



华中科技大学

数据库系统概论实验报告

姓 名:

学 院: 网络空间安全学院

专 业:

班 级:

学 号:

指导教师:

分数	
教师签名	

2024 年 4 月 27 日

目 录

1 课程任务概述	1
1.1 任务一 数据库定义与基本操作	1
1.2 任务二 SQL 的复杂操作	1
1.3 任务三 SQL 的高级实验	2
1.4 任务四 数据库设计	2
2 数据库定义与基本操作	3
2.1 任务要求	3
2.2 完成过程	3
2.3 任务总结	7
3 SQL 的复杂操作	8
3.1 任务要求	8
3.2 完成过程	8
3.3 任务总结	13
4 SQL 的高级实验	14
4.1 任务要求	14
4.2 完成过程	14
4.3 任务总结	23
5 数据库设计	24
5.1 任务要求	24
5.2 完成过程	24
5.3 任务总结	29
6 课程总结	31
附录	32

1 课程任务概述

1.1 任务一 数据库定义与基本操作

1.1.1 实验目的

- (1) 掌握 DBMS 的数据定义功能
- (2) 掌握 SQL 语言的数据定义语句
- (3) 掌握 DBMS 的数据单表查询功能
- (4) 掌握 SQL 语言的数据单表查询语句

1.1.2 实验要求

- (1) 熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select
- (2) 写出实验报告

1.1.3 实验步骤

- (1) 安装数据库管理系统 DBMS
- (2) 基于可视化界面或者命令行窗口创建数据库
- (3) 完成基本表操作
- (4) 在表中添加示例数据
- (5) 对学生关系 Student、课程关系 Course 和选修关系 SC 进行查询

1.2 任务二 SQL 的复杂操作

1.2.1 实验目的

掌握 SQL 语言的数据多表查询语句和更新操作

1.2.2 实验要求

- (1) 熟练掌握 SQL 的连接查询语句
- (2) 熟练掌握 SQL 的嵌套查询语句
- (3) 掌握表名前缀、别名前缀的用法
- (4) 掌握不相关子查询和相关子查询的区别和用法
- (5) 掌握不同查询之间的等价替换方法（一题多解）及限制
- (6) 熟练掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE
- (7) 记录实验结果，认真完成实验报告

1.2.3 实验步骤

- (1) 使用上次实验的数据库，如果没有保存，则重新建立，并输入数据。
- (2) 对学生关系 Student、课程关系 Course 和选修关系 SC 进行多表查询

1.3 任务三 SQL 的高级实验

1.3.1 实验目的

掌握 SQL 语言的视图、触发器、存储过程、安全等功能

1.3.2 实验要求

- (1) 掌握视图的定义与操作
- (2) 掌握对触发器的定义
- (3) 掌握对存储过程的定义
- (4) 掌握如何对用户进行授权和收回权限
- (5) 掌握用户定义完整性的方法
- (6) 写出实验报告

1.3.3 实验步骤

- (1) 创建表的视图
- (2) 利用视图完成表的查询
- (3) 删除表的视图
- (4) 创建触发器
- (5) 创建存储过程
- (6) 对用户进行授权和查询
- (7) 用户定义完整性

1.4 任务四 数据库设计

1.4.1 实验目的

掌握数据库设计和开发技巧，通过一个数据库具体设计实例，掌握数据库设计的方法。

1.4.2 实验要求

熟练掌握使用 SQL 语句设计数据库的方法，实现前述实验的学生管理系统，完成实验报告。

1.4.3 实验步骤

搭建 Python + MySQL 开发环境，完成数据库系统实现如下功能：

- 1) 新生入学信息增加，学生信息修改。
- 2) 课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。
- 3) 录入学生成绩，修改学生成绩。
- 4) 按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。
- 5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。
- 6) 输入学号，显示该学生的基本信息和选课信息。

2 数据库定义与基本操作

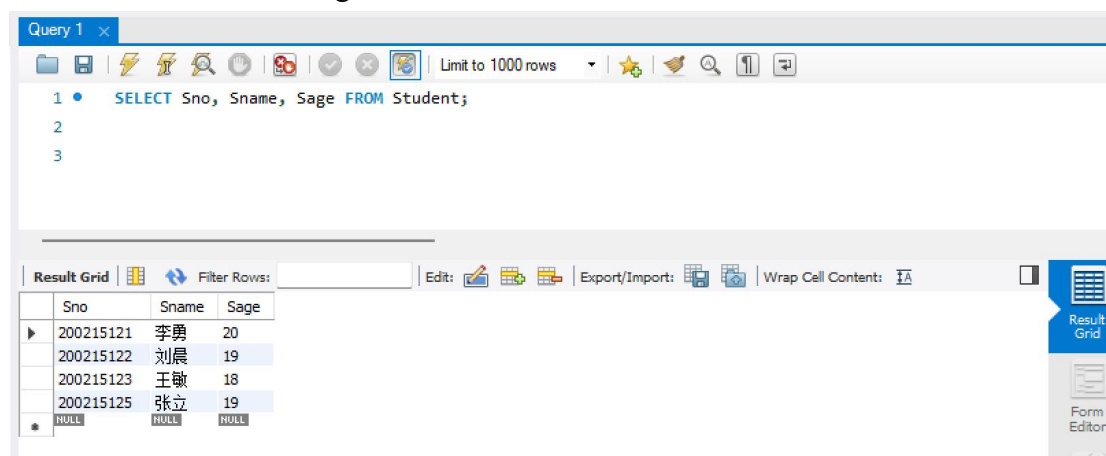
2.1 任务要求

- (1) 掌握 DBMS 的数据定义功能
- (2) 掌握 SQL 语言的数据定义语句
- (3) 掌握 DBMS 的数据单表查询功能
- (4) 掌握 SQL 语言的数据单表查询语句
- (5) 熟练掌握 SQL 的数据定义语句 CREATE、ALTER、DROP、Select
- (6) 写出实验报告

2.2 完成过程

2.2.1 查询全体学生的学号、姓名和年龄

SELECT Sno, Sname, Sage FROM Student;

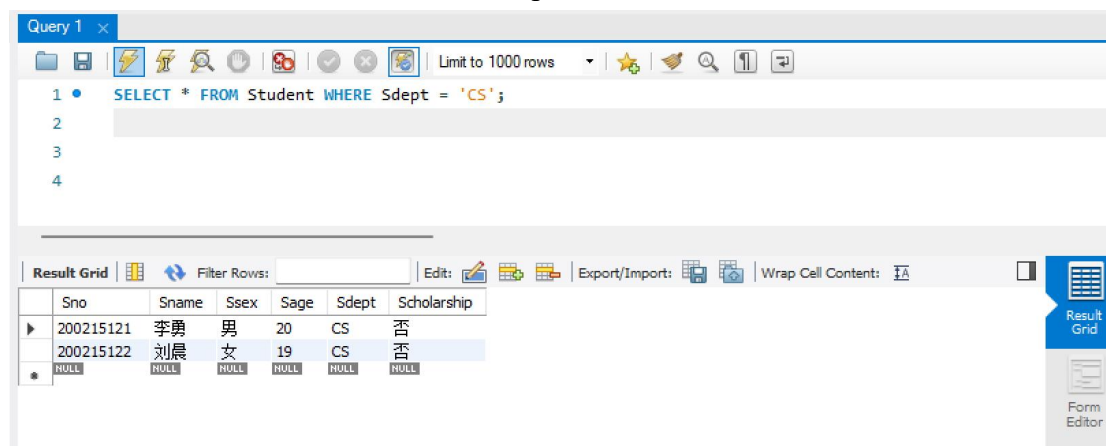


Sno	Sname	Sage
200215121	李勇	20
200215122	刘晨	19
200215123	王敏	18
200215125	张立	19

图 2.1 查询结果

2.2.2 查询所有计算机系学生的详细记录

SELECT * FROM Student WHERE Sdept = 'CS';

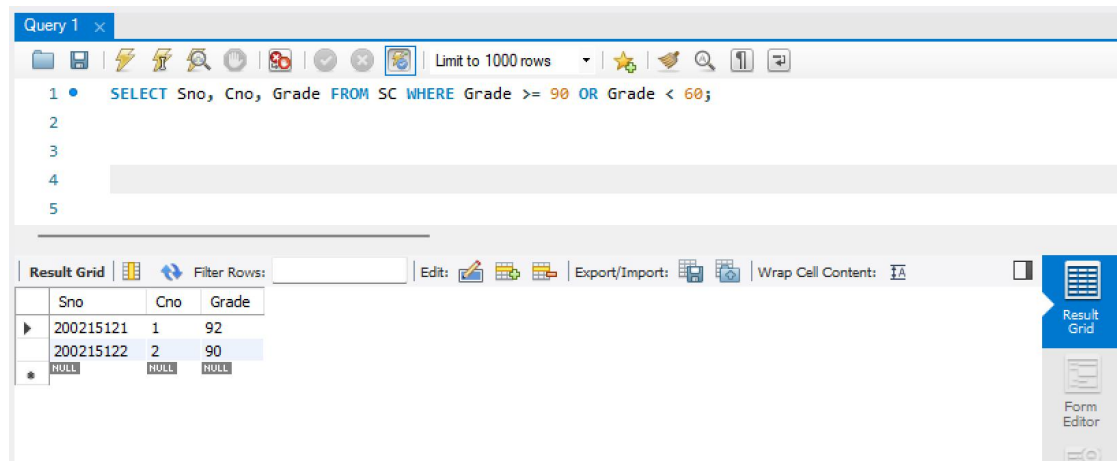


Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	20	CS	否
200215122	刘晨	女	19	CS	否

图 2.2 查询记录

2.2.3 找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩

```
SELECT Sno, Cno, Grade FROM SC WHERE Grade >= 90 OR Grade < 60;
```



Query 1

```
1 • SELECT Sno, Cno, Grade FROM SC WHERE Grade >= 90 OR Grade < 60;
```

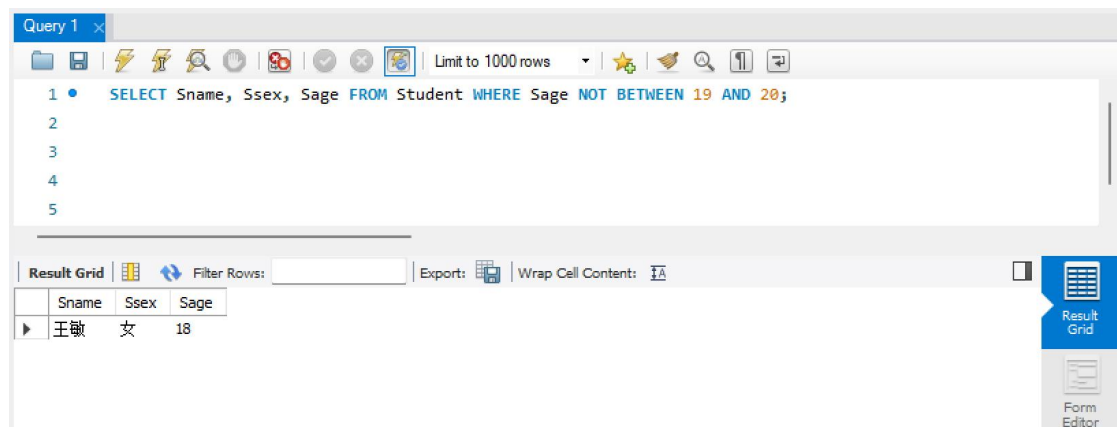
Result Grid

Sno	Cno	Grade
200215121	1	92
200215122	2	90
NULL	NULL	NULL

图 2.3 查询结果

2.2.4 查询年龄不在 19~20 岁之间的学生姓名、性别和年龄

```
SELECT Sname, Ssex, Sage FROM Student WHERE Sage NOT BETWEEN 19 AND 20;
```



Query 1

```
1 • SELECT Sname, Ssex, Sage FROM Student WHERE Sage NOT BETWEEN 19 AND 20;
```

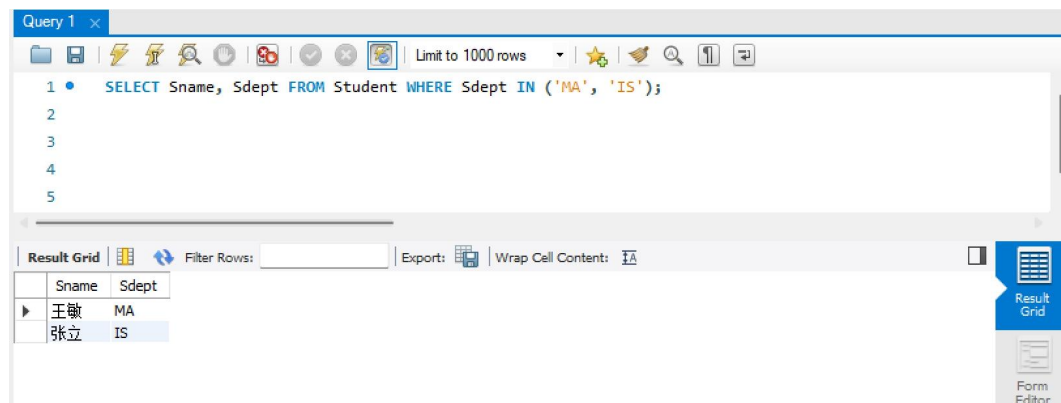
Result Grid

Sname	Ssex	Sage
王敏	女	18

图 2.4 查询结果

2.2.5 查询数学系（MA）、信息系（IS)的学生的姓名和所在系

```
SELECT Sname, Sdept FROM Student WHERE Sdept IN ('MA', 'IS');
```



Query 1

```
1 • SELECT Sname, Sdept FROM Student WHERE Sdept IN ('MA', 'IS');
```

Result Grid

Sname	Sdept
王敏	MA
张立	IS

图 2.5 查询结果

2.2.6 查询名称中包含“数据”的所有课程的课程号、课程名及其学分

SELECT Cno, Cname, Ccredit FROM Course WHERE Cname LIKE '%数据%';

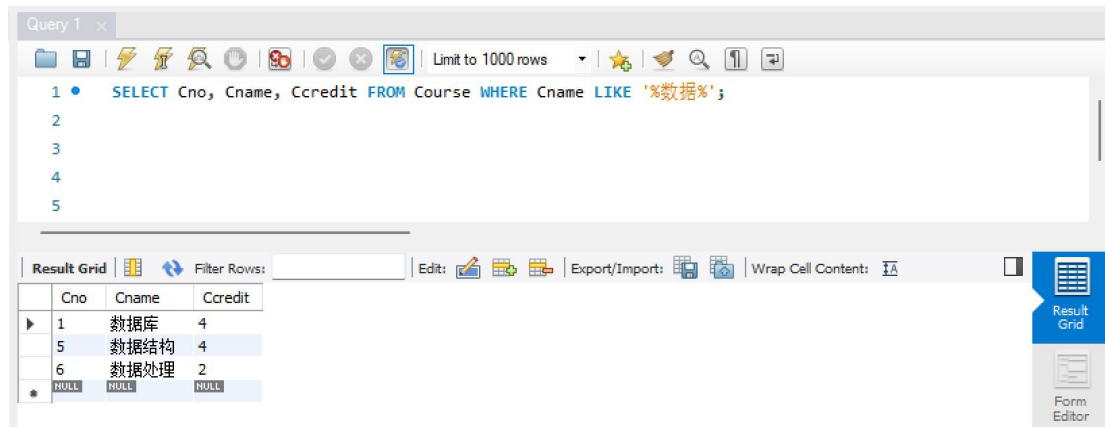


图 2.6 查询名称

2.2.7 找出所有没有选修课成绩的学生学号和课程号

SELECT Sno, Cno FROM SC WHERE Grade IS NULL;

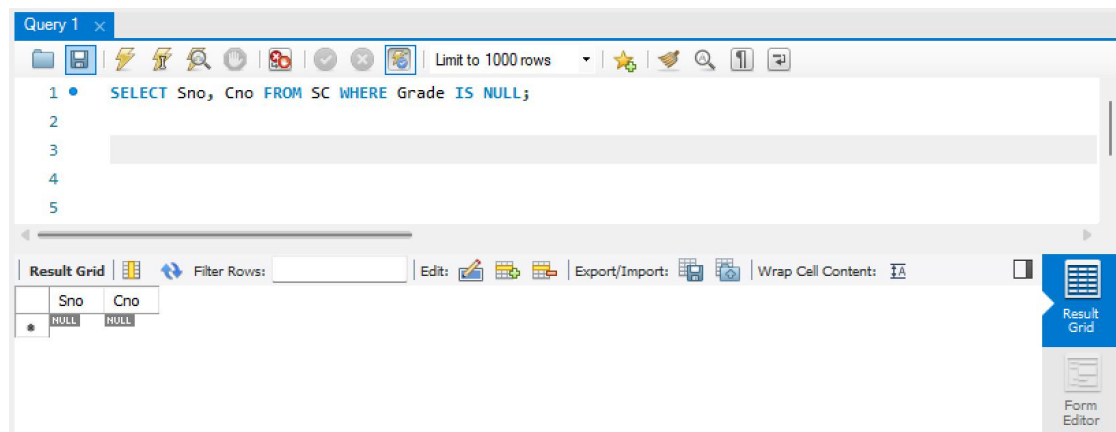


图 2.7 查询结果

2.2.8 查询学生 200215121 选修课的最高分、最低分以及平均成绩

SELECT MAX(Grade) AS HighestScore, MIN(Grade) AS LowestScore, AVG(Grade) AS AverageScore FROM SC WHERE Sno = '200215121';

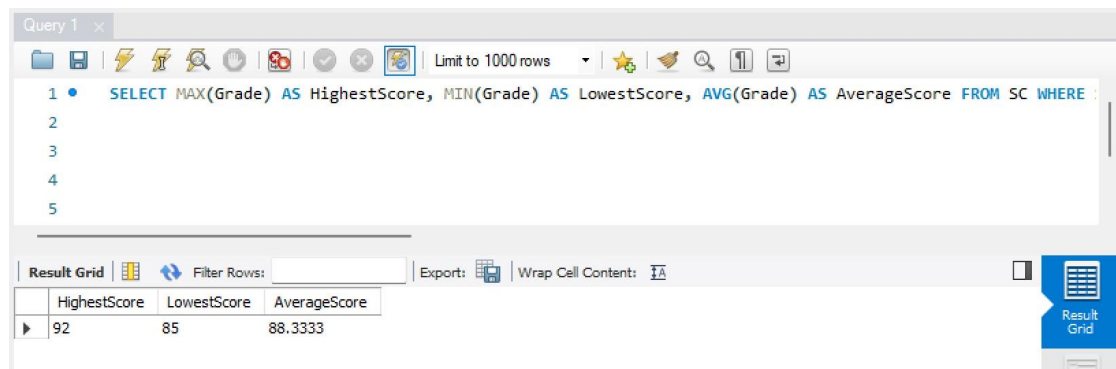


图 2.8 查询结果

2.2.9 查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列

SELECT Sno, Grade FROM SC WHERE Cno = '2' ORDER BY Grade ASC;

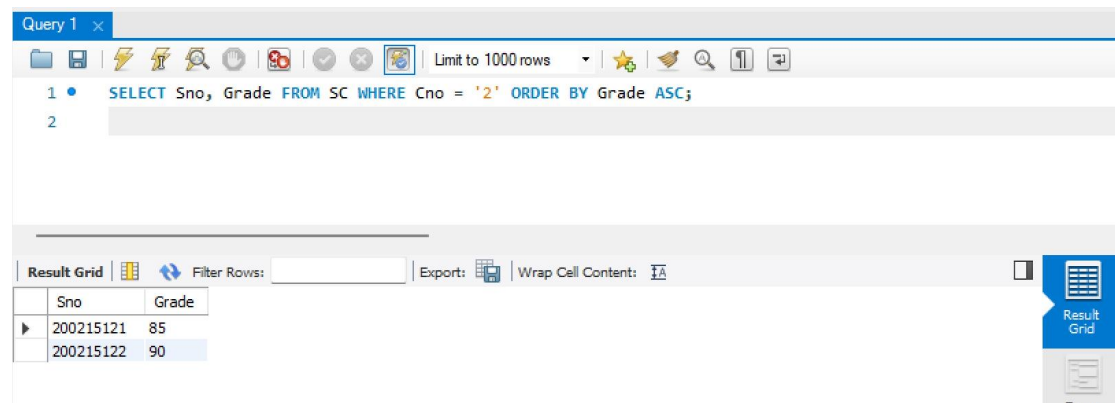


图 2.9 查询并升序排列

2.2.10 查询每个系名及其学生的平均年龄。

SELECT Sdept, AVG(Sage) AS AverageAge FROM Student GROUP BY Sdept;

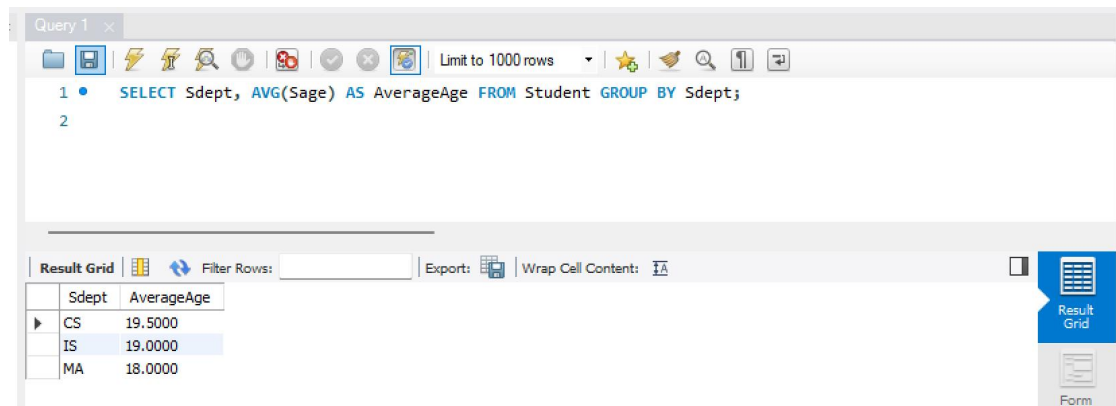


图 2.10 查询系名及平均年龄

2.2.11 思考：如何查询学生平均年龄在 19 岁以下（含 19 岁）的系别及其学生的平均年龄？

SELECT Sdept, AVG(Sage) AS AverageAge FROM Student GROUP BY Sdept
HAVING AVG(Sage) <= 19;

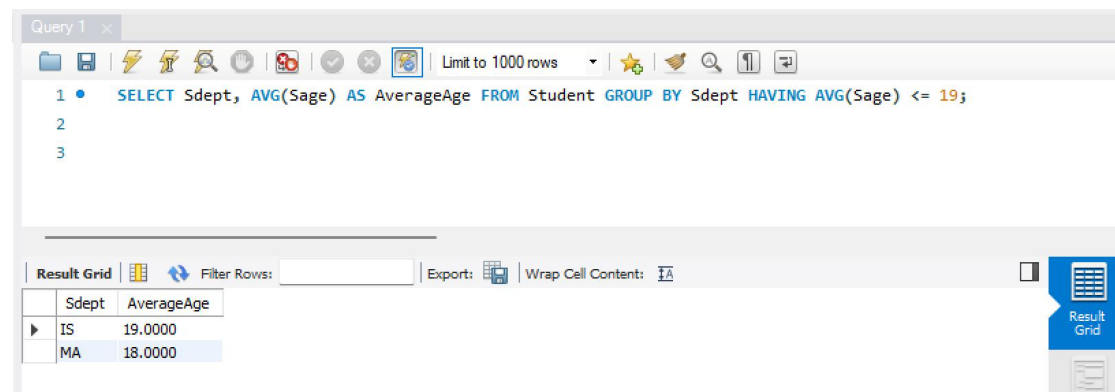


图 2.11 查询结果

2.3 任务总结

安装 MySQL 和配置可视化界面时遇到了一些问题，最后选择了 MySQL Workbench 这个简洁方便的可视化工具。在刚开始使用新工具时不熟悉，对很多功能感到迷茫，通过查找教程和学习博客逐渐了解了 MySQL Workbench 的使用，成功进行了表的创建和 SQL 基本语句的使用。

3 SQL 的复杂操作

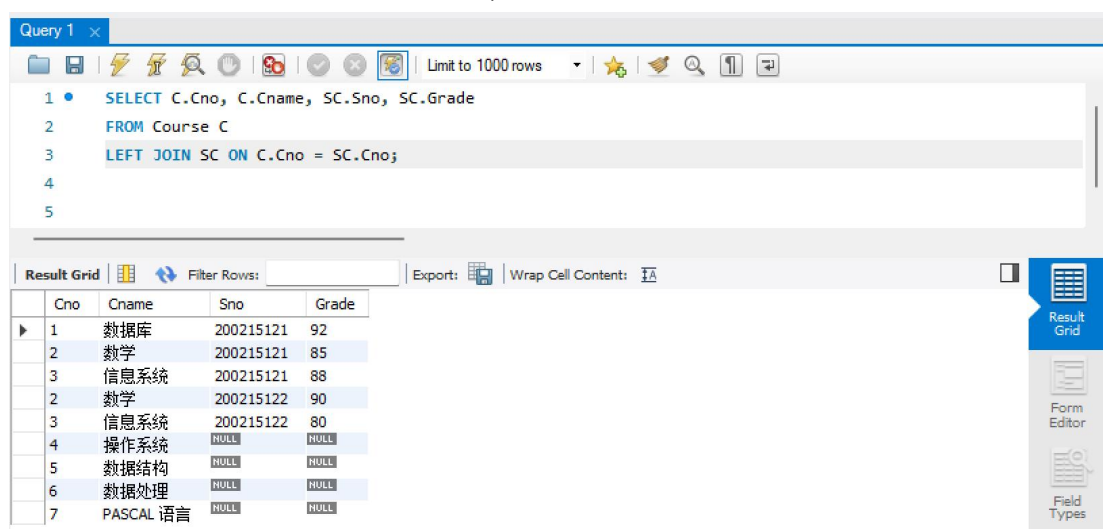
3.1 任务要求

- (1) 熟练掌握 SQL 的连接查询语句
- (2) 熟练掌握 SQL 的嵌套查询语句
- (3) 掌握表名前缀、别名前缀的用法
- (4) 掌握不相关子查询和相关子查询的区别和用法
- (5) 掌握不同查询之间的等价替换方法（一题多解）及限制
- (6) 熟练掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE
- (7) 记录实验结果，认真完成实验报告

3.2 完成过程

3.2.1 查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生学号及成绩--如果没有学生选择该课，则相应的学生学号及成绩为空值）。

```
SELECT C.Cno, C.Cname, SC.Sno, SC.Grade
FROM Course C
LEFT JOIN SC ON C.Cno = SC.Cno;
```



The screenshot shows a SQL query editor with the following query:

```
1 SELECT C.Cno, C.Cname, SC.Sno, SC.Grade
2 FROM Course C
3 LEFT JOIN SC ON C.Cno = SC.Cno;
4
5
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has four columns: Cno, Cname, Sno, and Grade. The results are as follows:

	Cno	Cname	Sno	Grade
1	数据库	200215121	92	
2	数学	200215121	85	
3	信息系统	200215121	88	
2	数学	200215122	90	
3	信息系统	200215122	80	
4	操作系统	NULL	NULL	
5	数据结构	NULL	NULL	
6	数据处理	NULL	NULL	
7	PASCAL 语言	NULL	NULL	

图 3.1 查询课程被选情况

3.2.2 查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

方法一：使用子查询

```
SELECT Sno, Sname, Sage
FROM Student
WHERE Sage = (SELECT Sage FROM Student WHERE Sname = '张立');
```

方法二：使用内连接

```
SELECT S1.Sno, S1.Sname, S1.Sage
FROM Student S1, Student S2
WHERE S1.Sage = S2.Sage AND S2.Sname = '张立' AND S1.Sname <> '张立';
```

方法三：使用 JOIN

```
SELECT S1.Sno, S1.Sname, S1.Sage
FROM Student S1
JOIN Student S2 ON S1.Sage = S2.Sage AND S2.Sname = '张立'
WHERE S1.Sname <> '张立';
```

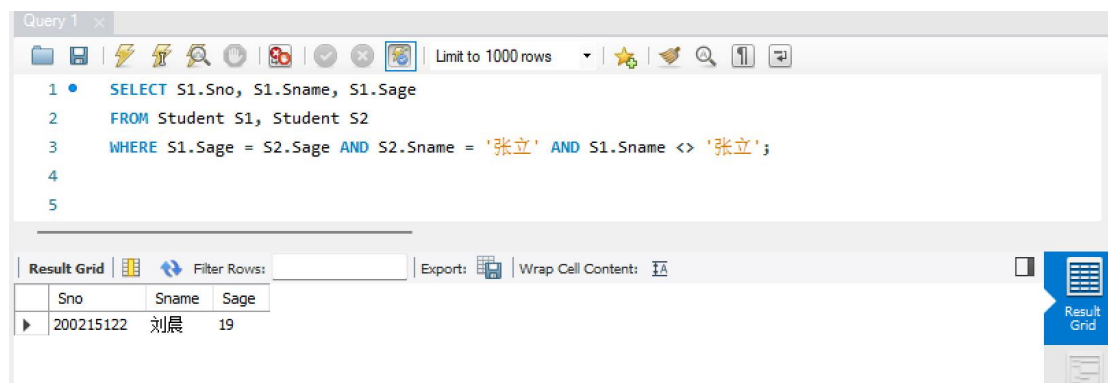


图 3.2 查询结果

3.2.3 查询选修了 3 号课程而且成绩为良好（80~89 分）的所有学生的学号和姓名。

```
SELECT Student.Sno, Student.Sname
FROM SC
JOIN Student ON SC.Sno = Student.Sno
WHERE SC.Cno = '3' AND SC.Grade BETWEEN 80 AND 89;
```

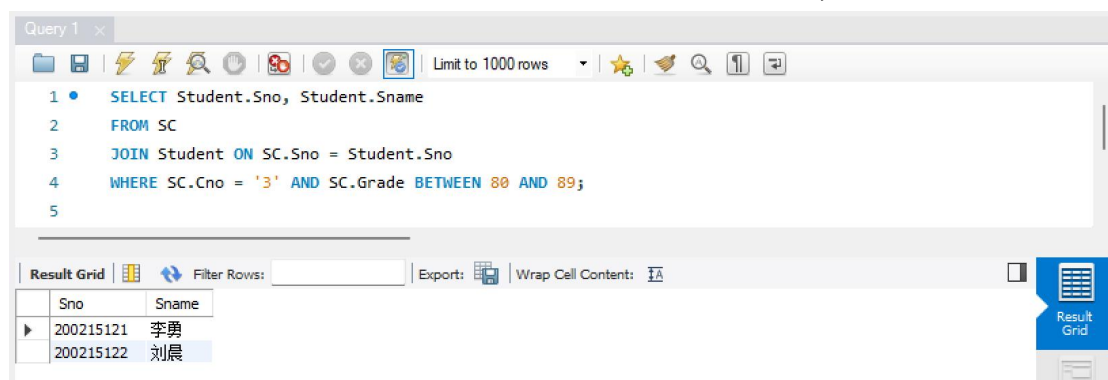


图 3.3 查询结果

3.2.4 查询学生 200215122 选修的课程号、课程名

```
SELECT SC.Cno, Course.Cname
FROM SC
JOIN Course ON SC.Cno = Course.Cno
WHERE SC.Sno = '200215122';
```

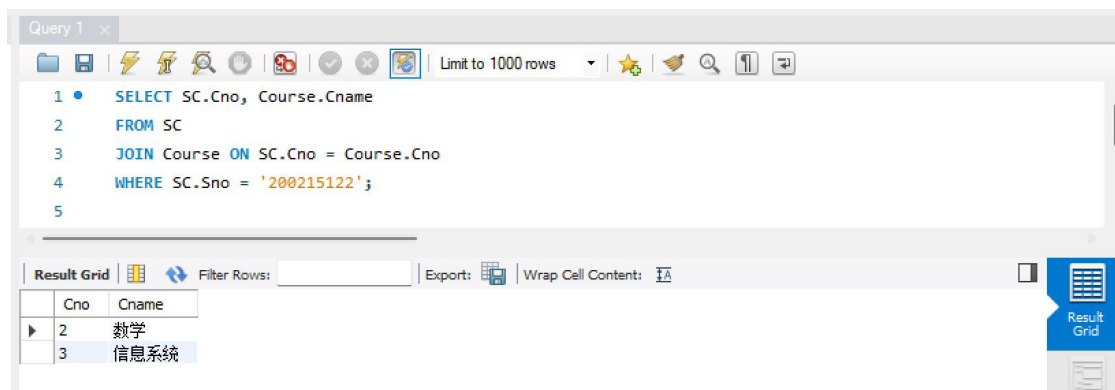


图 3.4 查询结果

（思考：如何查询学生 200215122 选修的课程号、课程名及成绩？）

```

SELECT SC.Cno, Course.Cname, SC.Grade
FROM SC
JOIN Course ON SC.Cno = Course.Cno
WHERE SC.Sno = '200215122';

```

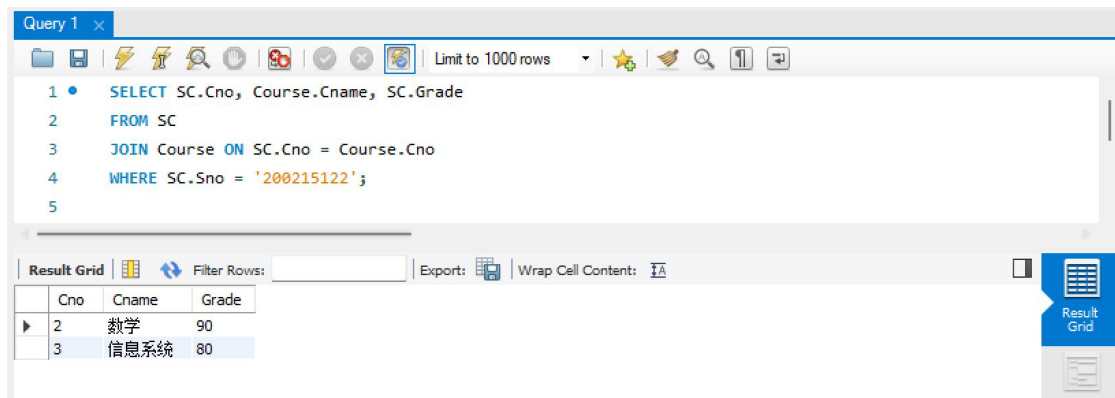


图 3.5 查询结果

3.2.5 找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。（输出学号和课程号）

```

SELECT SC.Sno, SC.Cno
FROM SC
WHERE SC.Grade < (
    SELECT AVG(Grade) - 5 FROM SC AS S2 WHERE SC.Sno = S2.Sno
GROUP BY Sno
);

```

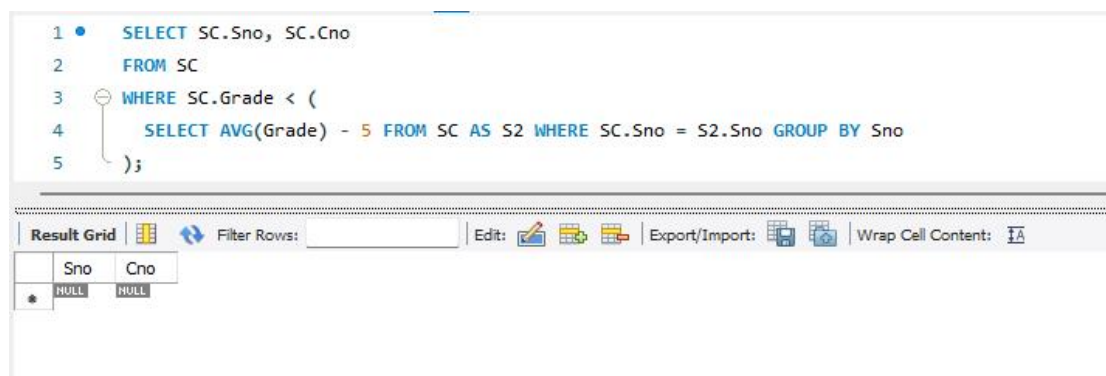


图 3.6 查询结果

3.2.6 查询比所有男生年龄都小的女生的学号、姓名和年龄。

```

SELECT Sno, Sname, Sage
FROM Student
WHERE Ssex = '女' AND Sage < ALL (SELECT Sage FROM Student WHERE
Ssex = '男');

```

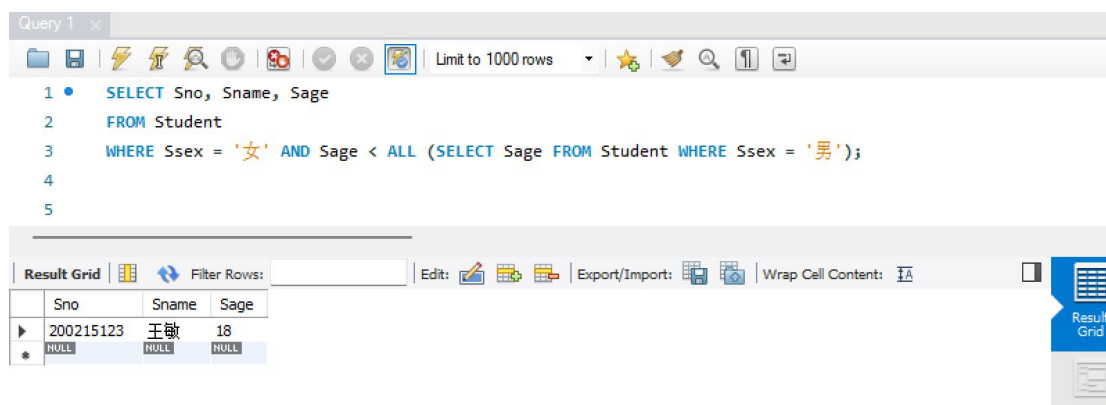


图 3.7 查询结果

3.2.7 查询所有选修了 2 号课程的学生姓名及所在系。

```

SELECT Student.Sname, Student.Sdept
FROM SC
JOIN Student ON SC.Sno = Student.Sno
WHERE SC.Cno = '2';

```

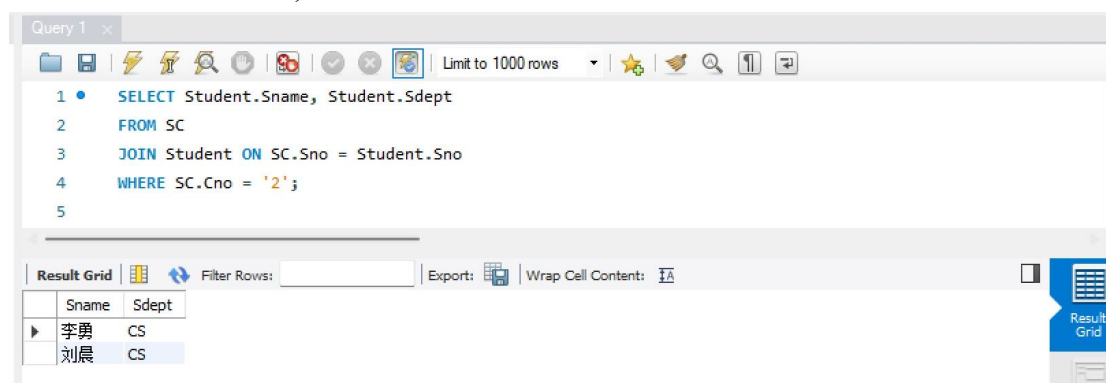


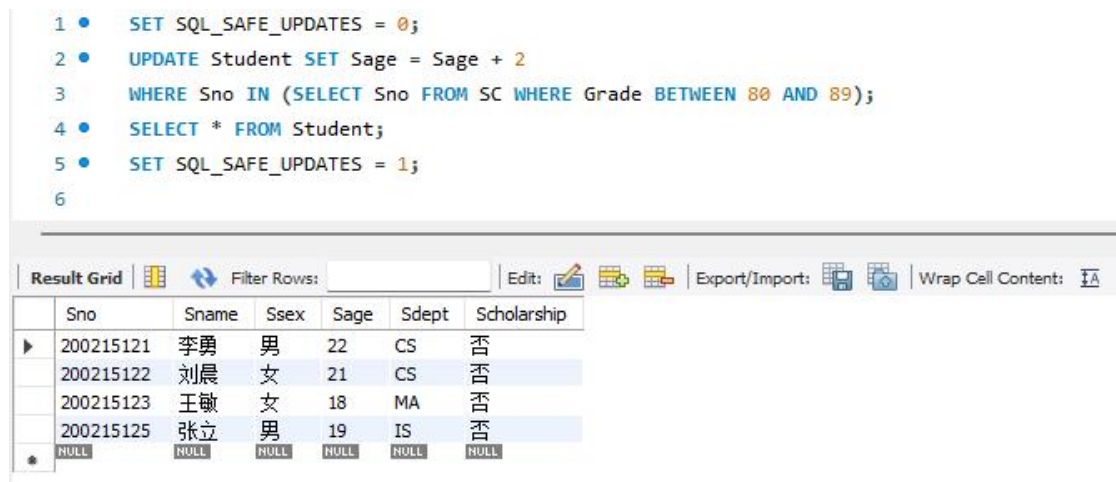
图 3.8 查询结果

3.2.8 使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

```
SET SQL_SAFE_UPDATES = 0;
UPDATE Student SET Sage = Sage + 2
WHERE Sno IN (SELECT Sno FROM SC WHERE Grade BETWEEN 80 AND
89);
```

```
SELECT * FROM Student;
```

```
SET SQL_SAFE_UPDATES = 1;
```



The screenshot shows a SQL query editor with the following code:

```
1 • SET SQL_SAFE_UPDATES = 0;
2 • UPDATE Student SET Sage = Sage + 2
3   WHERE Sno IN (SELECT Sno FROM SC WHERE Grade BETWEEN 80 AND 89);
4 • SELECT * FROM Student;
5 • SET SQL_SAFE_UPDATES = 1;
```

Below the code is a 'Result Grid' showing the results of the SELECT statement. The grid has columns: Sno, Sname, Ssex, Sage, Sdept, and Scholarship. The data is as follows:

Sno	Sname	Ssex	Sage	Sdept	Scholarship
200215121	李勇	男	22	CS	否
200215122	刘晨	女	21	CS	否
200215123	王敏	女	18	MA	否
200215125	张立	男	19	IS	否
NULL	NULL	NULL	NULL	NULL	NULL

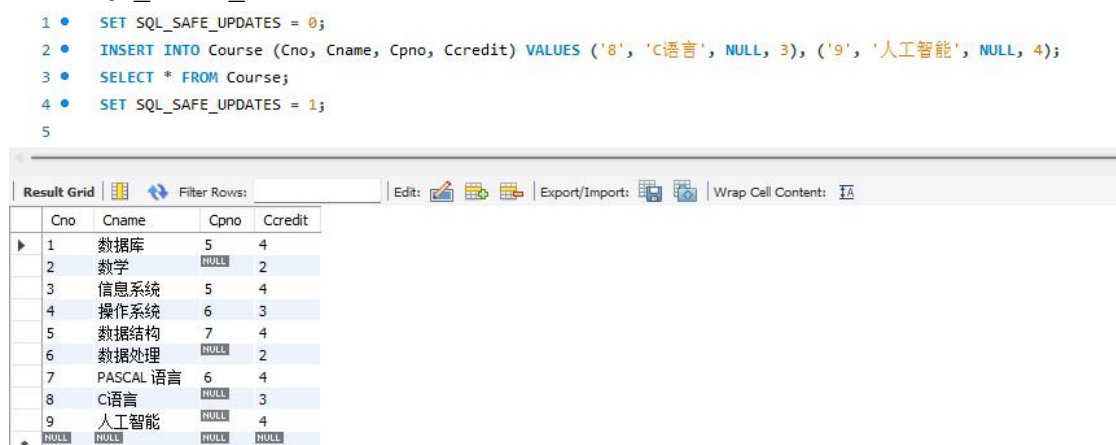
图 3.9 update 操作后查询结果

3.2.9 使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

```
SET SQL_SAFE_UPDATES = 0;
INSERT INTO Course (Cno, Cname, Cpno, Ccredit) VALUES ('8', 'C 语言',
NULL, 3), ('9', '人工智能', NULL, 4);
```

```
SELECT * FROM Course;
```

```
SET SQL_SAFE_UPDATES = 1;
```



The screenshot shows a SQL query editor with the following code:

```
1 • SET SQL_SAFE_UPDATES = 0;
2 • INSERT INTO Course (Cno, Cname, Cpno, Ccredit) VALUES ('8', 'C语言', NULL, 3), ('9', '人工智能', NULL, 4);
3 • SELECT * FROM Course;
4 • SET SQL_SAFE_UPDATES = 1;
```

Below the code is a 'Result Grid' showing the results of the SELECT statement. The grid has columns: Cno, Cname, Cpno, and Ccredit. The data is as follows:

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学	NULL	2
3	信息系统	5	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理	NULL	2
7	PASCAL 语言	6	4
8	C语言	NULL	3
9	人工智能	NULL	4
NULL	NULL	NULL	NULL

图 3.10 insert 操作后查询结果

3.2.10 使用 delete 语句把人工智能课程删除，并查询出来。

```
SET SQL_SAFE_UPDATES = 0;
```

```
DELETE FROM Course WHERE Cname = '人工智能';
```

```
SELECT * FROM Course;
SET SQL_SAFE_UPDATES = 1;
```

```
1 • SET SQL_SAFE_UPDATES = 0;
2 • DELETE FROM Course WHERE Cname = '人工智能';
3 • SELECT * FROM Course;
4 • SET SQL_SAFE_UPDATES = 1;
5
```

	Cno	Cname	Cpno	Ccredit
▶	1	数据库	5	4
	2	数学	NULL	2
	3	信息系统	5	4
	4	操作系统	6	3
	5	数据结构	7	4
	6	数据处理	NULL	2
	7	PASCAL 语言	6	4
	8	C语言	NULL	3
*	NULL	NULL	NULL	NULL

图 3.11 delete 操作后查询结果

3.3 任务总结

直接使用 insert 语句和 delete 语句时出现报错，需要通过 SET SQL_SAFE_UPDATES = 0;命令关闭严格的安全更新模式，在完成语句后再通过 SET SQL_SAFE_UPDATES = 1; 启用严格的安全更新模式。

4 SQL 的高级实验

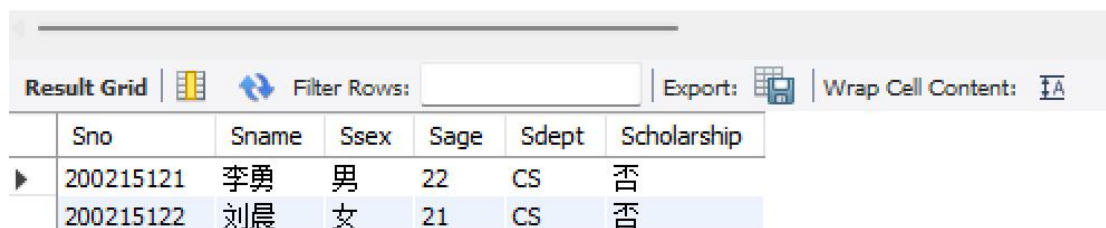
4.1 任务要求

- (1) 掌握视图的定义与操作
- (2) 掌握对触发器的定义
- (3) 掌握对存储过程的定义
- (4) 掌握如何对用户进行授权和收回权限
- (5) 掌握用户定义完整性的方法
- (6) 写出实验报告

4.2 完成过程

4.2.1 创建 CS 系的视图 CS_View

```
CREATE VIEW CS_View AS  
SELECT *  
FROM Student  
WHERE Sdept = 'CS';
```

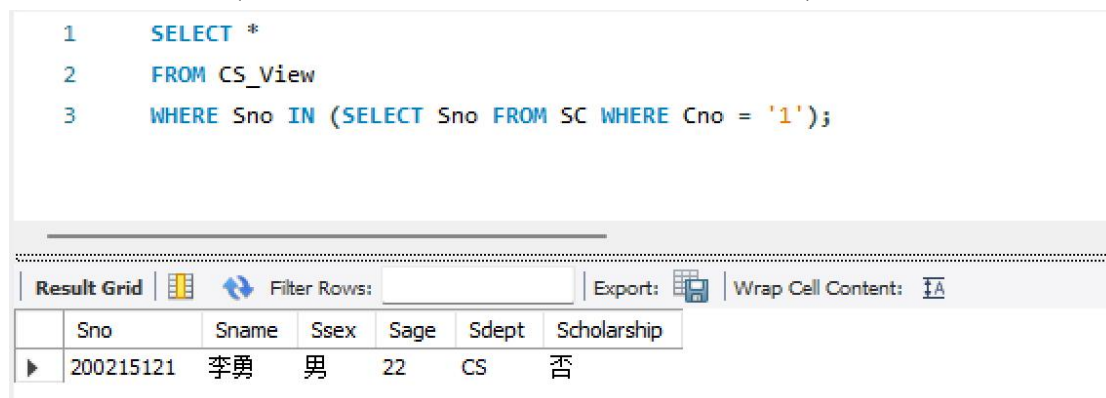


	Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶	200215121	李勇	男	22	CS	否
	200215122	刘晨	女	21	CS	否

图 4.1 创建 CS 系视图

4.2.2 在视图 CS_View 上查询 CS 系选修了 1 号课程的学生

```
SELECT *  
FROM CS_View  
WHERE Sno IN (SELECT Sno FROM SC WHERE Cno = '1');
```



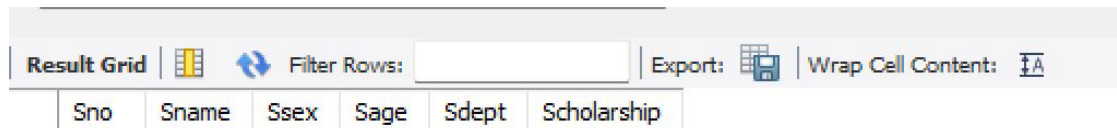
```
1  SELECT *  
2  FROM CS_View  
3  WHERE Sno IN (SELECT Sno FROM SC WHERE Cno = '1');
```

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶	200215121	李勇	男	22	CS	否

图 4.2 查询视图

4.2.3 创建 IS 系成绩大于 80 的学生的视图 IS_View

```
CREATE VIEW IS_View AS
SELECT Student.*
FROM Student
JOIN SC ON Student.Sno = SC.Sno
WHERE Student.Sdept = 'IS' AND SC.Grade > 80;
```

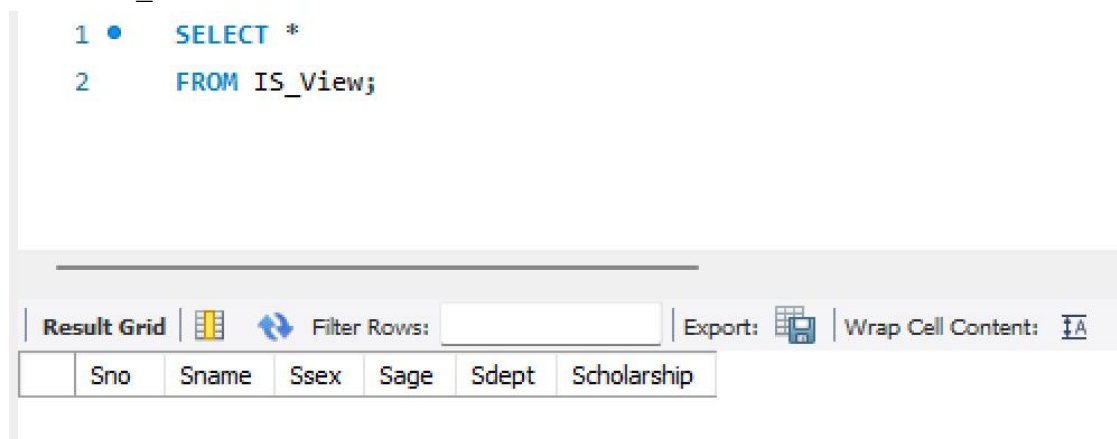


Sno	Sname	Ssex	Sage	Sdept	Scholarship
-----	-------	------	------	-------	-------------

图 4.3 创建视图

4.2.4 在视图 IS_View 查询 IS 系成绩大于 80 的学生

```
SELECT *
FROM IS_View;
```



Sno	Sname	Ssex	Sage	Sdept	Scholarship
-----	-------	------	------	-------	-------------

图 4.4 查询视图

4.2.5 删除视图 IS_View

```
DROP VIEW IS_View;
```



图 4.5 删除视图

4.2.6 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授予 Student 表的查询和更新的权限,给 U2 对 SC 表授予插入的权限。然后用 U1 登录,分别 1) 查询学生表的信息; 2) 把所有学生的年龄增加 1 岁,然后查询; 3) 删除 IS 系的学生; 4) 查询 CS 系的选课信息。用 U2 登录,分别 1) 在 SC 表中插入 1 条记录 (‘200215122’, ‘1’, 75); 2) 查询 SC 表的信息, 3) 查询视图 CS_View 的信息。

创建 2 个不同的用户

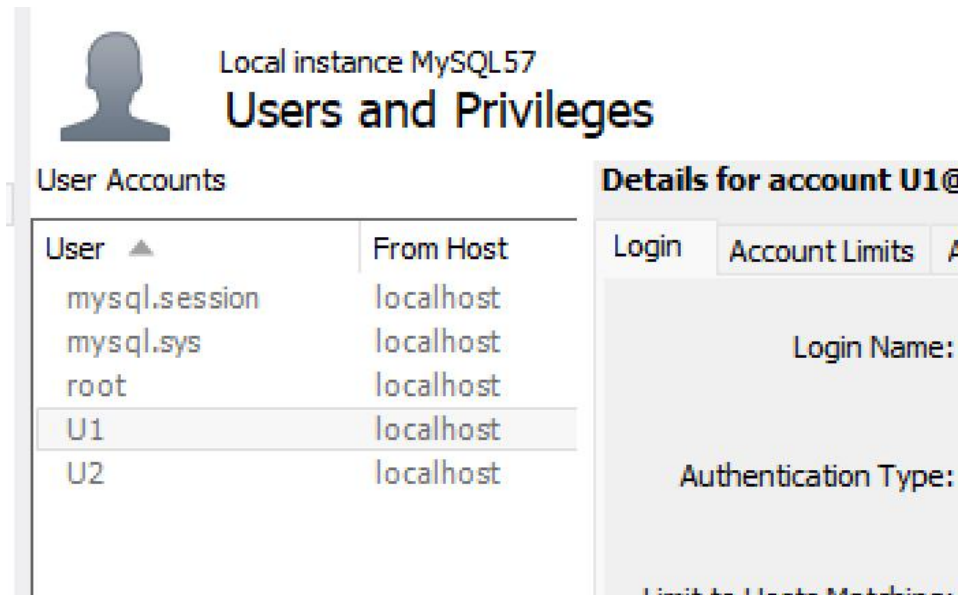


图 4.6 创建用户

1) 查询学生表的信息

```
SELECT * FROM student;
```

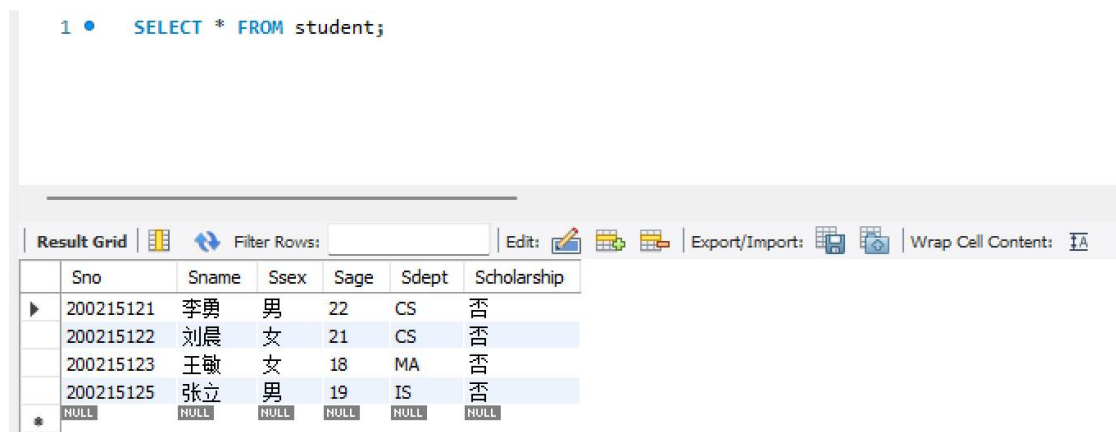


图 4.7 查询信息

2) 把所有学生的年龄增加 1 岁, 然后查询

```
SET SQL_SAFE_UPDATES = 0;
```

```
UPDATE Student SET Sage = Sage + 1;
```

```
SET SQL_SAFE_UPDATES = 1;
```

Result Grid						
	Sno	Sname	Ssex	Sage	Sdept	Scholarship
	200215121	李勇	男	23	CS	否
	200215122	刘晨	女	22	CS	否
	200215123	王敏	女	19	MA	否
	200215125	张立	男	20	IS	否
	NULL	NULL	NULL	NULL	NULL	NULL

图 4.8 增加年龄后查询

3) 删除 IS 系的学生

```
SET SQL_SAFE_UPDATES = 0;
```

```
DELETE FROM Student WHERE Sdept = 'IS';
```

```
SET SQL_SAFE_UPDATES = 1;
```

Query 1 x student

```

1 • SET SQL_SAFE_UPDATES = 0;
2 • DELETE FROM Student WHERE Sdept = 'IS';
3 • SET SQL_SAFE_UPDATES = 1;

```

Output

Action Output

#	Time	Action	Message
15	14:46:30	DELETE FROM Student WHERE Sdept = 'IS'	Error Code: 1142. DELETE command denied to user 'U1'@'localhost' for table 'stu..

图 4.9 删除后查询

4) 查询 CS 系的选课信息

```
SELECT * FROM SC JOIN Student ON SC.Sno = Student.Sno WHERE Sdept = 'CS';
```

Result Grid									
	Sno	Cno	Grade	Sno	Sname	Ssex	Sage	Sdept	Scholarship
	200215121	1	92	200215121	李勇	男	23	CS	否
	200215121	2	85	200215121	李勇	男	23	CS	否
	200215121	3	88	200215121	李勇	男	23	CS	否
	200215122	2	90	200215122	刘晨	女	22	CS	否
	200215122	3	80	200215122	刘晨	女	22	CS	否

图 4.10 查询选课信息

U2 登录， 1) 在 SC 表中插入 1 条记录（‘200215122’，‘1’，75）；

```
SET SQL_SAFE_UPDATES = 0;
```

```
INSERT INTO SC (Sno, Cno, Grade) VALUES ('200215122', '1', 75);
```

```
SET SQL_SAFE_UPDATES = 1;
```

	Sno	Cno	Grade
▶	200215121	1	92
	200215121	2	85
	200215121	3	88
	200215122	1	75
	200215122	2	90
	200215122	3	80
*	NULL	NULL	NULL

图 4.11 插入记录

2) 查询 SC 表的信息

SELECT * FROM SC;

1 SELECT * FROM SC;

Result Grid

Filter Rows:

Edit:

	Sno	Cno	Grade
▶	200215121	1	92
	200215121	2	85
	200215121	3	88
	200215122	1	75
	200215122	2	90
	200215122	3	80
*	NULL	NULL	NULL


图 4.12 查询 SC 表信息


3) 查询视图 CS_View 的信息


SELECT * FROM CS_View;


1 • `SELECT * FROM CS_View;`

Result Grid



 Filter Rows:

Export: 

Wrap Cell Content: 

	Sno	Sname	Ssex	Sage	Sdept	Scholarship
▶	200215121	李勇	男	23	CS	否
	200215122	刘晨	女	22	CS	否

图 4.13 查询视图信息

4.2.7 用系统管理员登录，收回 U1 的所有权限

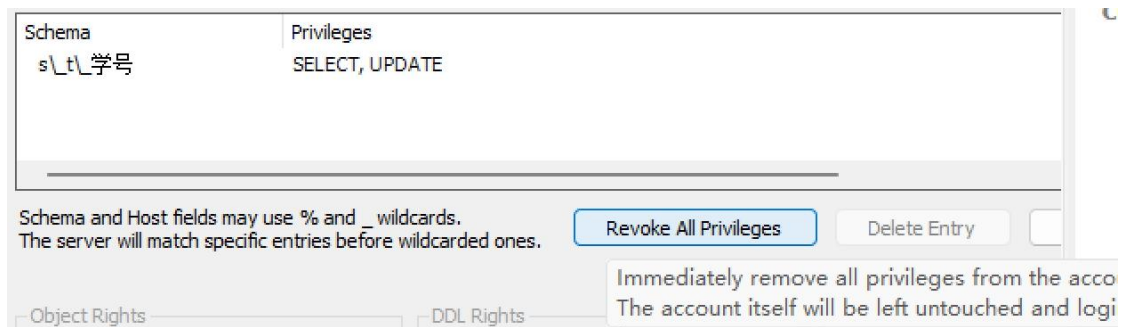


图 4.14 收回 U1 权限

4.2.8 用 U1 登录，查询学生表的信息

```
SELECT * FROM s_t_学号.student;
```

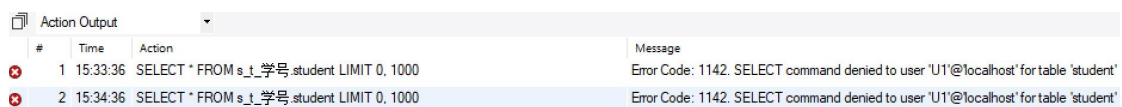


图 4.15 查询学生表信息

4.2.9 用系统管理员登录

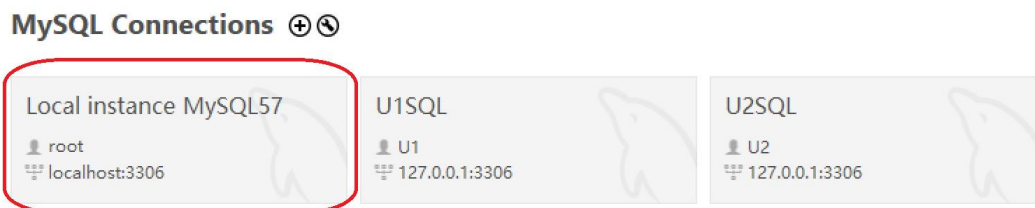


图 4.16 用管理员登录

4.2.10 对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩是大于 95 时，则把其奖学金修改为“否”。然后进行成绩修改，并进行验证是否触发器正确执行。1) 首先把某个学生成绩修改为 98，查询其奖学金。2) 再把刚才的成绩修改为 80，再查询其奖学金。

```
DELIMITER //
```

```
CREATE TRIGGER UpdateScholarship AFTER UPDATE ON SC  
FOR EACH ROW
```

```
BEGIN
```

```
-- 如果成绩更新后大于等于 95，检查奖学金状态，必要时更新为“是”
```

```
IF NEW.Grade >= 95 THEN
```

```
    UPDATE Student SET Scholarship = '是'
```

```
    WHERE Sno = NEW.Sno AND Scholarship = '否';
```

```

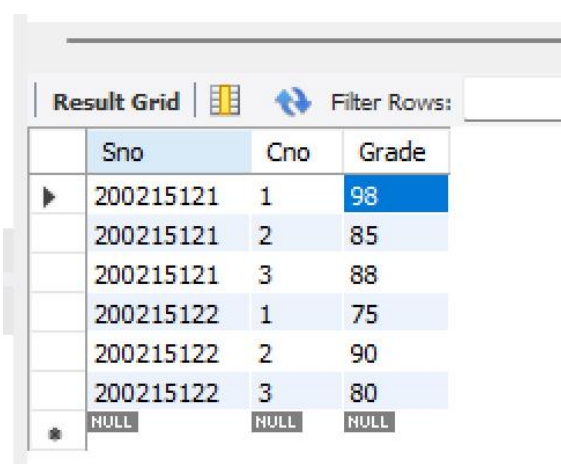
-- 如果成绩更新后小于 95，并且更新前的成绩是大于等于 95
ELSEIF NEW.Grade < 95 AND OLD.Grade >= 95 THEN
-- 检查是否存在其他成绩大于等于 95 的记录
IF NOT EXISTS (
    SELECT * FROM SC WHERE Sno = NEW.Sno AND Grade >= 95
) THEN
-- 如果没有其他成绩大于等于 95，将奖学金状态更新为“否”
UPDATE Student SET Scholarship = '否'
WHERE Sno = NEW.Sno;
END IF;
END IF;
END;
//

```

DELIMITER ;

1) 首先把某个学生成绩修改为 98，查询其奖学金。

UPDATE SC SET Grade = 98 WHERE Sno = '200215121' AND Cno = '1';

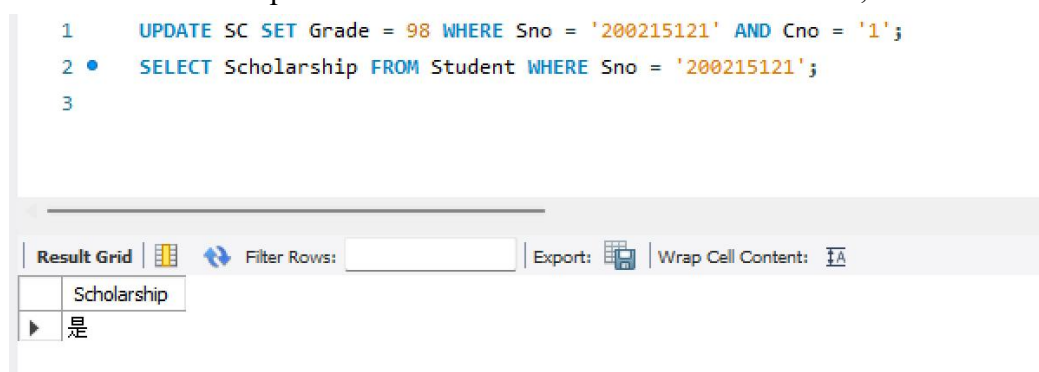


The screenshot shows a 'Result Grid' window with a table containing student records. The first row is highlighted in blue, showing a student with Sno '200215121' and Cno '1' having a Grade of 98. Other rows show the same student with different Cno values (2, 3) and other students with different Sno and Cno values.

	Sno	Cno	Grade
▶	200215121	1	98
	200215121	2	85
	200215121	3	88
	200215122	1	75
	200215122	2	90
	200215122	3	80
✱	NULL	NULL	NULL

图 4.17 修改成绩

SELECT Scholarship FROM Student WHERE Sno = '200215121';



The screenshot shows a 'Result Grid' window with a table containing the scholarship status for the student with Sno '200215121'. The first row is highlighted in blue, showing the Scholarship status as '是' (Yes).

	Scholarship
▶	是

图 4.18 查询奖学金

2) 再把刚才的成绩修改为 80，再查询其奖学金。

```
UPDATE SC SET Grade = 80 WHERE Sno = '200215121' AND Cno = '1';
SELECT Scholarship FROM Student WHERE Sno = '200215121';
```

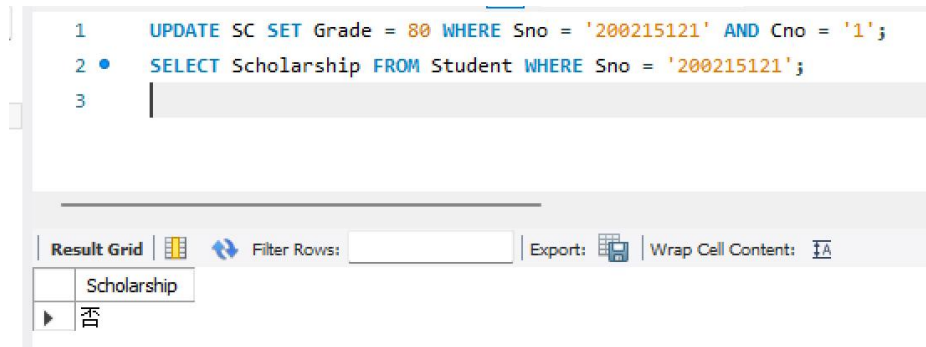


图 4.19 修改成绩查询奖学金

4.2.11 删除刚定义的触发器

```
DROP TRIGGER IF EXISTS UpdateScholarship;
```

```
1 SHOW TRIGGERS;
2
```

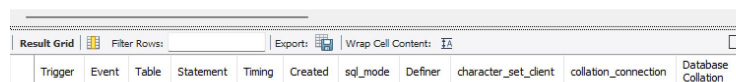


图 4.20 删除触发器

4.2.12 定义一个存储过程计算 CS 系的课程的平均成绩和最高成绩，在查询分析器或查询编辑器中执行存储过程，查看结果。

```
DELIMITER //
```

```
CREATE PROCEDURE CalculateCSStats()
BEGIN
    SELECT AVG(SC.Grade) AS 'Average Grade', MAX(SC.Grade) AS
'Maximum Grade'
    FROM SC
    JOIN Course ON SC.Cno = Course.Cno
    JOIN Student ON SC.Sno = Student.Sno
    WHERE Student.Sdept = 'CS';
END;

//

DELIMITER ;

CALL CalculateCSStats();
```

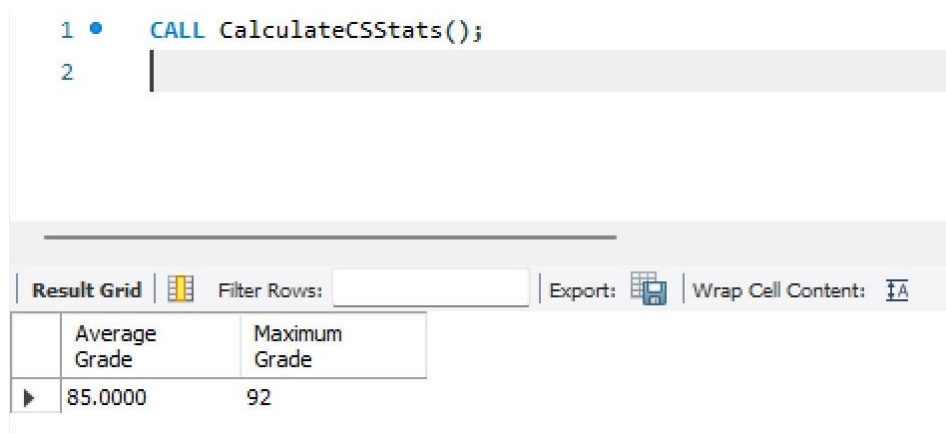



图 4.21 定义存储过程

4.2.13 定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓名。进行验证。

```
DELIMITER //
```

```
CREATE PROCEDURE GetStudentGrades(IN studentSno CHAR(9))
```

```
BEGIN
```

```
    SELECT Student.Sname, Course.Cname, SC.Grade
```

```
    FROM SC
```

```
    JOIN Course ON SC.Cno = Course.Cno
```

```
    JOIN Student ON SC.Sno = Student.Sno
```

```
    WHERE SC.Sno = studentSno;
```

```
END;
```

```
//
```

```
DELIMITER ;
```

```
CALL GetStudentGrades('200215121');
```

```
1 • CALL GetStudentGrades('200215121');
```

```
2
```

Sname	Cname	Grade
李勇	数据库	92
李勇	数学	85
李勇	信息系统	88

图 4.22 查询结果

4.2.14 把上一题改成函数。再进行验证。

MySQL 的函数仅能返回单一的值，而不是结果集。

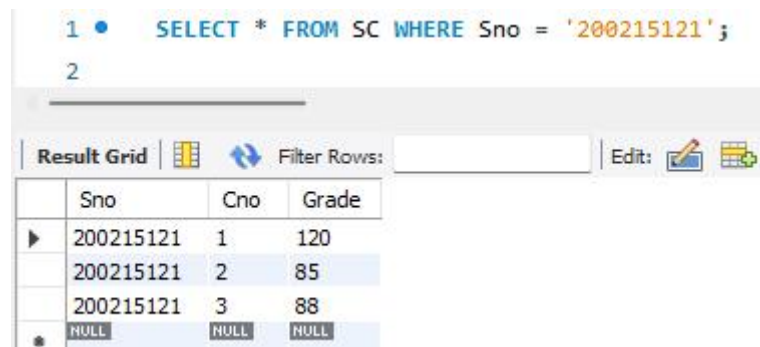
4.2.15 在 SC 表上定义一个完整性约束，要求成绩再 0-100 之间。定义约束前，先把某个学生的成绩修改成 120，进行查询，再修改回来。定义约束后，再把该学生成绩修改为 120，然后进行查询。

在定义约束之前，首先尝试更新成绩

```
UPDATE SC SET Grade = 120 WHERE Sno = '200215121' AND Cno = '1';
```

然后查询修改后的成绩

```
SELECT * FROM SC WHERE Sno = '200215121';
```



The screenshot shows a MySQL Workbench interface. At the top, a SQL query is entered: `SELECT * FROM SC WHERE Sno = '200215121';`. Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has four columns: 'Sno', 'Cno', and 'Grade'. There are three rows of data, all for 'Sno' = '200215121'. The first row has 'Cno' = 1 and 'Grade' = 120. The second row has 'Cno' = 2 and 'Grade' = 85. The third row has 'Cno' = 3 and 'Grade' = 88. There is also a row with 'Sno' = NULL, 'Cno' = NULL, and 'Grade' = NULL.

Sno	Cno	Grade
200215121	1	120
200215121	2	85
200215121	3	88
NULL	NULL	NULL

图 4. 23 查询成绩

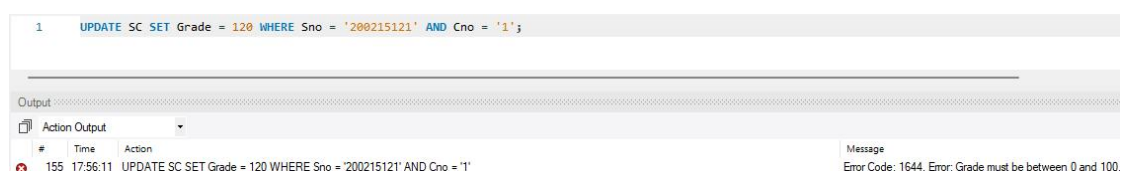
再把成绩改回合法范围

```
UPDATE SC SET Grade = 92 WHERE Sno = '200215121' AND Cno = '1';
```

定义完整性约束

```
ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100);
```

尝试再次修改成绩为非法值，失败



The screenshot shows a MySQL Workbench interface. At the top, a SQL query is entered: `UPDATE SC SET Grade = 120 WHERE Sno = '200215121' AND Cno = '1';`. Below the query editor, the 'Output' window is open, showing the results of the query. The output shows a message: 'Error Code: 1644. Error: Grade must be between 0 and 100.'

#	Time	Action	Message
155	17:56:11	UPDATE SC SET Grade = 120 WHERE Sno = '200215121' AND Cno = '1'	Error Code: 1644. Error: Grade must be between 0 and 100.

图 4. 24 修改失败

4.3 任务总结

在给 U1 授予 Student 表的查询和更新的权限，给 U2 对 SC 表授予插入的权限时遇到了问题，MySQL Workbench 可视化工具没有单独给某一个表授予权限的选项，只能给整个数据库授予权限。需要通过 MySQL 的命令行来完成对用户授予单独的表项的权限。

在定义完整性约束时遇到了如下图报错，查询后发现是 MySQL 版本过低，升级 MySQL 版本至 8.0 以上后成功解决了这个问题。



The screenshot shows a MySQL Workbench interface. At the top, a SQL query is entered: `ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100);`. Below the query editor, the 'Output' window is open, showing the results of the query. The output shows a message: 'Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.'

#	Time	Action	Message
43	14:21:31	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.
44	14:21:31	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.
45	14:21:31	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.
46	14:21:31	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.
47	14:21:32	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.
48	14:21:32	ALTER TABLE SC ADD CONSTRAINT chk_Grade CHECK (Grade BETWEEN 0 AND 100)	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.

5 数据库设计

5.1 任务要求

通过一个数据库具体设计实例，掌握数据库设计的方法。

- 1) 新生入学信息增加，学生信息修改。
- 2) 课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。
- 3) 录入学生成绩，修改学生成绩。
- 4) 按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。
- 5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。
- 6) 输入学号，显示该学生的基本信息和选课信息。

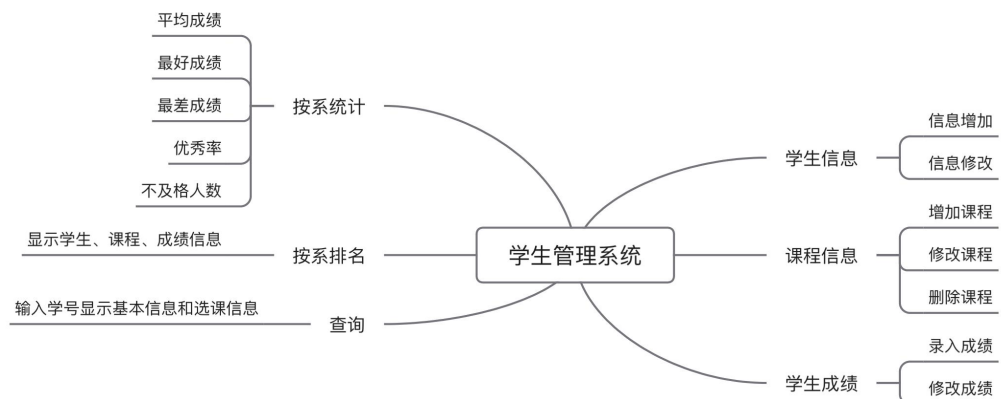


图 5.1 系统模块示意图

5.2 完成过程

5.2.1 系统开发环境

操作系统：Windows 10

数据库和驱动：MySQL+PyMySQL

开发工具：PyCharm

5.2.2 数据库概要

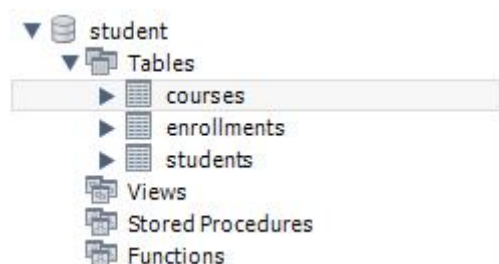


图 5.2 数据库结构示意图

表 5.1 数据库表项介绍及描述

英文表名	中文表名	描述
students	学生信息表	记录学号、姓名、性别、年龄、系
courses	课程信息表	记录课程号、课程名称
enrollments	成绩登记表	记录学号、课程号和对应的成绩

5.2.3 基本模块和实现

1) 新生入学信息增加，学生信息修改

```

1. Add New Student
2. Update Student Information
3. Add New Course
4. Update Course Information
5. Delete Course
6. Add Grade
7. Update Grade
8. Calculate Statistics
9. Rank by Department
10. Show Student Info
11. Exit

```

```

Choose an option: 1
Enter student number: 202112333
Enter name: Lucy
Enter gender: 女
Enter age: 21
Enter department: 物理
Student added successfully.

```

图 5.3 新生入学信息增加操作

	sno	name	gender	age	department
	202016777	Jack	男	21	数学
	202111233	y	男	22	cse
	202111992	z	女	21	cse
	202111999	r	男	22	cs
▶	202112333	Lucy	女	21	物理
	202112992	p	女	20	cs
★	NULL	NULL	NULL	NULL	NULL

图 5.4 新生入学信息增加结果

```

Choose an option: 2
Enter student number to update: 202112333
Current Information: {'sno': '202112333', 'name': 'Lucy', 'gender': '女', 'age': 21, 'department': '物理'}
Enter new name (leave blank to keep unchanged):
Enter new gender (leave blank to keep unchanged):
Enter new age (leave blank to keep unchanged): 20
Enter new department (leave blank to keep unchanged): 数学
Student information updated successfully.

```

图 5.5 学生信息修改操作

	sno	name	gender	age	department
	202016777	Jack	男	21	数学
	202111233	y	男	22	cse
	202111992	z	女	21	cse
	202111999	r	男	22	cs
	202112333	Lucy	女	20	数学
	202112992	p	女	20	cs
▶*	NULL	NULL	NULL	NULL	

图 5.6 学生信息修改结果

2) 课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。

```
Choose an option: 3
Enter course number: 3
Enter course name: 微积分
Course added successfully.
```

图 5.7 增加新课程操作

	cno	cname
▶	1	cdesign
	2	sql
	3	微积分
*	NULL	NULL

图 5.8 增加新课程结果

```
Choose an option: 4
Enter course number to update: 3
Enter new course name: 大学物理
Course updated successfully.
```

图 5.9 修改课程信息操作

	cno	cname
▶	1	cdesign
	2	sql
	3	大学物理
*	NULL	NULL

图 5.10 修改课程信息结果

```
Choose an option: 5
Enter course number to delete: 3
Course deleted successfully.
```

图 5.11 删除没有选课的课程信息操作

	cno	cname
▶	1	cdesign
	2	sql
*	NULL	NULL

图 5.12 删除没有选课的课程信息结果

3) 录入学生成绩，修改学生成绩。

```

Choose an option: 6
Enter student number: 202112333
Enter course number: 4
Enter grade: 99
Grade added successfully.

Choose an option: 6
Enter student number: 202112333
Enter course number: 3
Enter grade: 96
Grade added successfully.

```

图 5.13 录入学生成绩操作

	sno	cno	grade
	202111233	1	88.00
	202111233	2	96.00
	202111992	1	99.00
	202111992	2	100.00
	202111999	1	96.00
	202112333	3	96.00
▶	202112333	4	99.00
	202112992	1	98.00
*	NULL	NULL	NULL

图 5.14 录入学生成绩结果

```

Choose an option: 7
Enter student number: 202112333
Enter course number: 3
Enter new grade: 90
Grade updated successfully.

```

图 5.15 修改学生成绩操作

	sno	cno	grade
▶	202111233	1	88.00
	202111233	2	96.00
	202111992	1	99.00
	202111992	2	100.00
	202111999	1	96.00
	202112333	3	90.00
	202112333	4	99.00
	202112992	1	98.00
*	NULL	NULL	NULL

图 5.16 修改学生成绩结果

4) 按系统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。

```

Choose an option: 8
Department: 数学
  Average Grade: 78.50
  Highest Grade: 99.00
  Lowest Grade: 59.00
  Excellence Rate: 50.00%
  Fail Count: 1

Department: cse
  Average Grade: 90.12
  Highest Grade: 100.00
  Lowest Grade: 59.00
  Excellence Rate: 62.50%
  Fail Count: 1

Department: cs
  Average Grade: 84.25
  Highest Grade: 98.00
  Lowest Grade: 55.00
  Excellence Rate: 50.00%
  Fail Count: 1

```

图 5.17 按系统计

5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。

Choose an option: 9

Department	Course	Student No	Name	Grade	Rank
cs	cdesign	202112992	p	98.00	1
cs	cdesign	202111999	r	96.00	2
cs	sql	202112992	p	55.00	1
cs	微积分	202112992	p	88.00	1
cse	cdesign	202111992	z	99.00	1
cse	cdesign	202012656	小明	97.00	2
cse	cdesign	202111233	y	88.00	3
cse	sql	202111992	z	100.00	1
cse	sql	202111233	y	96.00	2
cse	sql	202012656	小明	92.00	3
cse	大学物理	202111992	z	98.00	1
cse	大学物理	202012656	小明	93.00	2
cse	大学物理	202111233	y	82.00	3
cse	微积分	202111992	z	99.00	1
cse	微积分	202012656	小明	90.00	2
cse	微积分	202111233	y	59.00	3
数学	大学物理	202112333	Lucy	99.00	1
数学	大学物理	202016777	Jack	66.00	2
数学	微积分	202112333	Lucy	90.00	1
数学	微积分	202016777	Jack	59.00	2

图 5.18 按系排名

6) 输入学号，显示该学生的基本信息和选课信息。

Choose an option: 10
Enter student number: 202111992

Student Information:
Student Number: 202111992
Name: z
Gender: 女
Age: 21
Department: cse

Courses and Grades:
Course: cdesign - Grade: 99.00
Course: sql - Grade: 100.00
Course: 微积分 - Grade: 99.00
Course: 大学物理 - Grade: 98.00

图 5.19 输入学号显示信息

5.3 任务总结

相比前面的任务只需要用到基本的 SQL 命令，最后一个任务更为复杂，需要实现 MySQL 与 Java 或者 Python 结合的数据库应用开发。我参考了《MySQL 从入门到精通》这本书，选择了 Python + MySQL 实现数据库系统的开发。

在完成的过程中遇到了不少困难，比如无法使用 SQL 语句中的排序命令，

查找报错后发现是 MySQL 版本过低,但是更新版本后依然报错,最后通过 Python 编写代码实现排序功能才解决了这个问题。

6 课程总结

略

附录

```
import pymysql

def connect_db():
    return pymysql.connect(
        host='localhost',
        user='root',
        password='123456', #数据库密码
        db='student',
        charset='utf8mb4',
        cursorclass=pymysql.cursors.DictCursor
    )

def initialize_db():
    """创建数据库和表结构"""
    db = connect_db()
    with db.cursor() as cursor:
        cursor.execute("CREATE DATABASE IF NOT EXISTS student")
        cursor.execute("USE student")
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS students (
                sno VARCHAR(10) NOT NULL,
                name VARCHAR(100) NOT NULL,
                gender VARCHAR(10) NOT NULL,
                age INT,
                department VARCHAR(100),
                PRIMARY KEY (sno)
            )
        """)
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS courses (
                cno VARCHAR(10) NOT NULL,
                cname VARCHAR(100) NOT NULL,
                PRIMARY KEY (cno)
            )
        """)
        cursor.execute("""
            CREATE TABLE IF NOT EXISTS enrollments (
                sno VARCHAR(10),
                cno VARCHAR(10),
                grade DECIMAL(5, 2),
                PRIMARY KEY (sno, cno),
                FOREIGN KEY (sno) REFERENCES students(sno),
                FOREIGN KEY (cno) REFERENCES courses(cno)
            )
        """)
    db.commit()
    db.close()
```

1) 新生入学信息增加, 学生信息修改。

```
def add_student():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sql = "INSERT INTO students (sno, name, gender, age, department)
VALUES (%s, %s, %s, %s, %s)"
            sno = input("Enter student number: ")
            name = input("Enter name: ")
            gender = input("Enter gender: ")
            age = input("Enter age: ")
            department = input("Enter department: ")
            cursor.execute(sql, (sno, name, gender, age, department))
            db.commit()
            print("Student added successfully.")
    finally:
        db.close()

def update_student():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sno = input("Enter student number to update: ")

            # 获取当前学生的信息
            cursor.execute("SELECT * FROM students WHERE sno = %s", (sno,))
            current_data = cursor.fetchone()
            if not current_data:
                print("No student found with the provided student number.")
                return

            print(f"Current Information: {current_data}")

            # 用户选择要更新的信息
            name = input("Enter new name (leave blank to keep unchanged): ")
            gender = input("Enter new gender (leave blank to keep unchanged): ")
            age = input("Enter new age (leave blank to keep unchanged): ")
            department = input("Enter new department (leave blank to keep
unchanged): ")

            # 构建更新 SQL 语句
            update_fields = []
            data_to_update = []

            if name:
                update_fields.append("name = %s")
                data_to_update.append(name)
            if gender:
                update_fields.append("gender = %s")
```

```

        data_to_update.append(gender)
    if age:
        update_fields.append("age = %s")
        data_to_update.append(age)
    if department:
        update_fields.append("department = %s")
        data_to_update.append(department)

    if not update_fields:
        print("No updates were made as no new information was
provided.")
        return

    # 更新数据库
    update_query = "UPDATE students SET " + ", ".join(update_fields) +
" WHERE sno = %s"
    data_to_update.append(sno)
    cursor.execute(update_query, tuple(data_to_update))
    db.commit()

    print("Student information updated successfully.")
finally:
    db.close()

```

2) 课程信息维护

```

def add_course():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sql = "INSERT INTO courses (cno, cname) VALUES (%s, %s)"
            cno = input("Enter course number: ")
            cname = input("Enter course name: ")
            cursor.execute(sql, (cno, cname))
            db.commit()
            print("Course added successfully.")
    finally:
        db.close()

def update_course():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            cno = input("Enter course number to update: ")
            cname = input("Enter new course name: ")
            cursor.execute("UPDATE courses SET cname = %s WHERE cno
= %s", (cname, cno))
            db.commit()
            print("Course updated successfully.")
    finally:

```

```

        db.close()

def delete_course():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            cno = input("Enter course number to delete: ")
            cursor.execute("DELETE FROM courses WHERE cno = %s AND cno
NOT IN (SELECT cno FROM enrollments)", (cno,))
            db.commit()
            print("Course deleted successfully.")
    finally:
        db.close()

```

3) 录入学生成绩，修改学生成绩

```

def add_grade():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sno = input("Enter student number: ")
            cno = input("Enter course number: ")
            grade = input("Enter grade: ")
            sql = "INSERT INTO enrollments (sno, cno, grade) VALUES
(%s, %s, %s)"
            cursor.execute(sql, (sno, cno, grade))
            db.commit()
            print("Grade added successfully.")
    finally:
        db.close()

```

```

def update_grade():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sno = input("Enter student number: ")
            cno = input("Enter course number: ")
            grade = input("Enter new grade: ")
            cursor.execute("UPDATE enrollments SET grade = %s WHERE sno
= %s AND cno = %s", (grade, sno, cno))
            db.commit()
            print("Grade updated successfully.")
    finally:
        db.close()

```

4) 按系统统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数

```

def calculate_statistics():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            cursor.execute("""

```

```

SELECT
    s.department,
    AVG(e.grade) AS average,
    MAX(e.grade) AS highest,
    MIN(e.grade) AS lowest,
    SUM(CASE WHEN e.grade >= 90 THEN 1 ELSE 0 END)
AS excellence_count,
    SUM(CASE WHEN e.grade < 60 THEN 1 ELSE 0 END)
AS fail_count,
    COUNT(e.grade) AS total_count
FROM enrollments e
JOIN students s ON e.sno = s.sno
GROUP BY s.department
"""
)
results = cursor.fetchall()

if results:
    for result in results:
        if result['total_count'] > 0:
            excellence_rate = (result['excellence_count'] /
result['total_count']) * 100
            print(f"Department: {result['department']}")
            print(f"  Average Grade: {result['average']:.2f}")
            print(f"  Highest Grade: {result['highest']}")
            print(f"  Lowest Grade: {result['lowest']}")
            print(f"  Excellence Rate: {excellence_rate:.2f}%")
            print(f"  Fail Count: {result['fail_count']}\n")
        else:
            print(f"Department: {result['department']} - No grade
data available to calculate statistics.")
    else:
        print("No grade data available in any department.")

finally:
    db.close()

```

5) 按系对学生成绩进行排名，同时显示出学生、课程和成绩信息

def rank_by_department():

""" 按部门和课程对学生成绩进行排名 """

db = connect_db()

try:

with db.cursor() as cursor:

查询每个学生的学号、系、姓名、课程名称和成绩，并按系、课程和成绩降序排列

cursor.execute("""

```

SELECT s.sno, s.department, s.name, c.cname, e.grade
FROM enrollments e
JOIN students s ON e.sno = s.sno
JOIN courses c ON e.cno = c.cno
ORDER BY s.department, c.cname, e.grade DESC

```

```

        """)
    results = cursor.fetchall()

    # 初始化排名和上一个记录的变量
    last_department = None
    last_course = None
    last_grade = None
    department_course_rank = 1 # 部门和课程内的排名

    # 格式化输出标题
    print(f'{"Department":15} | {"Course":15} | {"Student No":10} | {"Name":20} | {"Grade":6} | {"Rank":4}')
    print("-" * 80) # 输出分隔线

    # 输出结果并计算排名
    for row in results:
        if row['department'] != last_department or row['cname'] != last_course:
            # 如果部门或课程改变了，重置部门和课程内的排名
            department_course_rank = 1
            last_department = row['department']
            last_course = row['cname']
        elif row['grade'] != last_grade:
            # 如果成绩变了，在同一部门和课程内增加排名
            department_course_rank += 1

        # 更新最后一个记录的成绩，为下一行比较做准备
        last_grade = row['grade']

        # 格式化输出每行数据
        print(f'{row["department"]:15} | {row["cname"]:15} | {row["sno"]:10} | {row["name"]:20} | {row["grade"]:6} | {department_course_rank:4}')
    finally:
        db.close()

```

6) 输入学号，显示该学生的基本信息和选课信息

```

def show_student_info():
    db = connect_db()
    try:
        with db.cursor() as cursor:
            sno = input("Enter student number: ")
            cursor.execute("SELECT * FROM students WHERE sno = %s", (sno,))
            student_info = cursor.fetchone()

            if student_info:
                # 格式化并输出学生基本信息
                print("\nStudent Information:")
                print(f"Student Number: {student_info['sno']}")

```

```

        print(f"   Name: {student_info['name']}")
        print(f"   Gender: {student_info['gender']}")
        print(f"   Age: {student_info['age']}")
        print(f"   Department: {student_info['department']}")

        # 查询并输出学生的课程信息
        cursor.execute(
            "SELECT c.cname, e.grade FROM enrollments e JOIN
courses c ON e.cno = c.cno WHERE e.sno = %s",
            (sno,))
        courses = cursor.fetchall()
        if courses:
            print("\nCourses and Grades:")
            for course in courses:
                print(f"   Course: {course['cname']} - Grade:
{course['grade']}")
        else:
            print("   No course information available.")
    else:
        print("No student found with that student number.")

finally:
    db.close()

```

```

# 主程序
def main():
    while True:
        print("""
1. Add New Student
2. Update Student Information
3. Add New Course
4. Update Course Information
5. Delete Course
6. Add Grade
7. Update Grade
8. Calculate Statistics
9. Rank by Department
10. Show Student Info
11. Exit
""")
        choice = int(input("Choose an option: "))
        if choice == 1:
            add_student()
        elif choice == 2:
            update_student()
        elif choice == 3:
            add_course()
        elif choice == 4:
            update_course()

```



```
elif choice == 5:
    delete_course()
elif choice == 6:
    add_grade()
elif choice == 7:
    update_grade()
elif choice == 8:
    calculate_statistics()
elif choice == 9:
    rank_by_department()
elif choice == 10:
    show_student_info()
elif choice == 11:
    break
else:
    print("Invalid option")

if __name__ == "__main__":
    main()
```