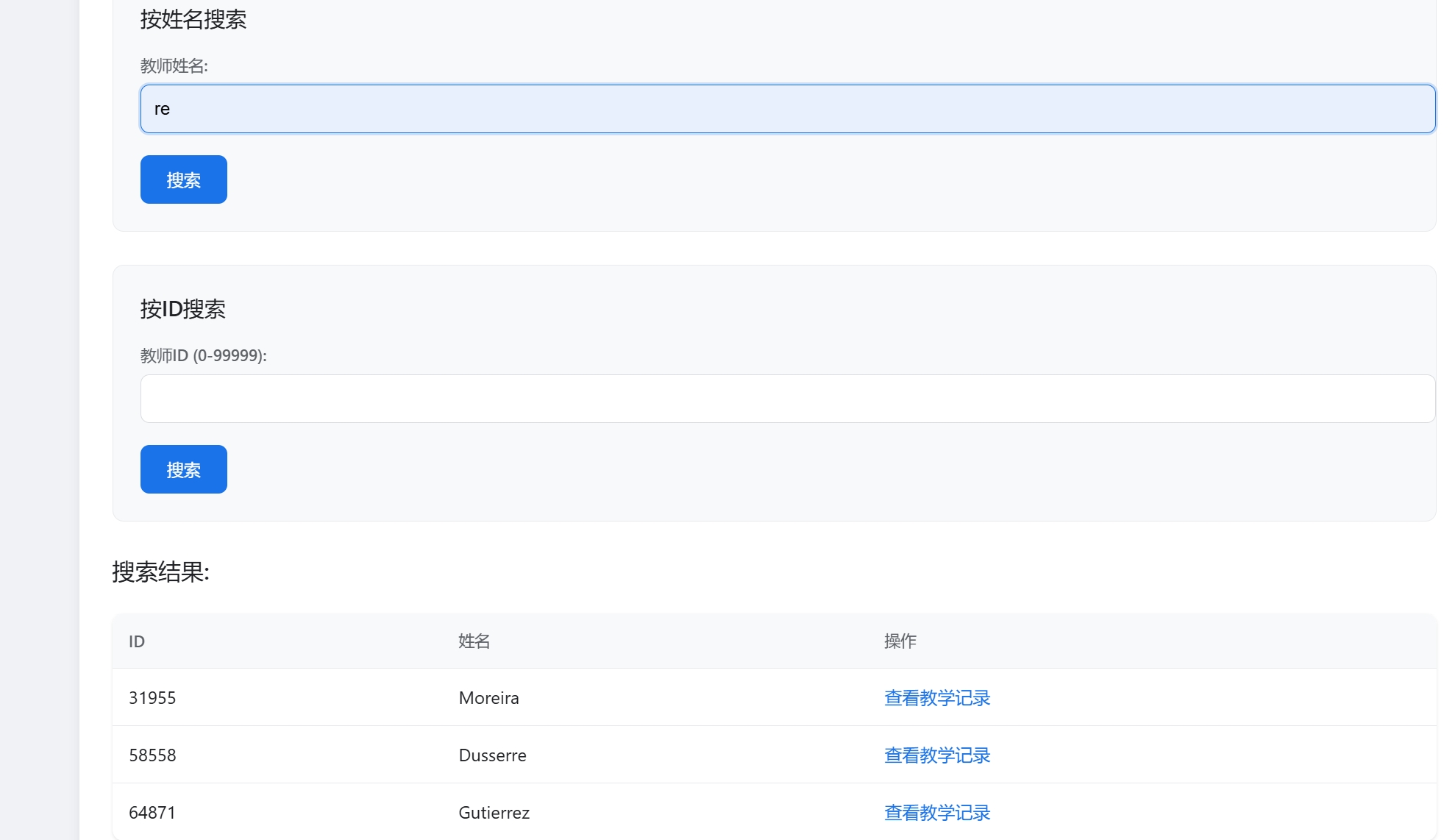
**图2：命令行实现结果**

### **4.1.6 Web界面实现**

在命令行版本的基础上，我使用**Flask框架**开发了Web版本，提供了更友好的用户界面：

* 前端：使用HTML、CSS构建用户界面，CSS文件美化页面格式，html提供网页模板
* 后端：使用Flask处理路由和请求
* 数据库交互：使用PyMySQL进行数据库操作
* 会话管理：实现用户登录状态维护

web页面实现结果：



**图3：按姓名查询结果**



**图4：按ID查询结果**



**图5：教学记录结果**

## **4.2 查询操作具体实现**

### **4.2.1 数据库连接**

def connect\_to\_database():  
 """连接到数据库,返回连接对象"""  
 while True:  
 try:  
 username = input("请输入数据库用户名: ")  
 password = getpass("请输入数据库密码: ")  
   
 connection = pymysql.connect(  
 host='localhost',  
 user=username,  
 password=password,  
 database='uni', # 假设数据库名为university  
 charset='utf8mb4'  
 )  
 print("成功连接到数据库!")  
 return connection  
 except pymysql.Error as e:  
# 错误处理...

通过pymysql实现和数据库的连接，用户输入用户名和密码。

### **4.2.2 教师查询**

实现了两种查询方式：

1. 按姓名模糊查询：

SELECT ID, name FROM instructor WHERE name LIKE %s

1. 按ID精确查询：

SELECT ID, name FROM instructor WHERE ID = %s

查询代码主要由python函数直接将sql语句封装，发送到数据库执行sql语句，再将查询结果读出来即可，比如按姓名模糊查询：

def search\_instructor(connection):  
 """搜索教师并返回匹配的结果"""  
 cursor = connection.cursor()  
   
 while True:  
 search\_string = input("请输入要搜索的教师姓名子串: ")  
 try:  
 query = """  
 SELECT ID, name   
 FROM instructor   
 WHERE name LIKE %s  
 """  
 cursor.execute(query, f"%{search\_string}%")  
 results = cursor.fetchall()  
   
 if results:  
 print("\n找到以下匹配的教师:")  
 print("ID\t姓名")  
 print("-" \* 30)  
 for id, name in results:  
 print(f"{id}\t{name}")  
 return True  
 else:  
 print("未找到匹配的教师")  
 continue

将查询的sql语句存储在query中，之后再连接数据并在其中执行语句即可。

### **4.2.3 教学记录查询**

核心查询语句：

SELECT i.dept\_name, c.course\_id, c.title, s.sec\_id,   
 s.semester, s.year, COUNT(t.ID) as enrollment  
FROM instructor i  
JOIN teaches te ON i.ID = te.ID  
JOIN section s ON te.course\_id = s.course\_id   
 AND te.sec\_id = s.sec\_id   
 AND te.semester = s.semester   
 AND te.year = s.year  
JOIN course c ON s.course\_id = c.course\_id  
LEFT JOIN takes t ON s.course\_id = t.course\_id   
 AND s.sec\_id = t.sec\_id   
 AND s.semester = t.semester   
 AND s.year = t.year  
WHERE i.ID = %s  
GROUP BYi.dept\_name, c.course\_id, c.title, s.sec\_id, s.semester, s.year  
ORDER BY i.dept\_name, c.course\_id, s.year, s.semester

#### **4.2.3.1 教学记录查询sql语句分析：**

1. **表连接关系**：
   * 从 instructor(i) 表开始，获取教师**基本**信息
   * 通过 teaches(te) 表获取教师教授的**课程**信息
   * 通过 section(s) 表获取具体**课程段**信息
   * 通过 course(c) 表获取**课程详细**信息
   * 通过 takes(t) 表获取**选课学生**信息
2. **连接类型**：
   * JOIN teaches: 内连接，获取教师实际教授的课程
   * JOIN section: 内连接，获取课程段详细信息
   * JOIN course: 内连接，获取课程名称等信息
   * LEFT JOIN takes: 左连接，即使没有学生选课也会显示课程信息
3. **连接条件**：
   * teaches 和 section 的连接使用了复合条件：
   * te.course\_id = s.course\_id   
     AND te.sec\_id = s.sec\_id   
     AND te.semester = s.semester   
     AND te.year = s.year
   * 确保课程段信息的完全匹配
   * COUNT(t.ID) as enrollment: 计算每个课程段的选课人数
   * 使用 GROUP BY 子句确保正确统计每个课程段的学生数
4. **结果排序**：

* ORDER BY i.dept\_name, c.course\_id, s.year, s.semester
* 按照系名、课程ID、年份和学期进行多级排序，符合题目要求

1. **查询结果字段**：
   * dept\_name: 系的名称
   * course\_id: 课程标识
   * title: 课程名称
   * sec\_id: 课程段号
   * semester: 学期
   * year: 年份
   * enrollment: 选课人数

查询语句确保能够准确获取教师的完整教学记录，并按要求计算选课人数和排序结果。使用LEFT JOIN确保即使没有学生选课的课程段也会显示在结果中，enrollment值为0。

## **4.3 Web端设计**

### **4.3.1 Web界面布局**

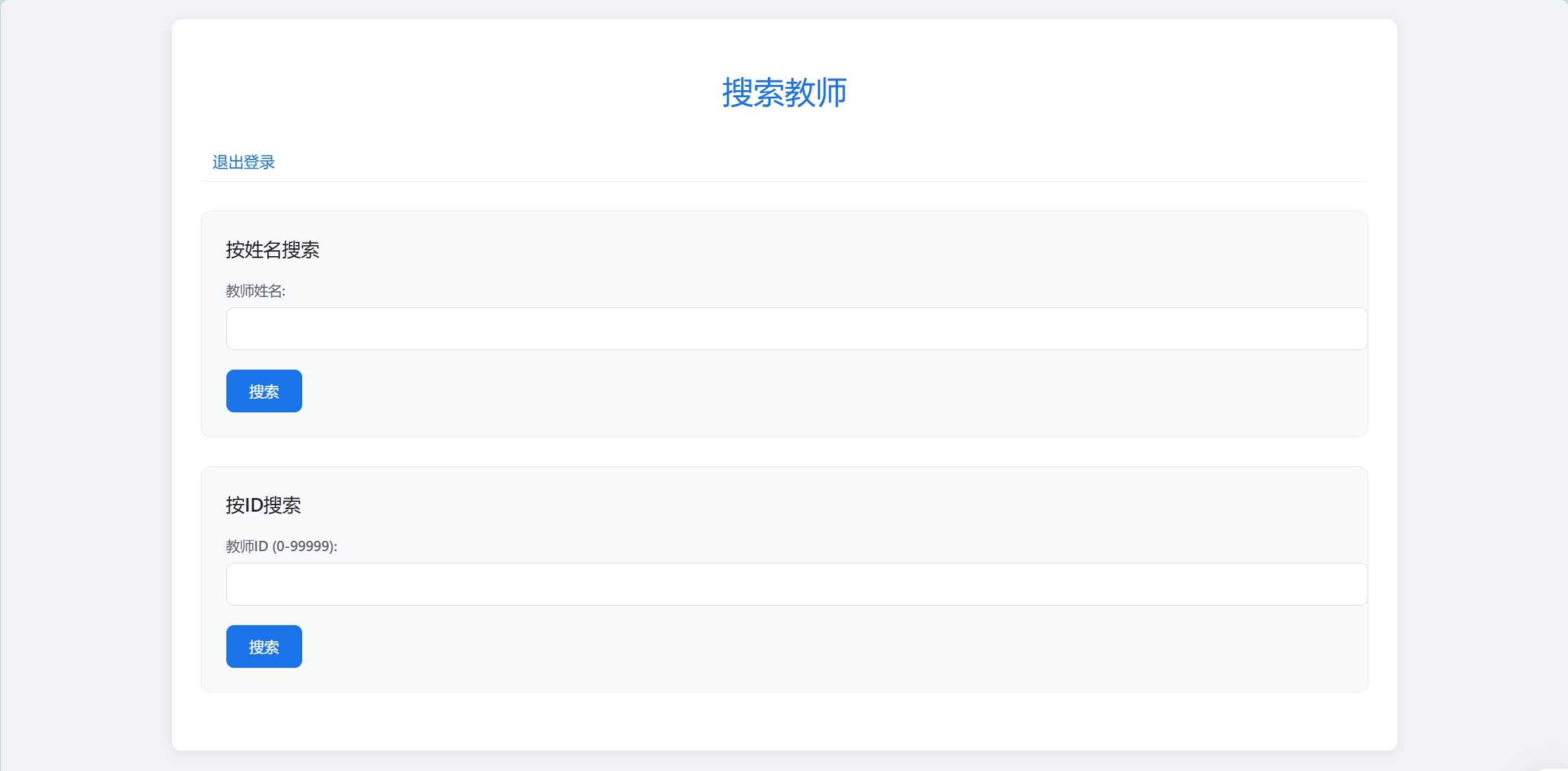
* 登录页面：用户名密码输入
* 搜索页面：支持姓名搜索和ID搜索两种方式
* 结果显示页面：展示查询到的教师和其教学记录

登录页面：



**图6：登录页面**

查询页面：



**图7：查询页面**

### **4.3.2 路由实现**

Web应用中实现了以下主要路由：

1. 登录路由：

@app.route('/', methods=['GET'])  
def login():  
 return render\_template('login.html')  
   
@app.route('/login', methods=['POST'])  
def do\_login():  
 username = request.form['username']  
 password = request.form['password']  
 # 处理登录逻辑...

1. 搜索路由：

@app.route('/search', methods=['GET', 'POST'])  
@login\_required  
def search():  
 if request.method == 'POST':  
 search\_type = request.form['search\_type']  
 # 处理搜索逻辑...

1. 教学记录路由：

@app.route('/record/<instructor\_id>')  
@login\_required  
def teaching\_record(instructor\_id):  
 # 获取并显示教学记录...

1. 登出路由：

@app.route('/logout')  
def logout():  
 session.clear()  
 return redirect(url\_for('login'))

利用**Flask框架**搭建的后端的端口号固定为5000，开启后端之后，直接在网页中访问： http::localhost:5000网址即可。这相当于访问了本地网络。

### **4.3.3 会话管理**

使用Flask的session机制实现用户登录状态维护：

app.secret\_key = 'your\_secret\_key' # 用于session加密  
   
def login\_required(f):  
 @wraps(f)  
 def decorated\_function(\*args, \*\*kwargs):  
 if 'logged\_in' not in session:  
 return redirect(url\_for('login'))  
 return f(\*args, \*\*kwargs)  
 return decorated\_function

## **4.4 异常处理**

本项目实现了完整的异常处理机制，主要包括：

### **4.4.1 数据库连接异常处理**

try:  
 connection = pymysql.connect(  
 \*\*DB\_CONFIG,  
 user=username,  
 password=password  
 )  
except pymysql.Error as e:  
 flash(f'登录失败: {str(e)}', 'error')  
 return redirect(url\_for('login'))

处理的异常情况包括：

* 用户名或密码错误
* 数据库服务未启动
* 网络连接问题
* 数据库权限问题

### **4.4.2 用户输入验证**

# ID输入验证  
try:  
 id\_num = int(instructor\_id)  
 if 0 <= id\_num <= 99999:  
 query = "SELECT ID, name FROM instructor WHERE ID = %s"  
 cursor.execute(query, (instructor\_id,))  
 else:  
 flash('ID必须在0到99999之间', 'error')  
 return render\_template('search.html')  
except ValueError:  
 flash('请输入有效的数字ID', 'error')  
 return render\_template('search.html')

验证内容包括：

* ID格式验证（必须为数字）
* ID范围验证（0-99999）
* 搜索字符串非空验证
* 表单数据完整性验证

### **4.4.3 会话状态异常处理**

@login\_required  
def decorated\_function(\*args, \*\*kwargs):  
 if 'logged\_in' not in session:  
 return redirect(url\_for('login'))  
 return f(\*args, \*\*kwargs)

处理的情况包括会话过期、未登录访问、非法访问保护路由或者会话状态丢失等

### **4.4.4 资源释放处理**

try:  
 # 数据库操作  
finally:  
 connection.close()

确保数据库连接正确关闭，系统资源及时回收

### **4.4.5 Flask错误提示**

使用Flask的flash消息系统提供友好的错误提示：

flash(f'错误信息', 'error')

错误提示包括登录失败提示、查询无结果提示、输入验证错误提示、系统错误提示等

### **4.4.6 异常恢复**

* 登录失败允许重试
* 搜索无结果可以继续搜索
* 输入错误可以重新输入
* 会话过期自动跳转登录

通过这些异常处理机制，确保了程序的稳定性、安全性和可维护性。

# **五、课设设计总结和心得体会**

**项目总结**

本次课程设计实现了一个教师教学记录查询系统，主要完成了：

1. **双版本实现**：
   * 完成了命令行版本的基础功能实现
   * 基于Flask框架开发了Web版本，提供了更友好的用户界面
2. **功能完整性**：
   * 实现了用户登录验证
   * 支持按姓名模糊搜索和ID精确搜索
   * 完整展示教师教学记录
   * 按要求实现了结果排序
3. **异常处理**：
   * 完善的输入验证机制
   * 数据库操作异常处理
   * 用户友好的错误提示

**心得体会与收获**

通过本次课程设计，我在以下几个方面有了深刻的体会和收获：

1. **数据库应用开发能力的提升**

通过本次设计我更加熟悉了 PyMySQL 库进行数据库操作的方法，深入理解了复杂的多表连接查询原理，查询教学记录时需要将多个表同时连接获得完整信息。 另外在实际应用中需要设计和优化 SQL 查询语句，理解了数据库规范化设计对查询效率的影响。

1. **Web开发技术的学习**

通过此次设计我也学习了 Flask 框架的使用，理解了 MVC 架构。简单掌握了前后端分离开发的基本思想，通过设计一个简单但是不失美观的界面，提升了自我的 HTML/CSS 的编写能力。

1. **工程模块化和异常处理**

我学会了模块化的程序设计方法，掌握了异常处理的重要性和实现方式。将工程不同功能模块化，不仅有利于提升代码可读性，还会很容易扩展。比如在此次设计中，我需要将数据库操作和路由处理的功能分离开，分离开的好处就是便于调试和修改。

对于异常处理，实际上mysql和python本身提供了大量的异常处理机制，这需要我在对应的地方应用到才可以。

通过本次课程设计，我不仅将数据库理论知识应用到了实践中，还学习了完整的软件开发流程。从需求分析到具体实现，再到用户界面设计，每个环节都让我对软件开发有了更深的认识。这些经验对我今后的学习和工作都有重要的帮助。

# **六、附录**

## **6.1 数据库代码**

**数据库构建：**

**create** **table** department

    (dept\_name      **varchar**(20),

     building       **varchar**(15),

     budget             **numeric**(12,2) **check** (budget > 0),

**primary key** (dept\_name)

    );

**create** **table** course

    (course\_id      **varchar**(8),

     title          **varchar**(50),

     dept\_name      **varchar**(20),

     credits        **numeric**(2,0) **check** (credits >= 0),

**primary key** (course\_id),

**constraint** fk\_dept\_name **foreign key** (dept\_name) **references** department(dept\_name)

**on delete** **set** **null**

    );

**create** **table** instructor

    (ID         **varchar**(5),

**name**           **varchar**(20) **not null**,

     dept\_name      **varchar**(20),

     salary         **numeric**(8,2) **check** (salary > 29000),

**primary key** (ID),

**constraint** fk\_instructor\_dept **foreign key** (dept\_name) **references** department(dept\_name)

**on delete** **set** **null**

    );

**create** **table** section

    (course\_id      **varchar**(8),

     sec\_id         **varchar**(8),

     semester       **varchar**(6)

**check** (semester **in** ('Fall', 'Winter', 'Spring', 'Summer')),

**year**           **numeric**(4,0) **check** (**year** > 1701 **and** **year** < 2100),

     building       **varchar**(15),

     room\_number        **varchar**(7),

     time\_slot\_id       **varchar**(4),

**primary key** (course\_id, sec\_id, semester, **year**),

**constraint** fk\_section\_course **foreign key** (course\_id) **references** course(course\_id)

**on delete cascade**,

**constraint** fk\_section\_classroom **foreign key** (building, room\_number) **references** classroom(building, room\_number)

**on delete** **set** **null**

    );

**create** **table** teaches

    (ID         **varchar**(5),

     course\_id      **varchar**(8),

     sec\_id         **varchar**(8),

     semester       **varchar**(6),

**year**           **numeric**(4,0),

**primary key** (ID, course\_id, sec\_id, semester, **year**),

**constraint** fk\_teaches\_section **foreign key** (course\_id,sec\_id, semester, **year**) **references** section(course\_id,sec\_id, semester, **year**)

**on delete cascade**,

**constraint** fk\_teaches\_instructor **foreign key** (ID) **references** instructor(ID)

**on delete cascade**

    );

**create** **table** student

    (ID         **varchar**(5),

**name**           **varchar**(20) **not null**,

     dept\_name      **varchar**(20),

     tot\_cred       **numeric**(3,0) **check** (tot\_cred >= 0),

**primary key** (ID),

**constraint** fk\_student\_dept **foreign key** (dept\_name) **references** department(dept\_name)

**on delete** **set** **null**

    );

**create** **table** takes

    (ID         **varchar**(5),

     course\_id      **varchar**(8),

     sec\_id         **varchar**(8),

     semester       **varchar**(6),

**year**           **numeric**(4,0),

     grade              **varchar**(2),

**primary key** (ID, course\_id, sec\_id, semester, **year**),

**constraint** fk\_takes\_section **foreign key** (course\_id,sec\_id, semester, **year**) **references** section(course\_id,sec\_id, semester, **year**)

**on delete cascade**,

**constraint** fk\_takes\_student **foreign key** (ID) **references** student(ID)

**on delete cascade**

    );

**数据库查询语句：**

SELECT i.dept\_name, c.course\_id, c.title, s.sec\_id,

                   s.semester, s.year, COUNT(t.ID) as enrollment

            FROM instructor i

            JOIN teaches te ON i.ID = te.ID

            JOIN section s ON te.course\_id = s.course\_id

                AND te.sec\_id = s.sec\_id

                AND te.semester = s.semester

                AND te.year = s.year

            JOIN course c ON s.course\_id = c.course\_id

            LEFT JOIN takes t ON s.course\_id = t.course\_id

                AND s.sec\_id = t.sec\_id

                AND s.semester = t.semester

                AND s.year = t.year

            WHERE i.ID = %s

            GROUP BY i.dept\_name, c.course\_id, c.title, s.sec\_id, s.semester, s.year

            ORDER BY i.dept\_name, c.course\_id, s.year, s.semester

## **6.2 系统设计核心代码**

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'  # 用于session和flash消息

# 数据库配置

DB\_CONFIG = {

    'host': 'localhost',

    'database': 'uni',

    'charset': 'utf8mb4'

}

# 登录配置

**def** login\_required(f):

    @wraps(f)

**def** decorated\_function(\*args, \*\*kwargs):

**if** 'logged\_in' not in session:

**return** redirect(url\_for('login'))

**return** f(\*args, \*\*kwargs)

**return** decorated\_function

@app.route('/', methods=['GET'])

**def** login():

**return** render\_template('login.html')

@app.route('/login', methods=['POST'])

**def** do\_login():

    username = request.form['username']

    password = request.form['password']

**try**:

        connection = pymysql.connect(

            \*\*DB\_CONFIG,

            user=username,

            password=password

        )

        session['logged\_in'] = **True**

        session['db\_username'] = username

        session['db\_password'] = password

        connection.close()

**return** redirect(url\_for('search'))

**except** pymysql.Error **as** e:

        flash(**f**'登录失败: {str(e)}', 'error')

**return** redirect(url\_for('login'))

@app.route('/search', methods=['GET', 'POST'])

@login\_required

**def** search():

**if** request.method == 'POST':

**try**:

            connection = pymysql.connect(

                \*\*DB\_CONFIG,

                user=session['db\_username'],

                password=session['db\_password']

            )

            cursor = connection.cursor()

            search\_type = request.form['search\_type']

**if** search\_type == 'name':

                search\_string = request.form['search\_string']

                query = "SELECT ID, name FROM instructor WHERE name LIKE %s"

                cursor.execute(query, **f**"%{search\_string}%")

**else**:  # search\_type == 'id'

                instructor\_id = request.form['instructor\_id']

**try**:

                    id\_num = int(instructor\_id)

**if** 0 <= id\_num <= 99999:

                        query = "SELECT ID, name FROM instructor WHERE ID = %s"

                        cursor.execute(query, (instructor\_id,))

**else**:

                        flash('ID必须在0到99999之间', 'error')

**return** render\_template('search.html')

**except** ValueError:

                    flash('请输入有效的数字ID', 'error')

**return** render\_template('search.html')

            results = cursor.fetchall()

            connection.close()

**if** not results:

                flash('未找到匹配的教师', 'error')

**return** render\_template('search.html', results=results)

**except** pymysql.Error **as** e:

            flash(**f**'查询错误: {str(e)}', 'error')

**return** render\_template('search.html')

**return** render\_template('search.html')

@app.route('/record/<instructor\_id>')

@login\_required

**def** teaching\_record(instructor\_id):

**try**:

        connection = pymysql.connect(

            \*\*DB\_CONFIG,

            user=session['db\_username'],

            password=session['db\_password']

        )

        cursor = connection.cursor()

        # 检查教师是否存在

        cursor.execute("SELECT name FROM instructor WHERE ID = %s", (instructor\_id,))

**if** not cursor.fetchone():

            flash('未找到该ID的教师', 'error')

**return** redirect(url\_for('search'))

        # 获取教学记录

        query = """

            SELECT i.dept\_name, c.course\_id, c.title, s.sec\_id,

                   s.semester, s.year, COUNT(t.ID) as enrollment

            FROM instructor i

            JOIN teaches te ON i.ID = te.ID

            JOIN section s ON te.course\_id = s.course\_id

                AND te.sec\_id = s.sec\_id

                AND te.semester = s.semester

                AND te.year = s.year

            JOIN course c ON s.course\_id = c.course\_id

            LEFT JOIN takes t ON s.course\_id = t.course\_id

                AND s.sec\_id = t.sec\_id

                AND s.semester = t.semester

                AND s.year = t.year

            WHERE i.ID = %s

            GROUP BY i.dept\_name, c.course\_id, c.title, s.sec\_id, s.semester, s.year

            ORDER BY i.dept\_name, c.course\_id, s.year, s.semester

        """

        cursor.execute(query, (instructor\_id,))

        records = cursor.fetchall()

        connection.close()

**return** render\_template('record.html', records=records)

**except** pymysql.Error **as** e:

        flash(**f**'查询错误: {str(e)}', 'error')

**return** redirect(url\_for('search'))

@app.route('/logout')

**def** logout():

    session.clear()

**return** redirect(url\_for('login'))