|  |
| --- |
|  |

**数据库原理与实践期末项目**

**学号：**

**姓名：**



数据库原理与实践期末项目

20 —20 学年 学期

|  |  |
| --- | --- |
| 课程名称： |  |
| 专业学院： |  |
| 专业班级： |  |
| 学 号： |  |
| 姓 名： |  |
| 指导教师： |  |

目 录

[1、 项目背景及需求分析 1](#_Toc153742448)

[1.1项目背景 1](#_Toc153742449)

[1.2需求分析 1](#_Toc153742450)

[2、 概念结构设计 2](#_Toc153742451)

[2.1抽象出系统实体 2](#_Toc153742452)

[2.2局部E-R图 2](#_Toc153742453)

[2.3全局E-R图 2](#_Toc153742454)

[3、 逻辑结构设计 3](#_Toc153742455)

[3.1简单说明 3](#_Toc153742456)

[3.2关系模式 3](#_Toc153742457)

[3.3表结构 3](#_Toc153742458)

[4、 创建数据库及SQL相关操作 4](#_Toc153742459)

[4.1 创建数据库 4](#_Toc153742460)

[4.1.1 创建表 4](#_Toc153742461)

[4.1.2 添加记录 4](#_Toc153742462)

[4.1.3 修改记录 5](#_Toc153742463)

[4.2 删除记录 5](#_Toc153742464)

[4.3查询问题及查询结果 5](#_Toc153742465)

[4.3.1 选择列 5](#_Toc153742466)

[4.3.2函数使用 5](#_Toc153742467)

[4.3.3 LIKE子句 6](#_Toc153742468)

[4.3.4 条件查询 6](#_Toc153742469)

[4.4.5多表查询 6](#_Toc153742470)

[4.4.6子查询 6](#_Toc153742471)

[4.5建立视图 7](#_Toc153742472)

[4.5.1 定义视图 7](#_Toc153742473)

[4.5.2利用视图插入数据 7](#_Toc153742474)

[5、建立存储过程及函数 8](#_Toc153742475)

[5.1创建存储过程 8](#_Toc153742476)

[5.2存储过程应用 8](#_Toc153742477)

[6、建立触发器 9](#_Toc153742478)

[7、项目设计总结 10](#_Toc153742479)

# 1、 项目背景及需求分析

## 1.1项目背景

随着教育信息化的发展，传统的纸质学生档案管理方式已经不能满足现代高校管理的需求。学生档案管理系统作为高校信息化建设中的重要组成部分，旨在高效管理学生的基本信息、课程信息、成绩信息等数据，提升数据存储、查询和更新的效率，同时减少人工操作中的错误风险。本项目的目标是基于MySQL数据库设计一个学生档案管理系统，提供安全、高效、便捷的数据管理平台，满足高校对于学生档案全面管理的需求。该系统将服务于院系管理人员、教师及相关工作人员，实现数据共享、查询和统计分析功能，为高校管理提供决策支持。

## 1.2需求分析

 **基本信息管理**

* 添加、删除、修改学生档案信息，包括学号、姓名、性别、出生日期、政治面貌、班级编号、毕业学校等。
* 支持班级和专业的查询与管理。

 **课程与成绩管理**

* 管理课程信息，包括课程编号、课程名称、课程类别、学分等。
* 管理学生成绩，记录学生每门课程的考试成绩及考试次数。

 **教师信息管理**

* 记录教师的授课信息，包括课程编号、教师编号、授课地点、班级编号、教师职称等。

 **学院和班级管理**

* 管理学院信息，包括学院编号及学院名称。
* 管理班级信息，包括班级编号、班级名称、班级人数、所属专业等

# 2、 概念结构设计

## 2.1抽象出系统实体

学生档案表:学号、姓名。性别、出生日期、政治面貌、班级编号、毕业学校

专业表:专业名称、学院编号

教师表:课程编号、教师编号、班级编号、授课地点、职称

学生成绩表:学号、课程编号、成绩、考试次数

课程名表:课程编号、课程名、课程类别、学分

班级表:班级编号、班级名称、人数、专业编号

学院表：学院编号，学院名

## 2.2局部E-R图

## 单个实体ER图

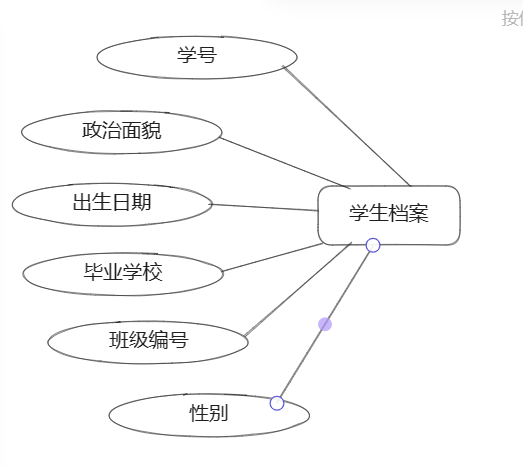


图1：学生档案表

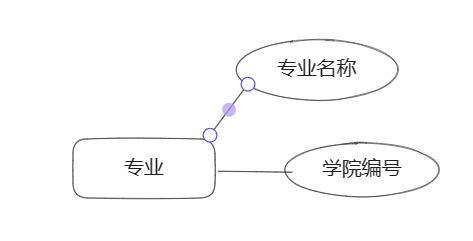


图2：专业表

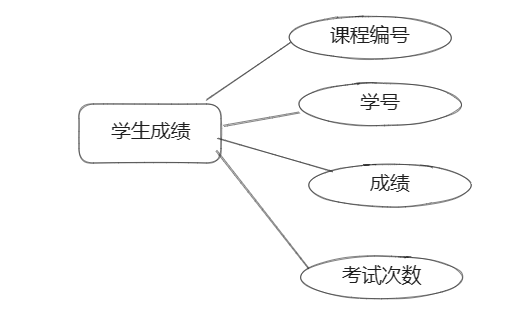


图3：学生成绩表

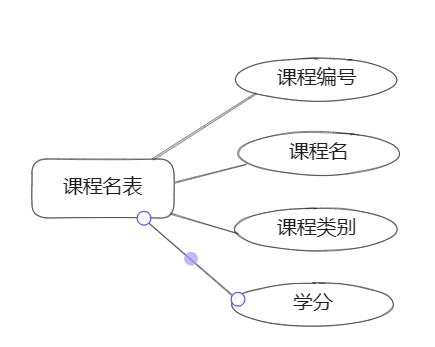


图4：课程名表

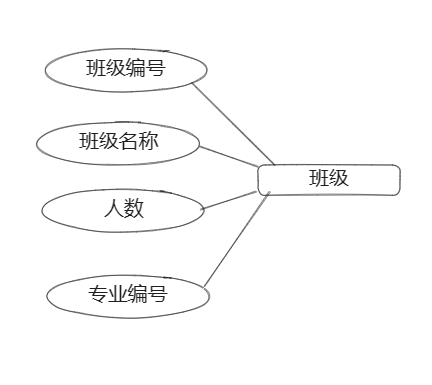


图5：班级表

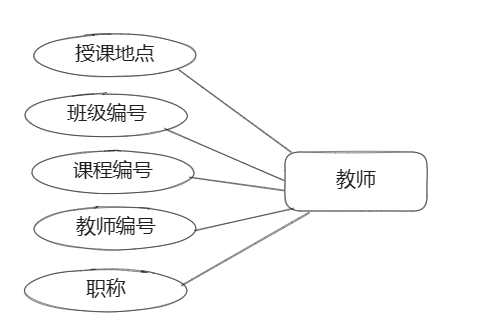


图6：教师表

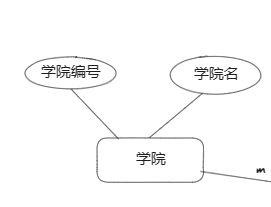
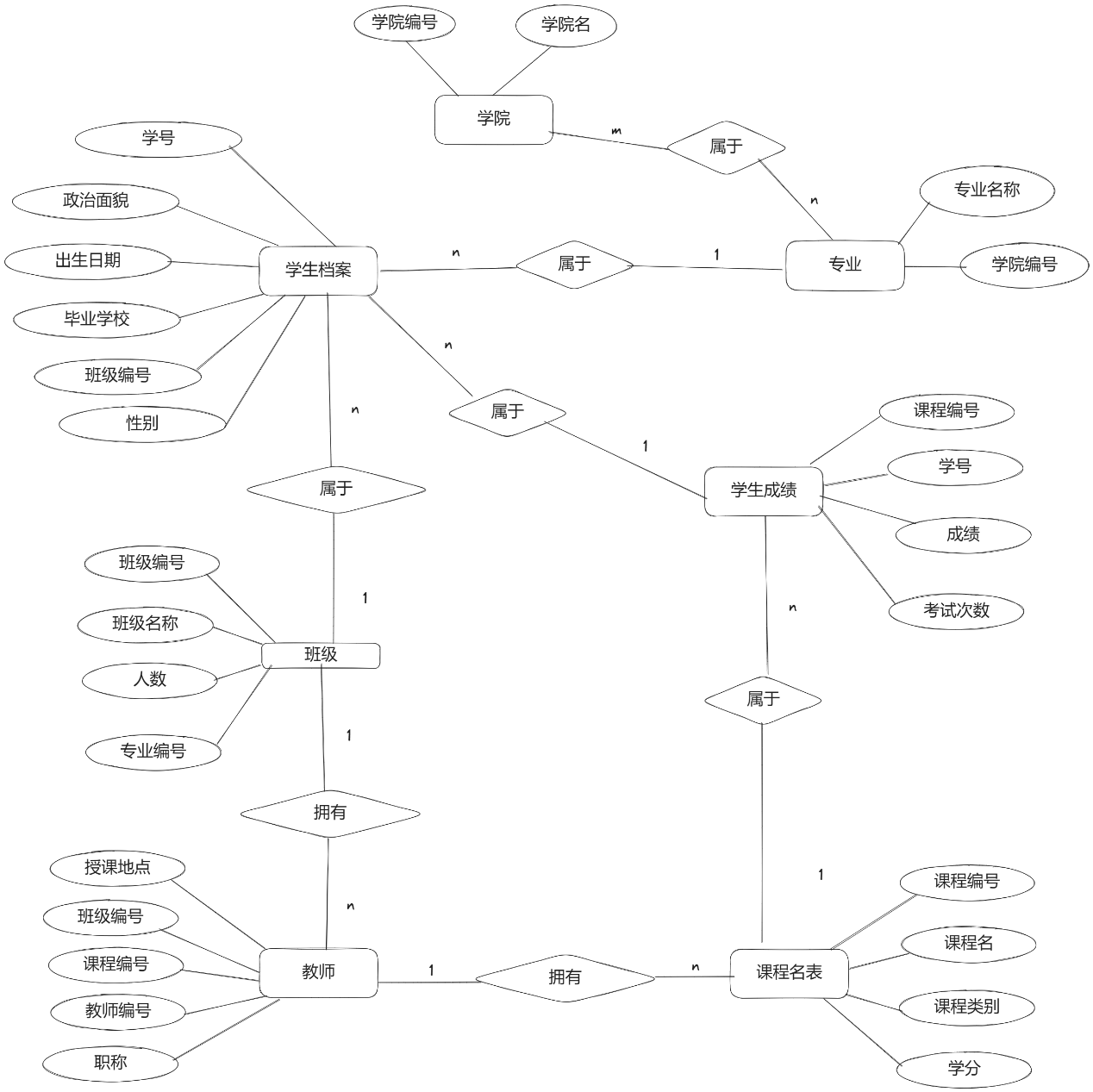


图7：学院表

## 2.3全局E-R图

画出全局E-R图并标明各实体之间的关系。



# 3、 逻辑结构设计

## 3.1简单说明

**1. 学生档案表**

* **表名**：学生档案表
* **字段**：
  + 学号（Primary Key）
  + 姓名
  + 性别
  + 出生日期
  + 政治面貌
  + 班级编号（Foreign Key，关联班级表）
  + 毕业学校
* **描述**：
  + 用于存储学生的基本信息。
  + 通过班级编号与班级表关联。

**2. 专业表**

* **表名**：专业表
* **字段**：
  + 专业名称（Primary Key）
  + 学院编号（Foreign Key，关联学院表）
* **描述**：
  + 用于存储每个专业的名称及其所属学院信息。
  + 通过学院编号与学院表关联。

**3. 教师表**

* **表名**：教师表
* **字段**：
  + 教师编号（Primary Key）
  + 课程编号（Foreign Key，关联课程名表）
  + 班级编号（Foreign Key，关联班级表）
  + 授课地点
  + 职称
* **描述**：
  + 用于存储教师信息及其授课安排。
  + 通过课程编号与课程名表关联，通过班级编号与班级表关联。

**4. 学生成绩表**

* **表名**：学生成绩表
* **字段**：
  + 学号（Foreign Key，关联学生档案表）
  + 课程编号（Foreign Key，关联课程名表）
  + 成绩
  + 考试次数
* **主键**：
  + 学号 + 课程编号（联合主键）
* **描述**：
  + 用于存储学生的课程成绩和考试次数。
  + 通过学号与学生档案表关联，通过课程编号与课程名表关联。

**5. 课程名表**

* **表名**：课程名表
* **字段**：
  + 课程编号（Primary Key）
  + 课程名
  + 课程类别
  + 学分
* **描述**：
  + 用于存储课程的基本信息。
  + 与教师表和学生成绩表关联。

**6. 班级表**

* **表名**：班级表
* **字段**：
  + 班级编号（Primary Key）
  + 班级名称
  + 人数
  + 专业编号（Foreign Key，关联专业表）
* **描述**：
  + 用于存储班级的基本信息。
  + 通过专业编号与专业表关联。

**7. 学院表**

* **表名**：学院表
* **字段**：
  + 学院编号（Primary Key）
  + 学院名
* **描述**：
  + 用于存储学院的基本信息。
  + 与专业表关联。

**表之间的关系描述**

## 3.2关系模式

| **表名** | **关联表名** | **关联字段** | **关系类型** |
| --- | --- | --- | --- |
| 学生档案表 | 班级表 | 班级编号（Foreign Key） | 多对一 |
| 专业表 | 学院表 | 学院编号（Foreign Key） | 多对一 |
| 教师表 | 课程名表、班级表 | 课程编号（Foreign Key），班级编号（Foreign Key） | 多对多 |
| 学生成绩表 | 学生档案表、课程名表 | 学号（Foreign Key），课程编号（Foreign Key） | 多对多 |
| 班级表 | 专业表 | 专业编号（Foreign Key） | 多对一 |

## 3.3表结构

**1. 学生档案表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 学号 | INT | 主键 | 学生唯一编号 |
| 姓名 | VARCHAR(50) |  | 学生姓名 |
| 性别 | VARCHAR(10) |  | 学生性别 |
| 出生日期 | DATE |  | 学生出生日期 |
| 政治面貌 | VARCHAR(20) |  | 学生政治面貌 |
| 班级编号 | INT | 外键 (班级表) | 关联所属班级编号 |
| 毕业学校 | VARCHAR(100) |  | 学生毕业的学校 |

**2. 专业表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 专业名称 | VARCHAR(50) | 主键 | 专业唯一名称 |
| 学院编号 | INT | 外键 (学院表) | 关联所属学院编号 |

**3. 教师表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 教师编号 | INT | 主键 | 教师唯一编号 |
| 课程编号 | INT | 外键 (课程名表) | 关联授课课程编号 |
| 班级编号 | INT | 外键 (班级表) | 关联授课班级编号 |
| 授课地点 | VARCHAR(100) |  | 授课地点 |
| 职称 | VARCHAR(50) |  | 教师职称 |

**4. 学生成绩表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 学号 | INT | 外键 (学生档案表) | 关联学生编号 |
| 课程编号 | INT | 外键 (课程名表) | 关联课程编号 |
| 成绩 | FLOAT |  | 学生成绩 |
| 考试次数 | INT |  | 学生参加考试的次数 |
| 主键 | 学号+课程编号 | 联合主键 | 唯一标识学生课程成绩 |

**5. 课程名表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 课程编号 | INT | 主键 | 课程唯一编号 |
| 课程名 | VARCHAR(100) |  | 课程名称 |
| 课程类别 | VARCHAR(50) |  | 课程类型（必修/选修等） |
| 学分 | FLOAT |  | 课程学分 |

**6. 班级表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 班级编号 | INT | 主键 | 班级唯一编号 |
| 班级名称 | VARCHAR(50) |  | 班级名称 |
| 人数 | INT |  | 班级总人数 |
| 专业编号 | VARCHAR(50) | 外键 (专业表) | 关联班级所属专业编号 |

**7. 学院表**

| **字段名** | **数据类型** | **主键/外键** | **描述** |
| --- | --- | --- | --- |
| 学院编号 | INT | 主键 | 学院唯一编号 |
| 学院名 | VARCHAR(50) |  | 学院名称 |

# 4、 创建数据库及SQL相关操作

## 4.1 创建数据库

建立数据库，数据库名称自取，设置字符集为UTF-8,排序规则为utf8\_general\_ci。

代码：

-- 创建数据库

CREATE DATABASE StudentManagement

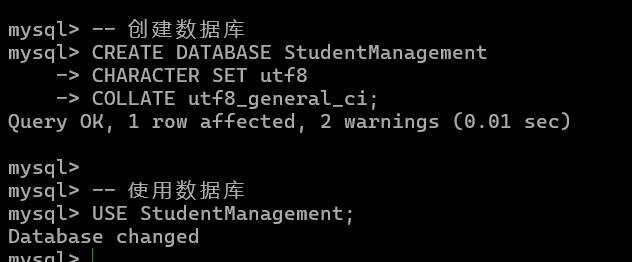
CHARACTER SET utf8

COLLATE utf8\_general\_ci;

-- 使用数据库

USE StudentManagement;

截图结果：



### 4.1.1 创建表

根据需要，在你们创建的数据库中创建你们的每个数据表。

表一

代码：

-- 1. 创建学院表（最基础的表，没有外键）

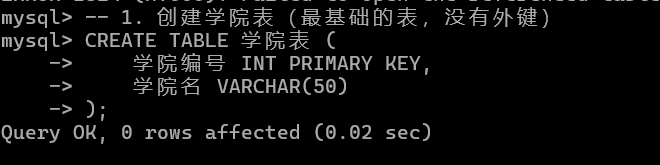
CREATE TABLE 学院表 (

学院编号 INT PRIMARY KEY,

学院名 VARCHAR(50)

);

结果截图：



表二

代码：

-- 2. 创建专业表（引用学院表的学院编号）

CREATE TABLE 专业表 (

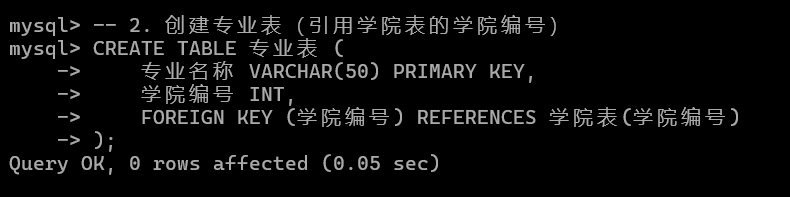
专业名称 VARCHAR(50) PRIMARY KEY,

学院编号 INT,

FOREIGN KEY (学院编号) REFERENCES 学院表(学院编号)

);

结果截图：



表三

代码：

-- 3. 创建班级表（引用专业表的专业名称）

CREATE TABLE 班级表 (

班级编号 INT PRIMARY KEY,

班级名称 VARCHAR(50),

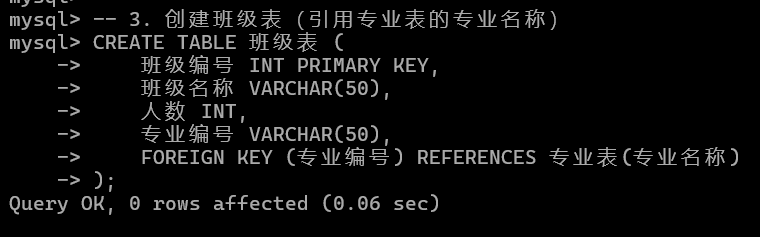
人数 INT,

专业编号 VARCHAR(50),

FOREIGN KEY (专业编号) REFERENCES 专业表(专业名称)

);

结果截图：



表四

代码：

-- 4. 创建课程名表（不依赖于其他表）

CREATE TABLE 课程名表 (

课程编号 INT PRIMARY KEY,

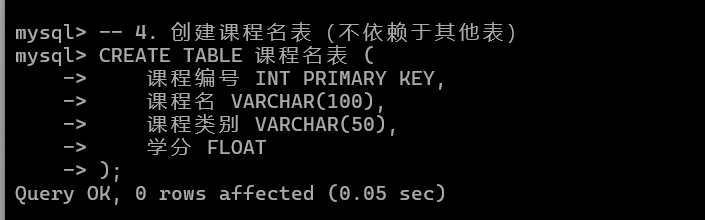
课程名 VARCHAR(100),

课程类别 VARCHAR(50),

学分 FLOAT

);

结果截图：



表五

代码：

-- 5. 创建学生档案表（引用班级表的班级编号）

CREATE TABLE 学生档案表 (

学号 INT PRIMARY KEY,

姓名 VARCHAR(50),

性别 VARCHAR(10) NOT NULL,

出生日期 DATE,

政治面貌 VARCHAR(20),

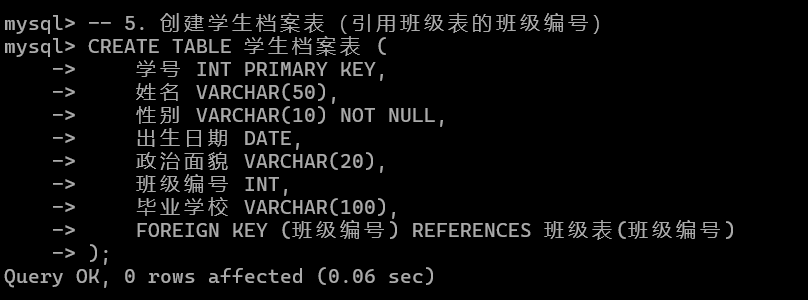
班级编号 INT,

毕业学校 VARCHAR(100),

FOREIGN KEY (班级编号) REFERENCES 班级表(班级编号)

);

结果截图：



表六

代码：

-- 6. 创建教师表（引用课程名表的课程编号和班级表的班级编号）

CREATE TABLE 教师表 (

教师编号 INT PRIMARY KEY,

课程编号 INT,

班级编号 INT,

授课地点 VARCHAR(100),

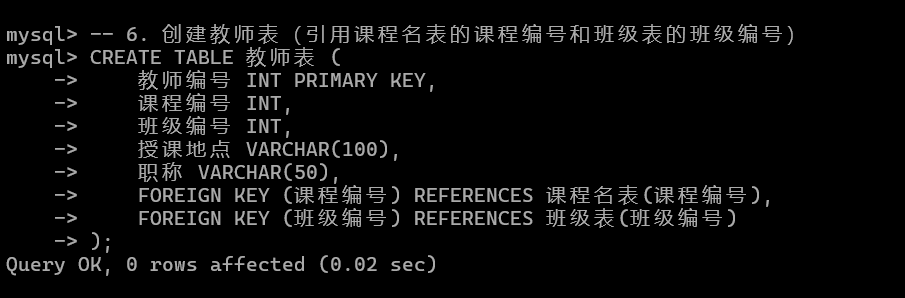
职称 VARCHAR(50),

FOREIGN KEY (课程编号) REFERENCES 课程名表(课程编号),

FOREIGN KEY (班级编号) REFERENCES 班级表(班级编号)

);

结果截图：



表七

代码：

-- 7. 创建学生成绩表（引用学生档案表的学号和课程名表的课程编号）

CREATE TABLE 学生成绩表 (

学号 INT,

课程编号 INT,

成绩 FLOAT,

考试次数 INT,

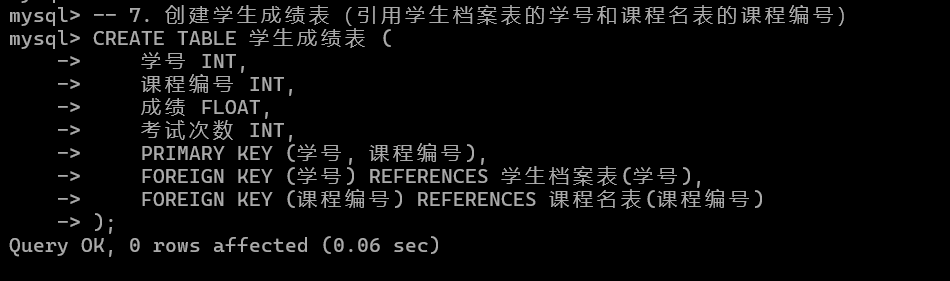
PRIMARY KEY (学号, 课程编号),

FOREIGN KEY (学号) REFERENCES 学生档案表(学号),

FOREIGN KEY (课程编号) REFERENCES 课程名表(课程编号)

);

结果截图：



### 4.1.2 添加记录

-- 插入学院表数据

INSERT INTO 学院表 (学院编号, 学院名) VALUES

(1, '计算机学院'),

(2, '机械学院'),

(3, '电子信息学院'),

(4, '土木工程学院'),

(5, '经济管理学院'),

(6, '外国语学院'),

(7, '化学与环境工程学院'),

(8, '数学与统计学院'),

(9, '艺术与设计学院'),

(10, '医学院');

-- 插入专业表数据

INSERT INTO 专业表 (专业名称, 学院编号) VALUES

('计算机科学与技术', 1),

('软件工程', 1),

('机械设计制造及其自动化', 2),

('通信工程', 3),

('电子信息工程', 3),

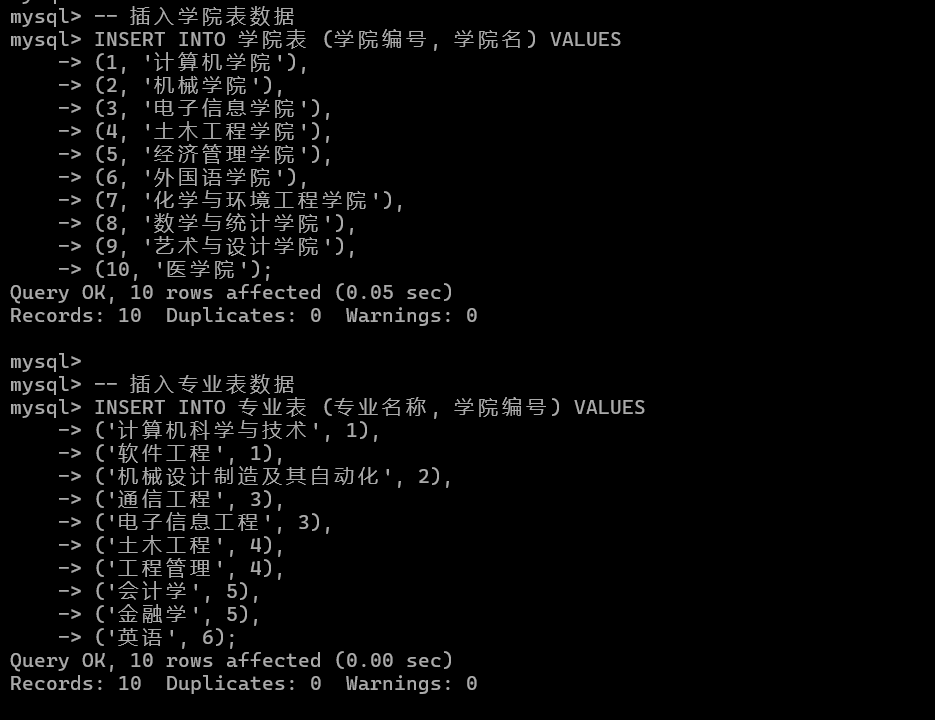
('土木工程', 4),

('工程管理', 4),

('会计学', 5),

('金融学', 5),

('英语', 6);



-- 插入班级表数据

INSERT INTO 班级表 (班级编号, 班级名称, 人数, 专业编号) VALUES

(101, '计算机2001班', 30, '计算机科学与技术'),

(102, '计算机2002班', 28, '计算机科学与技术'),

(103, '软件2001班', 32, '软件工程'),

(201, '机械2001班', 35, '机械设计制造及其自动化'),

(202, '机械2002班', 33, '机械设计制造及其自动化'),

(301, '通信2001班', 25, '通信工程'),

(302, '电子2001班', 27, '电子信息工程'),

(401, '土木2001班', 30, '土木工程'),

(501, '会计2001班', 29, '会计学'),

(601, '英语2001班', 22, '英语');

-- 插入课程名表数据

INSERT INTO 课程名表 (课程编号, 课程名, 课程类别, 学分) VALUES

(1, '高等数学', '必修', 5),

(2, '大学英语', '必修', 4),

(3, '数据结构', '必修', 4),

(4, '操作系统', '必修', 4),

(5, '数据库原理', '选修', 3),

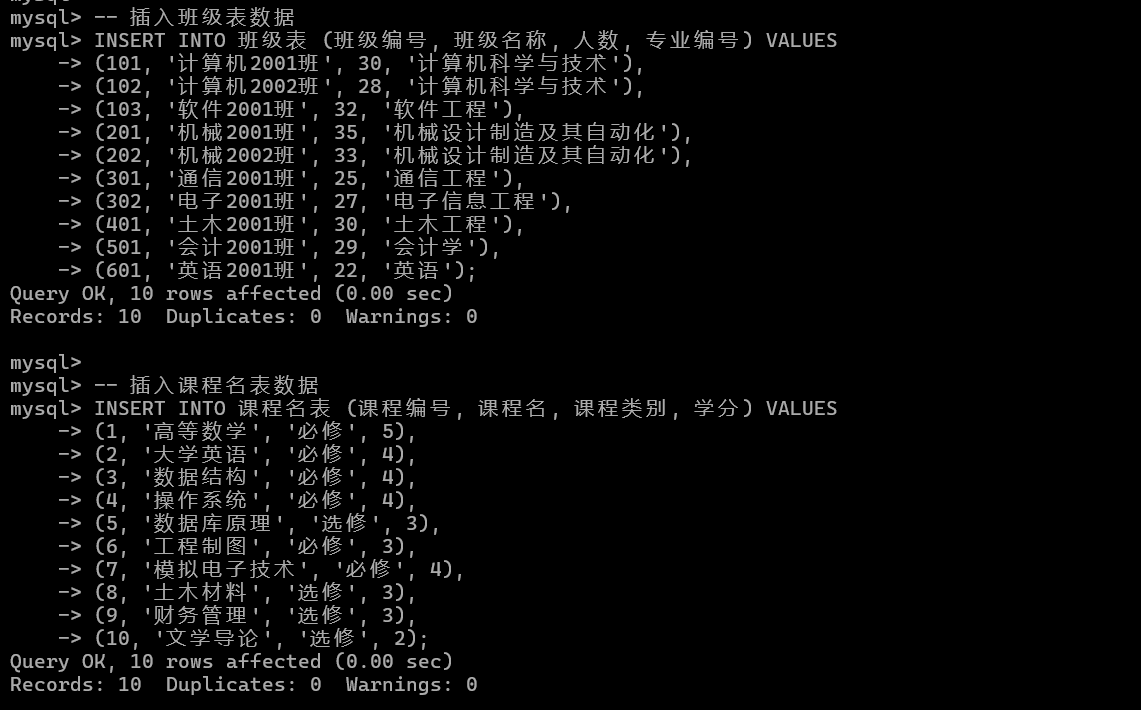
(6, '工程制图', '必修', 3),

(7, '模拟电子技术', '必修', 4),

(8, '土木材料', '选修', 3),

(9, '财务管理', '选修', 3),

(10, '文学导论', '选修', 2);



-- 插入学生档案表数据

INSERT INTO 学生档案表 (学号, 姓名, 性别, 出生日期, 政治面貌, 班级编号, 毕业学校) VALUES

(1001, '张三', '男', '2001-06-15', '团员', 101, '某中学'),

(1002, '李四', '女', '2002-03-22', '团员', 101, '某中学'),

(1003, '王五', '男', '2001-12-10', '团员', 102, '某中学'),

(1004, '赵六', '女', '2001-05-05', '团员', 102, '某中学'),

(1005, '钱七', '男', '2001-07-17', '团员', 103, '某中学'),

(1006, '孙八', '女', '2001-09-30', '群众', 103, '某中学'),

(1007, '周九', '男', '2001-11-02', '团员', 201, '某中学'),

(1008, '吴十', '女', '2002-01-12', '团员', 201, '某中学'),

(1009, '郑十一', '男', '2001-08-14', '群众', 202, '某中学'),

(1010, '王十二', '女', '2002-04-05', '团员', 301, '某中学');

-- 插入教师表数据

INSERT INTO 教师表 (教师编号, 课程编号, 班级编号, 授课地点, 职称) VALUES

(1, 1, 101, '教学楼101', '教授'),

(2, 2, 101, '教学楼102', '副教授'),

(3, 3, 102, '教学楼201', '讲师'),

(4, 4, 102, '教学楼202', '讲师'),

(5, 5, 103, '教学楼301', '副教授'),

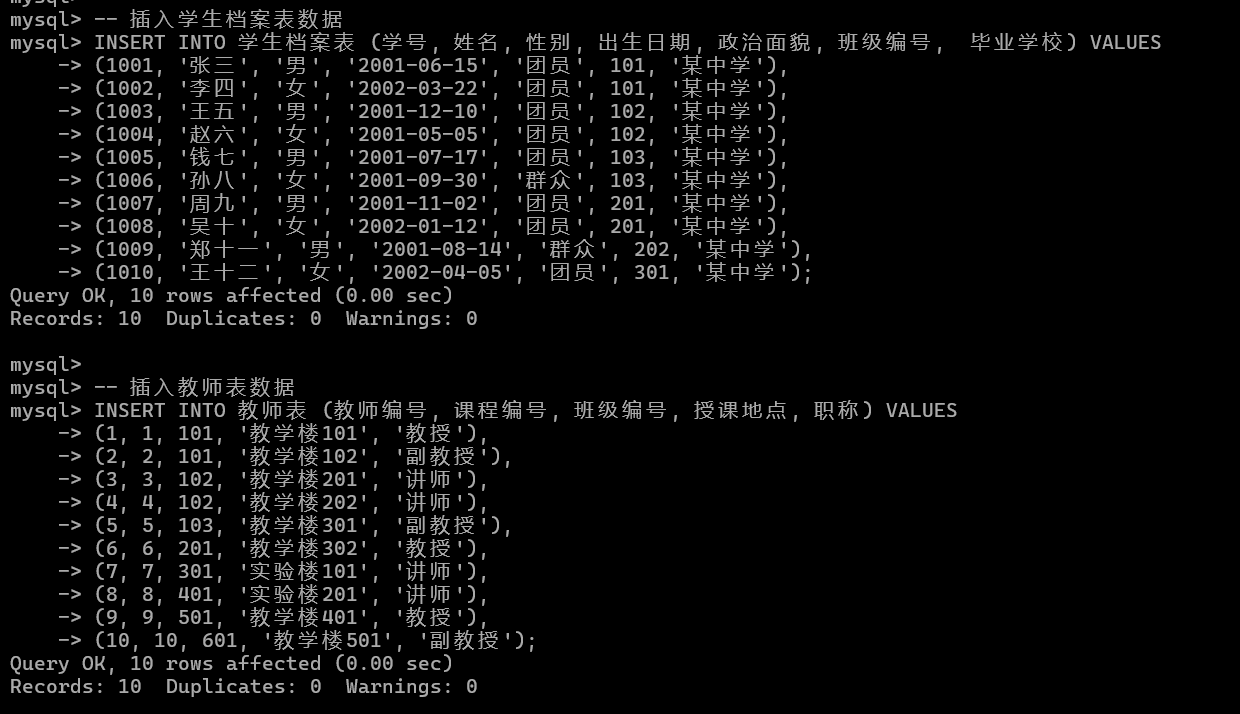
(6, 6, 201, '教学楼302', '教授'),

(7, 7, 301, '实验楼101', '讲师'),

(8, 8, 401, '实验楼201', '讲师'),

(9, 9, 501, '教学楼401', '教授'),

(10, 10, 601, '教学楼501', '副教授');



-- 插入学生成绩表数据

INSERT INTO 学生成绩表 (学号, 课程编号, 成绩, 考试次数) VALUES

(1001, 1, 85, 1),

(1002, 2, 90, 1),

(1003, 3, 78, 2),

(1004, 4, 88, 1),

(1005, 5, 92, 1),

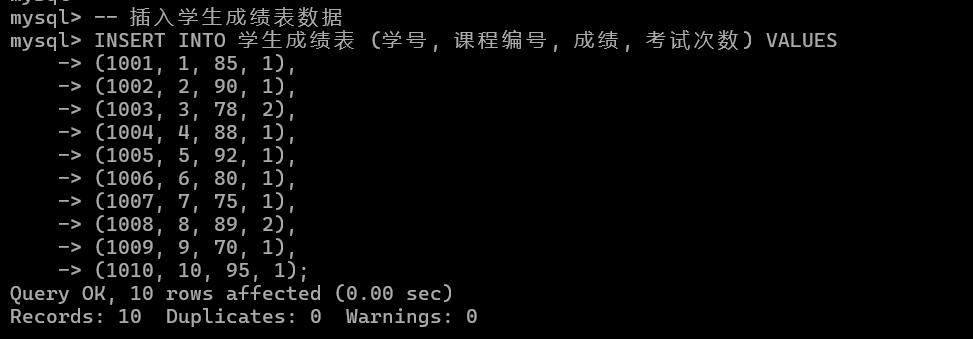
(1006, 6, 80, 1),

(1007, 7, 75, 1),

(1008, 8, 89, 2),

(1009, 9, 70, 1),

(1010, 10, 95, 1);



### 4.1.3 修改记录

表一

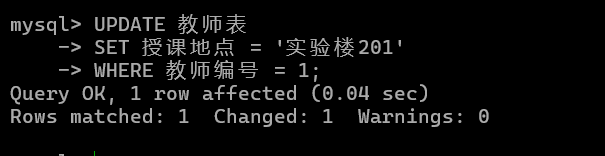
代码：

UPDATE 教师表

SET 授课地点 = '实验楼201'

WHERE 教师编号 = 1;

结果截图：



## 4.2 删除记录

删除一个表中的一条记录

代码：

DELETE FROM 学生成绩表

WHERE 学号 = 1003 AND 课程编号 = 3;

结果截图：



## 4.3查询问题及查询结果

### 4.3.1 选择列

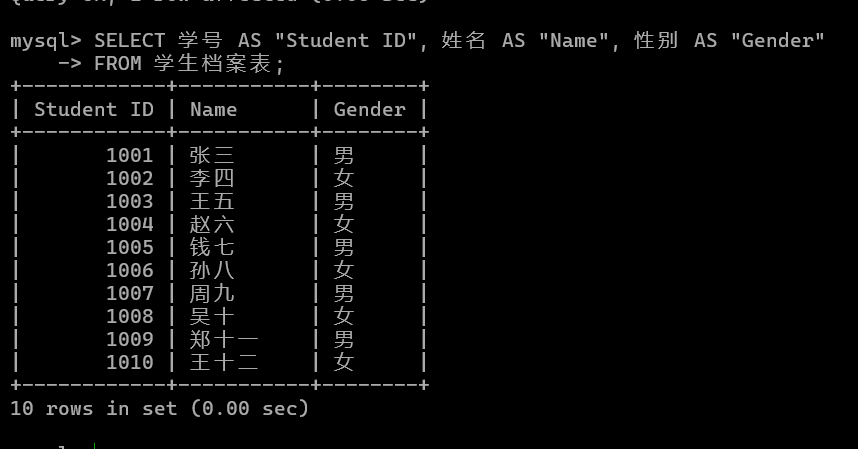
自行设计查询单表中的几列，并用别名代替。

代码：

SELECT 学号 AS "Student ID", 姓名 AS "Name", 性别 AS "Gender"

FROM 学生档案表;

结果截图：



### 4.3.2函数使用

计算满足条件的平均值，最大值，最小值。

代码：

-- 计算班级编号为101的学生成绩平均值

SELECT AVG(成绩) AS "Average Score"

FROM 学生成绩表

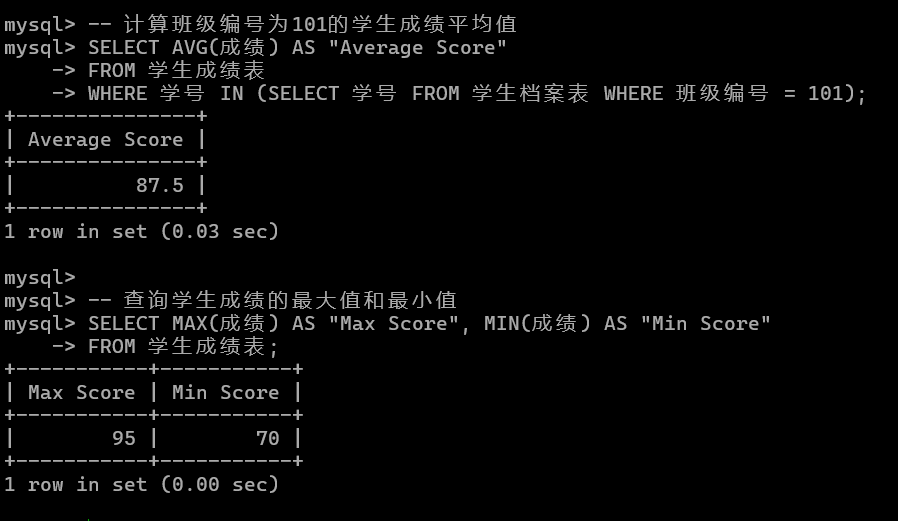
WHERE 学号 IN (SELECT 学号 FROM 学生档案表 WHERE 班级编号 = 101);

-- 查询学生成绩的最大值和最小值

SELECT MAX(成绩) AS "Max Score", MIN(成绩) AS "Min Score"

FROM 学生成绩表;

结果截图：



### 4.3.3 LIKE子句

用LIKE子句实现模糊查询

说明实现的功能：

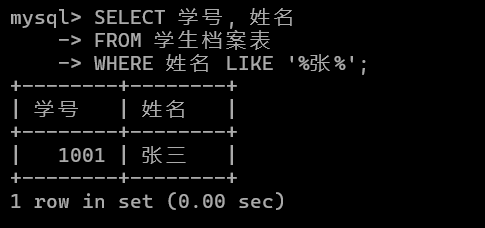
代码：

SELECT 学号, 姓名

FROM 学生档案表

WHERE 姓名 LIKE '%张%';

结果截图：



### 4.3.4 条件查询

用给定条件，查询结果

说明实现的功能：

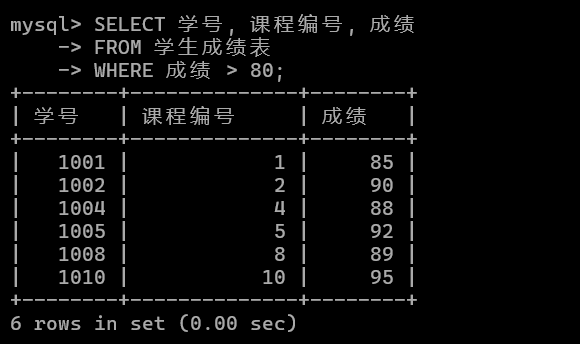
代码：

SELECT 学号, 课程编号, 成绩

FROM 学生成绩表

WHERE 成绩 > 80;

结果截图：



### 4.4.5多表查询

至少牵扯两个表，根据给定条件，查询结果

说明实现的功能：

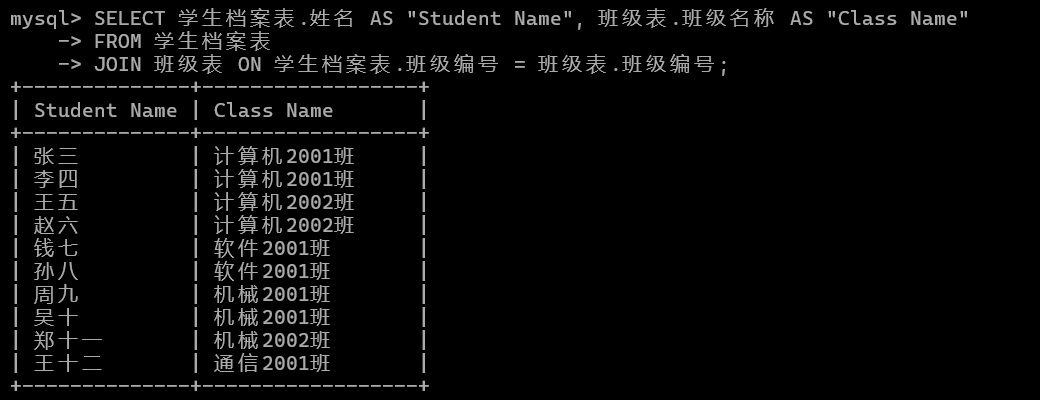
代码：

SELECT 学生档案表.姓名 AS "Student Name", 班级表.班级名称 AS "Class Name"

FROM 学生档案表

JOIN 班级表 ON 学生档案表.班级编号 = 班级表.班级编号;

结果截图：



### 4.4.6子查询

单行子查询举例

说明实现的功能：查询成绩最高的学生学号和成绩

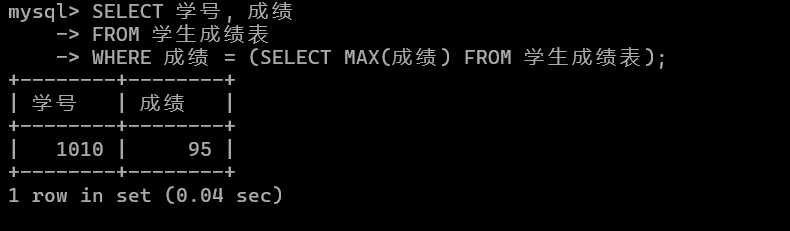
代码：

SELECT 学号, 成绩

FROM 学生成绩表

WHERE 成绩 = (SELECT MAX(成绩) FROM 学生成绩表);

结果截图：



多行子查询举例（用in，any,all任意一个均可）

说明实现的功能：查询学号在特定班级中的学生成绩

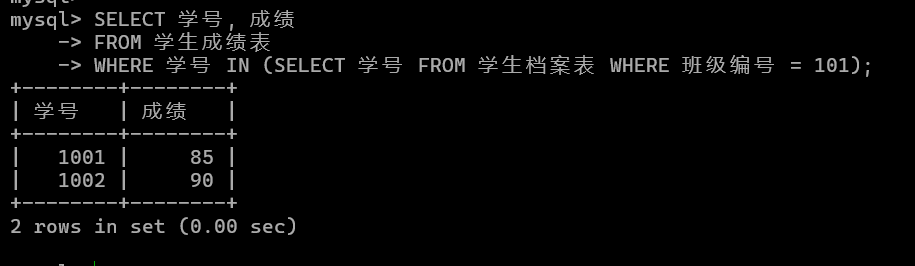
代码：

SELECT 学号, 成绩

FROM 学生成绩表

WHERE 学号 IN (SELECT 学号 FROM 学生档案表 WHERE 班级编号 = 101);

结果截图：



## 4.5建立视图

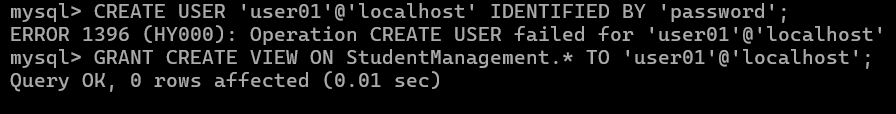
建立一个用户user01，给user01账户授予建立视图的权限。

代码：

CREATE USER 'user01'@'localhost' IDENTIFIED BY 'password';

GRANT CREATE VIEW ON StudentManagement.\* TO 'user01'@'localhost';

结果截图：



### 4.5.1 定义视图

自定义一个视图，要求实现多表查询。

代码：

-- 定义一个视图，查询学生姓名及其课程成绩

CREATE VIEW StudentScores AS

SELECT 学生档案表.姓名 AS "Student Name", 学生成绩表.成绩 AS "Score"

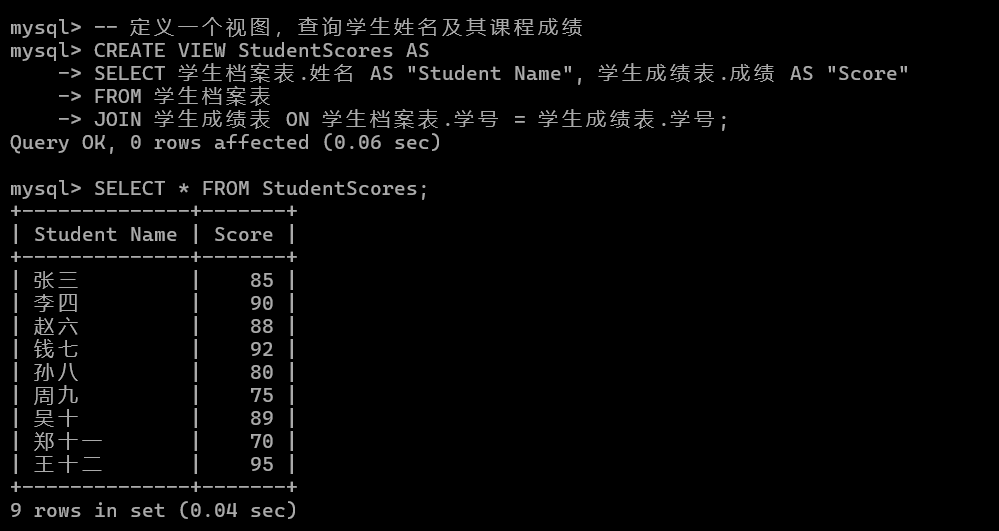
FROM 学生档案表

JOIN 学生成绩表 ON 学生档案表.学号 = 学生成绩表.学号;

查询自定义的视图代码：

SELECT \* FROM StudentScores;

结果截图：



### 4.5.2利用视图插入数据

创建简单视图代码：

CREATE VIEW SimpleView AS

SELECT 学号, 成绩

FROM 学生成绩表;

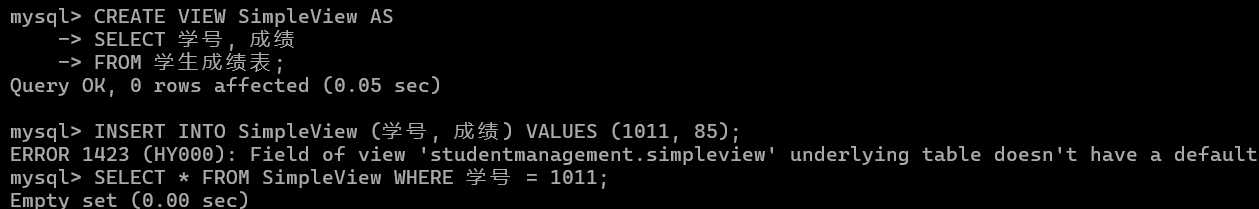
利用视图向表中插入一条记录代码：

INSERT INTO SimpleView (学号, 成绩) VALUES (1011, 85);

查询插入的记录代码：

SELECT \* FROM SimpleView WHERE 学号 = 1011;

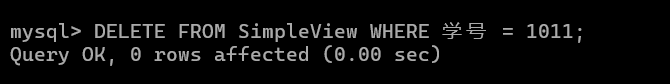
查询结果截图：



删除插入的结果代码：

SELECT \* FROM SimpleView WHERE 学号 = 1011;

删除结果截图：



# 5、建立存储过程及函数

## 5.1创建存储过程

创建一个存储过程，该存储过程实现输入某一个参数可以返回对应行上另外一个参数的值（如给定学号，返回选修课程信息等）。

代码：

-- 输入学号，返回学生选修的课程信息

DELIMITER //

CREATE PROCEDURE GetStudentCourses(IN stu\_id INT)

BEGIN

SELECT 课程名表.课程名

FROM 学生成绩表

JOIN 课程名表 ON 学生成绩表.课程编号 = 课程名表.课程编号

WHERE 学生成绩表.学号 = stu\_id;

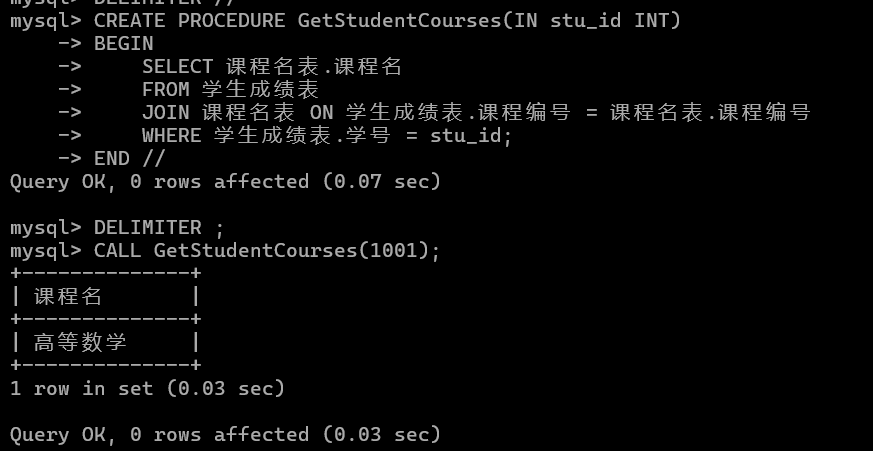
END //

DELIMITER ;

执行存储过程代码：

CALL GetStudentCourses(1001);

结果截图：



## 5.2存储过程应用

创建一个函数过程，实现统计功能（统计个数或平均值等）。

代码：

-- 统计指定班级的学生人数

DELIMITER //

CREATE FUNCTION CountStudentsInClass(class\_id INT)

RETURNS INT

BEGIN

DECLARE student\_count INT;

SELECT COUNT(\*) INTO student\_count

FROM 学生档案表

WHERE 班级编号 = class\_id;

RETURN student\_count;

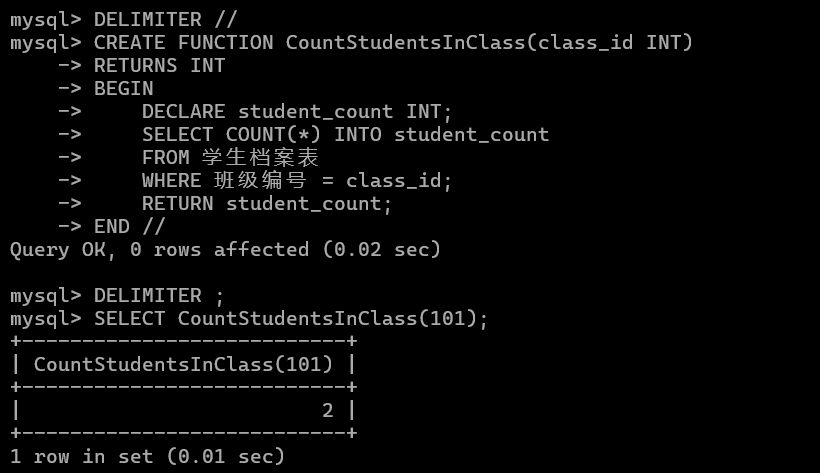
END //

DELIMITER ;

执行函数过程代码：

SELECT CountStudentsInClass(101);

结果截图：



# 6、建立触发器

在数据库的XXXX表中，定义一个触发器，当一个记录被删除时，把XXXX的主键添加到delXXXX（此表可单独创建）表中。

-- 创建空表代码：

CREATE TABLE del学生档案表 (

学号 INT,

删除时间 TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 创建触发器代码：

DELIMITER //

CREATE TRIGGER AfterDeleteStudent

AFTER DELETE ON 学生档案表

FOR EACH ROW

BEGIN

INSERT INTO del学生档案表 (学号) VALUES (OLD.学号);

END //

DELIMITER ;

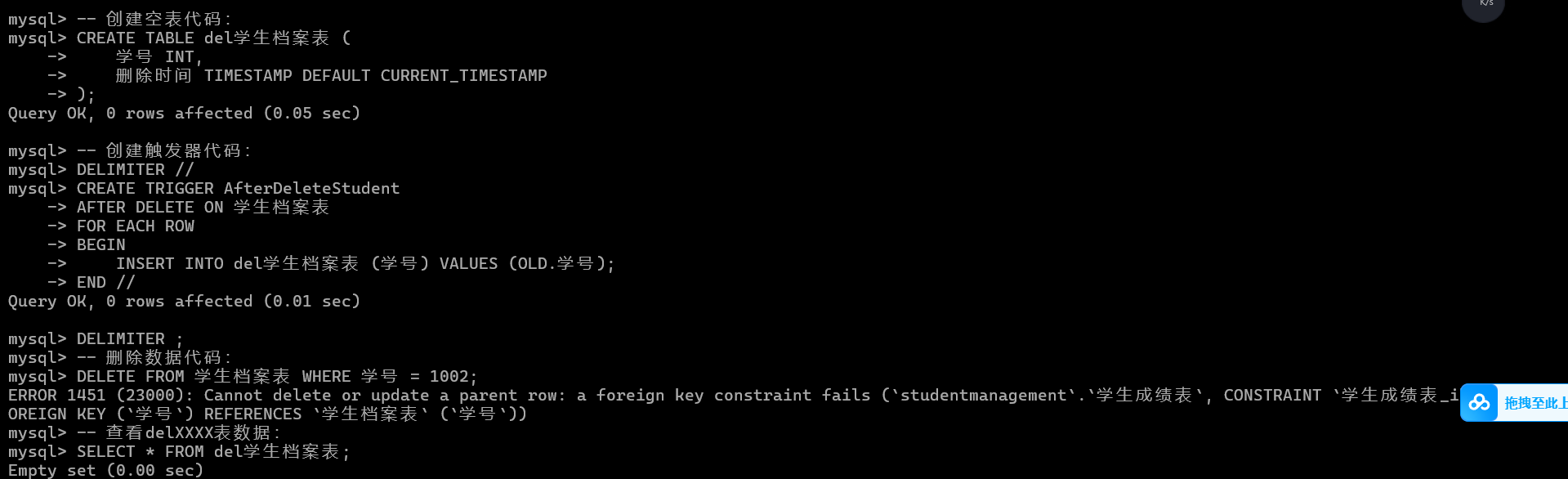
-- 删除数据代码：

DELETE FROM 学生档案表 WHERE 学号 = 1002;

-- 查看delXXXX表数据：

SELECT \* FROM del学生档案表;

结果截图：



# 7、项目设计总结

总结你的经验教训，所获得的收获（每个同学不能相同）

 **外键约束的复杂性**  
数据库表之间的外键约束设计在理论上看起来简单，但实际操作中需要注意表的创建顺序，以及数据插入时外键约束的匹配。例如，在建立“学生档案表”时，我因为“班级表”还未创建，导致外键错误。这让我学到了创建表和插入数据需要严格按照依赖顺序进行。

 **触发器和存储过程的细节问题**  
在创建触发器和存储过程中，我多次因为未考虑MySQL的安全约束（如DETERMINISTIC或log\_bin\_trust\_function\_creators变量设置问题）而失败。这让我认识到，深入理解数据库的配置和环境约束是成功运行复杂操作的基础。

 **LIKE模糊查询的优化**  
在实现模糊查询时，我发现直接用LIKE '%关键字%'在数据量大时性能较低。通过查阅资料，我了解到可以结合索引优化查询效率。这让我意识到性能优化是数据库设计和使用中不可忽视的一环。