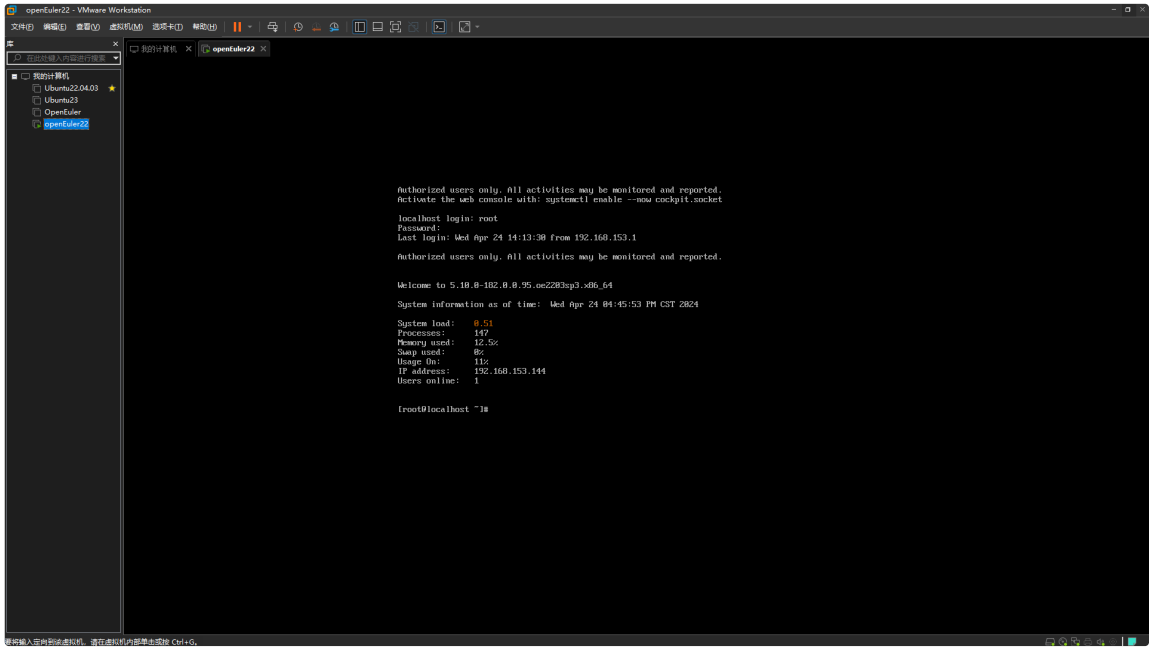


数据库实验

计算机2101 陈实 2215015058

实验平台

1. 操作系统：VMWare中安装的虚拟机openEuler 22.03



2. 数据库： openGauss 5.0

```
[opengauss@localhost ~]$ gaussdb -V
gaussdb (GaussDB Kernel V500R002C00 build ) compiled at 2023-12-28 14:27:08 commit 0 last mr
```

3. 数据库客户端：DataGrip+JDBC驱动

实验内容

第一题

1. 创建DATABASE：MYDB

```
1 | CREATE DATABASE MYDB;
```

2. 修改连接到MYDB

3. 创建表格

1. 对于学生表格S，观察可知Sno为定长8位字符，且存在前导0，所以选择CHAR(8)类型；Sname为变长字符，选择VARCHAR(30)类型；Sex为定长字符，选择CHAR(6)类型；Bdate为日期类型，选择DATE类型；Height为身高，选择DECIMAL(3,2)类型；Dorm为宿舍号，选择VARCHAR(20)类型。注意将Sno设置为主键。

```

1  --Sxxx (S#, SNAME, SEX, BDATE, HEIGHT, DORM)
2  CREATE TABLE S058 (
3      Sno CHAR(8) NOT NULL PRIMARY KEY,
4      Sname VARCHAR(30) NOT NULL,
5      Sex VARCHAR(6),
6      Bdate DATE,
7      Height DECIMAL(3,2),
8      Dorm VARCHAR(20)
9  );

```

2. 对于课程表格C, 观察可知**Cno为定长5位字符, 所以选择CHAR(5)类型**; Cname为变长字符, 选择VARCHAR(100)类型; Period为学时, 选择SMALLINT类型; Credit为学分, 选择DECIMAL(3,1)类型; Teacher为教师名, 选择VARCHAR(50)类型。将Cno设置为主键。

```

1  --Cxxx (C#, CNAME, PERIOD, CREDIT, TEACHER)
2  CREATE TABLE C058 (
3      Cno CHAR(6) NOT NULL PRIMARY KEY,
4      Cname VARCHAR(100) NOT NULL,
5      Period SMALLINT,
6      Credit DECIMAL(3,1),
7      Teacher VARCHAR(50)
8  );

```

3. 对于选课表格SC, 观察可知**Sno和Cno为外键**; Grade为成绩, 范围为0-100, 精确到小数点后一位, 选择DECIMAL(4,1)类型。将Sno和Cno设置为联合主键, 同时设置外键约束。

```

1  --SCxxx (S#, C#, GRADE) 其中 S#、C#均为外键
2  CREATE TABLE SC058 (
3      Sno CHAR(8) NOT NULL,
4      Cno CHAR(6) NOT NULL,
5      Grade DECIMAL(4,1),
6      PRIMARY KEY (Sno, Cno),
7      FOREIGN KEY (Sno) REFERENCES S058(Sno),
8      FOREIGN KEY (Cno) REFERENCES C058(Cno)
9  );

```

运行结果:

```
[2024-06-11 20:39:04] 在 5 ms 内完成
mydb.public> CREATE TABLE S058 (
    Sno CHAR(8) NOT NULL PRIMARY KEY,
    Sname VARCHAR(30) NOT NULL,
    Sex VARCHAR(6),
    Bdate DATE,
    Height DECIMAL(3,2),
    Dorm VARCHAR(20)
)
CREATE TABLE / PRIMARY KEY will create implicit index "s058_pkey" for table "s058"
[2024-06-11 20:39:04] 在 24 ms 内完成
mydb.public> CREATE TABLE C058 (
    Cno CHAR(6) NOT NULL PRIMARY KEY,
    Cname VARCHAR(100) NOT NULL,
    Period SMALLINT,
    Credit DECIMAL(3,1),
    Teacher VARCHAR(50)
)
CREATE TABLE / PRIMARY KEY will create implicit index "c058_pkey" for table "c058"
[2024-06-11 20:39:04] 在 6 ms 内完成
mydb.public> CREATE TABLE SC058 (
    Sno CHAR(8) NOT NULL,
    Cno CHAR(6) NOT NULL,
    Grade DECIMAL(4,1),
    PRIMARY KEY (Sno, Cno),
    FOREIGN KEY (Sno) REFERENCES S058(Sno),
    FOREIGN KEY (Cno) REFERENCES C058(Cno)
)
CREATE TABLE / PRIMARY KEY will create implicit index "sc058_pkey" for table "sc058"
```

第二题

1. 插入数据

```
1  INSERT INTO S058 VALUES ('01032010', '王涛', '男', '2003-4-5', 1.72, '东6舍221'),
2  ('01032023', '孙文', '男', '2004-6-10', 1.80, '东6舍221'),
3  ('01032001', '张晓梅', '女', '2004-11-17', 1.58, '东1舍312'),
4  ('01032005', '刘静', '女', '2003-1-10', 1.63, '东1舍312'),
5  ('01032112', '董蔚', '男', '2003-2-20', 1.71, '东6舍221'),
6  ('03031011', '王倩', '女', '2004-12-20', 1.66, '东2舍104'),
7  ('03031014', '赵思扬', '男', '2002-6-6', 1.85, '东18舍421'),
8  ('03031051', '周剑', '男', '2002-5-8', 1.68, '东18舍422'),
9  ('03031009', '田菲', '女', '2003-8-11', 1.60, '东2舍104'),
10 ('03031033', '蔡明明', '男', '2003-3-12', 1.75, '东18舍423'),
11 ('03031056', '曹子衿', '女', '2004-12-15', 1.65, '东2舍305');
```

```
mydb.public> INSERT INTO S058 VALUES ('01032010', '王涛', '男', '2003-4-5', 1.72, '东6舍221'),
    ('01032023', '孙文', '男', '2004-6-10', 1.80, '东6舍221'),
    ('01032001', '张晓梅', '女', '2004-11-17', 1.58, '东1舍312'),
    ('01032005', '刘静', '女', '2003-1-10', 1.63, '东1舍312'),
    ('01032112', '董蔚', '男', '2003-2-20', 1.71, '东6舍221'),
    ('03031011', '王倩', '女', '2004-12-20', 1.66, '东2舍104'),
    ('03031014', '赵思扬', '男', '2002-6-6', 1.85, '东18舍421'),
    ('03031051', '周剑', '男', '2002-5-8', 1.68, '东18舍422'),
    ('03031009', '田菲', '女', '2003-8-11', 1.60, '东2舍104'),
    ('03031033', '蔡明明', '男', '2003-3-12', 1.75, '东18舍423'),
    ('03031056', '曹子衿', '女', '2004-12-15', 1.65, '东2舍305')
[2024-06-11 20:39:05] 7 ms 中有 11 行受到影响
```

```
1 INSERT INTO C058 VALUES
2 ('CS-01', '数据结构', 60, 3, '张军'),
3 ('CS-02', '计算机组成原理', 80, 4, '王亚伟'),
4 ('CS-04', '人工智能', 40, 2, '李蕾'),
5 ('CS-05', '深度学习', 40, 2, '崔昀'),
6 ('EE-01', '信号与系统', 60, 3, '张明'),
7 ('EE-02', '数字逻辑电路', 100, 5, '胡海东'),
8 ('EE-03', '光电子学与光子学', 40, 2, '石韬');
```

```
mydb.public> INSERT INTO C058 VALUES
      ('CS-01', '数据结构', 60, 3, '张军'),
      ('CS-02', '计算机组成原理', 80, 4, '王亚伟'),
      ('CS-04', '人工智能', 40, 2, '李蕾'),
      ('CS-05', '深度学习', 40, 2, '崔昀'),
      ('EE-01', '信号与系统', 60, 3, '张明'),
      ('EE-02', '数字逻辑电路', 100, 5, '胡海东'),
      ('EE-03', '光电子学与光子学', 40, 2, '石韬')
[2024-06-11 20:39:05] 6 ms 中有 7 行受到影响
```

```
1 INSERT INTO SC058 VALUES
2 ('01032010', 'CS-01', 82.0),
3 ('01032010', 'CS-02', 91.0),
4 ('01032010', 'CS-04', 83.5),
5 ('01032001', 'CS-01', 77.5),
6 ('01032001', 'CS-02', 85.0),
7 ('01032001', 'CS-04', 83.0),
8 ('01032005', 'CS-01', 62.0),
9 ('01032005', 'CS-02', 77.0),
10 ('01032005', 'CS-04', 82.0),
11 ('01032023', 'CS-01', 55.0),
12 ('01032023', 'CS-02', 81.0),
13 ('01032023', 'CS-04', 76.0),
14 ('01032112', 'CS-01', 88.0),
15 ('01032112', 'CS-02', 91.5),
16 ('01032112', 'CS-04', 86.0),
17 ('01032112', 'CS-05', NULL),
18 ('03031033', 'EE-01', 93.0),
19 ('03031033', 'EE-02', 89.0),
20 ('03031009', 'EE-01', 88.0),
21 ('03031009', 'EE-02', 78.5),
22 ('03031011', 'EE-01', 91.0),
23 ('03031011', 'EE-02', 86.0),
24 ('03031051', 'EE-01', 78.0),
25 ('03031051', 'EE-02', 58.0),
26 ('03031014', 'EE-01', 79.0),
27 ('03031014', 'EE-02', 71.0);
28
```

```

mydb.public> INSERT INTO SC058 VALUES
('01032010', 'CS-01', 82.0),
('01032010', 'CS-02', 91.0),
('01032010', 'CS-04', 83.5),
('01032001', 'CS-01', 77.5),
('01032001', 'CS-02', 85.0),
('01032001', 'CS-04', 83.0),
('01032005', 'CS-01', 62.0),
('01032005', 'CS-02', 77.0),
('01032005', 'CS-04', 82.0),
('01032023', 'CS-01', 55.0),
('01032023', 'CS-02', 81.0),
('01032023', 'CS-04', 76.0),
('01032112', 'CS-01', 88.0),
('01032112', 'CS-02', 91.5),
('01032112', 'CS-04', 86.0),
('01032112', 'CS-05', NULL),
('03031033', 'EE-01', 93.0),
('03031033', 'EE-02', 89.0),
('03031009', 'EE-01', 88.0),
('03031009', 'EE-02', 78.5),
('03031011', 'EE-01', 91.0),
('03031011', 'EE-02', 86.0),
('03031051', 'EE-01', 78.0),
('03031051', 'EE-02', 58.0),
('03031014', 'EE-01', 79.0),
('03031014', 'EE-02', 71.0)
[2024-06-11 20:39:05] 6 ms 中有 26 行受到影响

```

第三题

第一小题

1. 查询电子工程系（EE）所开课程的课程编号、课程名称及学分数

```

1  --查询电子工程系（EE）所开课程的课程编号、课程名称及学分数
2  SELECT C058.Cno, C058.Cname, C058.Credit
3  FROM C058
4  WHERE C058.Cno LIKE 'EE%';

```

	cno	cname	credit
1	EE-01	信号与系统	3.0
2	EE-02	数字逻辑电路	5.0
3	EE-03	光电子学与光子学	2.0

2. 查询未选修课程“CS-02”的女生学号及其已选各课程编号、成绩。

```

1  --查询未选课程“CS-02”的女生学号及其已选各课程编号、成绩。
2  SELECT SC058.Sno, SC058.Cno, SC058.Grade
3  FROM SC058
4  WHERE SC058.Cno ≠ 'CS-02' AND SC058.Sno IN (
5      SELECT S058.Sno
6      FROM S058
7      WHERE S058.Sex = '女'
8  ) ORDER BY SC058.Sno;

```

	sno	cno	grade
1	01032001	CS	77.5
2	01032001	CS-04	83.0
3	01032005	CS-01	62.0
4	01032005	CS-04	82.0
5	03031009	EE-01	88.0
6	03031009	EE-02	78.5
7	03031011	EE-01	91.0
8	03031011	EE-02	86.0

3. 查询 2002 年~2003 年出生学生的基本信息。

```

1  --查询 2002 年~2003 年出生学生的基本信息。
2  SELECT *
3  FROM S058
4  WHERE S058.Bdate BETWEEN '2002-01-01' AND '2003-12-31';

```

	cno	cname	credit
1	EE-01	信号与系统	3.0
2	EE-02	数字逻辑电路	5.0
3	EE-03	光电子学与光子学	2.0

4. 查询每位学生的学号、学生姓名及其已选课程的学分总数。

本题应该考虑到部分同学没有选修任何课程，如果使用 WHERE S058.Sno = SC058.Sno AND C058.Cno = SC058.Cno 来进行连接，会导致没有选课的同学不会出现在查询结果中，所以应当使用做左连接来连接各个表。同时使用 COALESCE(SUM(C058.Credit), 0) 来求和以及将 NULL 转化为 0。

```

1  --查询每位学生的学号、学生姓名及其已选课程的学分总数。
2  SELECT S058.Sno, S058.Sname, COALESCE(SUM(C058.Credit), 0) AS
   TotalCredits
3  FROM S058
4  LEFT JOIN SC058 ON S058.Sno = SC058.Sno -- 左连接
5  LEFT JOIN C058 ON SC058.Cno = C058.Cno -- 左连接
6  GROUP BY S058.Sno;

```

	sno	sname	totalcredits
1	03031014	赵思扬	8
2	03031033	蔡明明	8
3	03031009	田菲	8
4	03031011	王倩	8
5	03031051	周剑	8
6	01032001	张晓梅	9
7	01032005	刘静	9
8	01032023	孙文	9
9	03031056	曹子衿	0
10	01032112	董蔚	11
11	01032010	王涛	9

5. 查询选修课程“CS-01”的学生中成绩第二高的学生学号。

明确成绩第二高的学生可能有多个，所以需要考虑是否需要返回多个学生的学号。找出第二高成绩，然后找出成绩等于第二高成绩的学生。

```

1  --查询选修课程“CS-01”的学生中成绩第二高的学生学号。
2  SELECT SC058.Sno
3  FROM SC058
4  WHERE SC058.Cno = 'CS-01' AND SC058.Grade = (
5      SELECT MAX(SC058.Grade)
6      FROM SC058
7      WHERE SC058.Cno = 'CS-01' AND SC058.Grade < (
8          SELECT MAX(SC058.Grade)
9          FROM SC058
10         WHERE SC058.Cno = 'CS-01'
11     )
12 );

```

sno
01032010

6. 查询平均成绩超过“王涛”同学的学生学号、姓名和平均成绩，并按学号进行降序排列。

由于王涛的姓名可能有多个，所以需要先查询出所有叫“王涛”的学生，然后让用户选择其中一个学生，再查询平均成绩超过该学生的学生。

下面给出一个Java程序来实现该功能。

```

1  import java.sql.*;
2  import java.util.Scanner;
3
4  public class StudentGradeComparison {
5
6      public static void main(String[] args) {
7          String url = "jdbc:postgresql://127.0.0.1:7654/mydb";
8          String username = "*****"; // 数据库用户名
9          String password = "*****"; // 数据库密码
10
11         // 查询所有叫“王涛”的学生

```

```

12         String queryWangTao = "SELECT Sno, Sname FROM S058 WHERE Sname
    = '王涛'";
13
14         try (Connection conn = DriverManager.getConnection(url,
    username, password);
15             Statement stmt = conn.createStatement()) {
16
17             // 执行查询, 获取叫“王涛”的学生列表
18             ResultSet rs = stmt.executeQuery(queryWangTao);
19
20             System.out.println("以下是所有叫‘王涛’的学生: ");
21             while (rs.next()) {
22                 System.out.println("学号: " + rs.getString("Sno") + ",
    姓名: " + rs.getString("Sname"));
23             }
24
25             // 让用户输入学号指定学生
26             Scanner scanner = new Scanner(System.in);
27             System.out.print("请输入上面其中一个学生的学号: ");
28             String selectedSno = scanner.nextLine();
29
30             // 获取指定学生的平均成绩
31             String queryAvgGrade = "SELECT AVG(Grade) as AvgGrade FROM
    SC058 WHERE Sno = '" + selectedSno + "'";
32             ResultSet rsAvgGrade = stmt.executeQuery(queryAvgGrade);
33
34             double selectedAvgGrade = 0.0;
35             if (rsAvgGrade.next()) {
36                 selectedAvgGrade = rsAvgGrade.getDouble("AvgGrade");
37                 System.out.println("学号 " + selectedSno + " 的平均成绩是:
    " + selectedAvgGrade);
38             }
39
40             // 查询平均成绩超过指定学生的其他学生, 并按学号降序排列
41             String queryStudentsExceed = "SELECT S058.Sno, S058.Sname,
    AVG(SC058.Grade) as AvgGrade "
42                 + "FROM SC058 INNER JOIN S058 ON SC058.Sno =
    S058.Sno "
43                 + "GROUP BY S058.Sno, S058.Sname "
44                 + "HAVING AVG(SC058.Grade) > " + selectedAvgGrade
45                 + " ORDER BY S058.Sno DESC";
46
47             ResultSet rsExceed =
    stmt.executeQuery(queryStudentsExceed);
48
49             System.out.println("平均成绩超过 " + selectedSno + " 的学
    生: ");
50             while (rsExceed.next()) {
51                 System.out.println("学号: " + rsExceed.getString("Sno")
    + ", 姓名: " + rsExceed.getString("Sname") + ", 平均成绩: " +
    rsExceed.getDouble("AvgGrade"));
52             }
53
54             } catch (SQLException e) {

```



```

55         e.printStackTrace();
56     }
57 }
58 }

```

Java程序运行结果:

```

以下是所有叫'王涛'的学生:
学号: 01032010, 姓名: 王涛
请输入上面其中一个学生的学号: 01032010
学号 01032010 的平均成绩是: 85.5
平均成绩超过 01032010 的学生:
学号: 03031033, 姓名: 蔡明明, 平均成绩: 91.0
学号: 03031011, 姓名: 王倩, 平均成绩: 88.5
学号: 01032112, 姓名: 董蔚, 平均成绩: 88.5

```

7. 查询选修了计算机专业全部课程（课程编号为“CS-xx”）的学生姓名及已获得的学分总数。

```

1  --查询选修了计算机专业全部课程（课程编号为“CS-xx”）的学生姓名及已获得的学分总数。
2  SELECT S058.Sname, SUM(C058.Credit) AS TotalCredits
3  FROM S058
4  LEFT JOIN SC058 ON S058.Sno = SC058.Sno
5  LEFT JOIN C058 ON SC058.Cno = C058.Cno
6  WHERE S058.Sno IN (SELECT S058.Sno
7                      FROM S058
8                      LEFT JOIN SC058 ON S058.Sno = SC058.Sno
9                      LEFT JOIN C058 ON SC058.Cno = C058.Cno
10                     WHERE C058.Cno LIKE 'CS-%'
11                     GROUP BY S058.Sno
12                     HAVING COUNT(distinct C058.Cno) = (SELECT
13                                                         COUNT(C058.Cno)
14                                                         FROM C058
15                                                         WHERE C058.Cno LIKE
16                                                         'CS-%'))
15  GROUP BY S058.Sno;

```

最下层的子查询用于计算计算机专业的课程数量，然后在外层查询中使用HAVING子句来筛选选修了全部计算机专业课程的学生。最后使用SUM(C058.Credit)来计算学生的学分总数。

本题值得注意的点：

- 在检查学生是否选修了全部计算机专业课程时，应该使用distinct关键字来保证不会重复计算同一门课程。
- 在计算获得的学分总数时，应该注意只有60分以上的课程才会计入学分总数。

```

mydb.public> INSERT INTO C058 VALUES ('CS-03', '离散数学', 64, 4, '陈建明')
[2024-06-11 21:09:35] 5 ms 中有 1 行受到影响

```

8. 查询选修了 3 门以上课程（包括 3 门）的学生中平均成绩最高的同学学号及姓名。

```

1  --查询选修了 3 门以上课程（包括 3 门）的学生中平均成绩最高的同学学号及姓名。
2  SELECT SC058.Sno, S058.Sname
3  FROM SC058
4  LEFT JOIN S058 ON SC058.Sno = S058.Sno
5  GROUP BY SC058.Sno, S058.Sname
6  HAVING COUNT(SC058.Cno) ≥ 3
7  ORDER BY AVG(SC058.Grade) DESC
8  LIMIT 1;

```

	sno	sname
1	01032112	董蔚

第二小题

1. 分别在 S×××和 C×××表中加入记录('01032005', '刘亮', '男', '2003-12-10', 1.75, '东 14 舍 312')及('CS-03', "离散数学", 64, 4, '陈建明')

1. 在S×××表中加入记录('01032005', '刘亮', '男', '2003-12-10', 1.75, '东 14 舍 312')

```

1  INSERT INTO S058 VALUES ('01032005', '刘亮', '男', '2003-12-10', 1.75, '东 14 舍 312');

```

```

mydb.public> INSERT INTO S058 VALUES ('01032005', '刘亮', '男', '2003-12-10', 1.75, '东 14 舍 312')
[2024-06-11 21:08:47] [23505] [127.0.0.1:5731/127.0.0.1:7654] ERROR: duplicate key value violates unique constraint "s058_pkey"
[2024-06-11 21:08:47] 详细: Key (sno)=(01032005) already exists.

```

因为我们在创建表格时将Sno设置为主键，所以如果插入重复的学号会导致插入失败。

2. 在C×××表中加入记录('CS-03', "离散数学", 64, 4, '陈建明')

```

1  INSERT INTO C058 VALUES ('CS-03', '离散数学', 64, 4, '陈建明');

```

CS-03作为主键没有重复，所以插入成功。

```

mydb.public> INSERT INTO C058 VALUES ('CS-03', '离散数学', 64, 4, '陈建明')
[2024-06-11 21:09:35] 5 ms 中有 1 行受到影响

```

第三小题

1. 将 S×××表中已修分数大于 60 的学生记录删除。

```

1  DELETE FROM S058
2  WHERE S058.Sno IN (
3      SELECT S058.Sno
4      FROM S058
5      LEFT JOIN SC058 ON S058.Sno = SC058.Sno
6      LEFT JOIN C058 ON SC058.Cno = C058.Cno
7      GROUP BY S058.Sno
8      HAVING COALESCE(SUM(C058.Credit), 0) > 60
9  );

```

```
[2024-06-11 21:09:35] 5 ms 中有 1 行受到影响
mydb.public> DELETE FROM S058
        WHERE S058.Sno IN (
            SELECT S058.Sno
            FROM S058
            LEFT JOIN SC058 ON S058.Sno = SC058.Sno
            LEFT JOIN C058 ON SC058.Cno = C058.Cno
            GROUP BY S058.Sno
            HAVING COALESCE(SUM(C058.Credit), 0) > 60
        )
[2024-06-11 21:10:31] 在 4 ms 内完成
```

因为此时没有学生的学分总数大于60，所以删除操作不会影响任何记录。

第四小题

1. 将“张明”老师负责的“信号与系统”课程的学时数调整为 64，同时增加一个学分。

```
1 | UPDATE C058
2 | SET Period = 64, Credit = Credit + 1
3 | WHERE C058.Cname = '信号与系统' AND C058.Teacher = '张明';
```

```
mydb.public> UPDATE C058
        SET Period = 64, Credit = Credit + 1
        WHERE C058.Cname = '信号与系统' AND C058.Teacher = '张明'
[2024-06-11 21:11:00] 3 ms 中有 1 行受到影响
```

查询此课程检查，确认修改正确

```
1 | SELECT * FROM C058 WHERE Cname = '信号与系统';
```

	cno	cname	period	credit	teacher
1	EE-01	信号与系统	64	4.0	张明

第五小题

建立如下视图：

1. 居住在“东 18 舍”的男生视图，包括学号、姓名、出生日期、身高等属性。

```
1 | CREATE VIEW BOYS_IN_dom18 AS
2 | SELECT Sno, Sname, Bdate, Height
3 | FROM S058
4 | WHERE Dorm Like '东18舍%' AND Sex = '男';
```

	sno	sname	bdate	height
1	03031014	赵思扬	2002-06-06 00:00:00	1.85
2	03031051	周剑	2002-05-08 00:00:00	1.68
3	03031033	蔡明明	2003-03-12 00:00:00	1.75

2. “张明”老师所开设课程情况的视图，包括课程编号、课程名称、平均成绩等属性。

```

1 CREATE VIEW COURSES_BY_ZHANGMING AS
2 SELECT C058.Cno, Cname, AVG(SC058.Grade) AS AvgGrade
3 FROM C058
4 LEFT JOIN SC058 ON C058.Cno = SC058.Cno
5 WHERE C058.Teacher = '张明'
6 GROUP BY C058.Cno, Cname
7 ORDER BY C058.Cno;

```

	cno	cname	avggrade
1	EE-01	信号与系统	85.8

3. 所有选修了“人工智能”课程的学生视图，包括学号、姓名、成绩等属性。

```

1 CREATE VIEW STUDENTS_TAKING_AI AS
2 SELECT S058.Sno, Sname, SC058.Grade
3 FROM S058
4 LEFT JOIN SC058 ON S058.Sno = SC058.Sno
5 LEFT JOIN C058 ON SC058.Cno = C058.Cno
6 WHERE C058.Cname = '人工智能';

```

	sno	sname	grade
1	01032010	王涛	83.5
2	01032001	张晓梅	83.0
3	01032005	刘静	82.0
4	01032023	孙文	76.0
5	01032112	董蔚	86.0

第四题

1. Student数据来源：西安交通大学计算机学院官网：2023年电信学部计算机科学与技术学院夏令营入营名单

2023年电信学部计算机科学与技术学院夏令营入营名单

来源： 日期：2023-06-21 编辑人：

现将2023年计算机科学与技术学院夏令营入营名单公示如下，请入营营员6月25日前在报名系统确认是否参加我院夏令营。确认截止后请入营同学注意查收邮件通知加入计算机学院夏令营微信群，以便后续相关事项通知。

联系人：王老师

电子邮件：pwang1982@mail.xjtu.edu.cn

联系电话：029-82668971

序号	报名号	姓名
1	20232200025	汪永嘉
2	20232200142	邓凯文
3	20232200285	陈瑾晗
4	20232200704	胡新宇
5	20232200724	崔正奇
6	20232200798	王家宇

1. 使用requests库发送HTTP GET请求到指定URL，获取HTML内容。
2. 使用正则表达式匹配学生名单。
3. 将学生名单写入文件。

西安交通大学

XI'AN JIAOTONG DAXUE

全校课表查询

学生组

全校课程安排查询

2023 - 2024学年 第二学期 更改

课程 时间

最近校公选课
请选择...

课程号

课程名

开课单位
请选择...

上课教师

学校校区
请选择...

校公选课类别
请选择...

结束节次
请选择...

上课周次

上课星期

搜索

清空条件 | 更多条件 收起 >

自定义列

操作	课程号	课程名	课序号	开课单位	学时	学分	上课教师	选课人数	上课班级	已排时间地点	学院
查看详情	GNED110400	大学生心理健康...	01	学工部/学生处/武装部	32	2	刘可娣,郑斌	198		1-2周,5-7周 星期一 第9节-第10节 主...	兴庆校区
查看详情	GNED110400	大学生心理健康...	03	学工部/学生处/武装部	32	2	李红高,王婷	93		1-7周 星期二 第9节-第10节 主楼C-1...	兴庆校区
查看详情	GNED110400	大学生心理健康...	05	学工部/学生处/武装部	32	2	阎昭,李美婷	198		1-2周,6周 星期一 第9节-第10节 主楼...	兴庆校区
查看详情	GNED110400	大学生心理健康...	06	学工部/学生处/武装部	32	2	董研林,王婷	174		1-8周 星期一 第9节-第10节 主楼D-1...	兴庆校区
查看详情	GNED110400	大学生心理健康...	07	学工部/学生处/武装部	32	2	赵颖,武如云	199		1-2周,6-8周 星期三 第9节-第10节 主...	兴庆校区
查看详情	SOCPI00190	课外实践B学分	01	学工部/学生处/武装部	0	8	任群,温景涛,梁浩,魏波	673			兴庆校区
查看详情	SOCPI00190	课外实践B学分	02	学工部/学生处/武装部	0	8	刘旭凤,苏钰豪,魏波	519			兴庆校区
查看详情	SOCPI00190	课外实践B学分	03	学工部/学生处/武装部	0	8	席博,李斌,闫飞,魏波	702			兴庆校区
查看详情	SOCPI00190	课外实践B学分	04	学工部/学生处/武装部	0	8	孙丹,魏波	250			兴庆校区
查看详情	SOCPI00190	课外实践B学分	05	学工部/学生处/武装部	0	8	杨旭阳,魏波	771			兴庆校区

1. 使用Selenium库打开本科教务平台。
2. 输入学号和密码登录。
3. 点击全校课表查询。
4. 获取课程信息。
5. 将课程信息写入文件。
6. 数据清洗：删除同名课程，只保留第一次出现的课程。

1. 完整程序见附件2，下面的代码片段仅展示部分代码作为示例。

1. 读取student_list.txt文件，将学生数据存储在2D数组中。
2. 使用JDBC连接数据库。
3. 使用预编译语句插入学生数据。
4. 执行批量插入。
5. 关闭连接。

```
1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4 import java.sql.*;
5 import java.util.ArrayList;
6 import java.util.Collections;
7 import java.util.List;
8
9 public class Database {
10     public static List<String[]> readTxtTo2DArray(String filePath)
11     throws IOException {
```

```

12 // 插入学生数据,根据len判断插入的数据量
13 private static void insertStudentData(Connection conn,
List<String[]> studentData) throws SQLException {
14 // SQL插入语句模板
15 String insertStudent = "INSERT INTO S058 (Sno, Sname, Sex,
Bdate, Height,Dorm) VALUES (?, ?, ?, ?, ?, ?)";
16 // 调用Connection的prepareStatement方法创建预编译语句对象。
17 try (PreparedStatement pstmt =
conn.prepareStatement(insertStudent)) {
18 for (String[] row : studentData) {
19 // 设置参数 set+数据类型(参数位置, 参数值)
20 pstmt.setString(1, row[0]); // Sno
21 pstmt.setString(2, row[1]); // Sname
22 pstmt.setString(3, row[2]); // Sex
23 pstmt.setString(4, row[3]); // Bdate
24 pstmt.setDouble(5, Double.parseDouble(row[4])); //
Height
25 pstmt.setString(6, row[5]); // Dorm
26 // 将预编译语句添加到批处理命令中
27 pstmt.addBatch();
28 }
29 pstmt.executeBatch(); // 执行批量插入
30 }
31 }
32
33 // 插入课程数据
34 //解析同上
35 private static void insertCourseData(Connection conn,
List<String[]> courseData) throws SQLException {
36 }
37
38 // 插入学生课程数据
39 private static void insertStudentCourseData(Connection conn,
List<String[]> studentCourseData) throws
40 }
41
42 //随机删除SC表中成绩低于60分的项: 筛选出成绩低于60分的学生, 然后随机排序, 删除
前200个, 主键为sno+cno
43 public static void deleteSCData(Connection conn) throws
SQLException {
44 }
45
46 //从user.txt中读取url,username,password
47 public static String[] readUser() throws IOException {
48 }
49
50 public static void main(String[] args) {
51 } catch (IOException e) {
52 System.err.println("Error reading file: " +
e.getMessage());
53 } catch (SQLException | InterruptedException e) {
54 System.err.println("Error: " + e.getMessage());
55 }
56 }

```

```
57 |  
58 |}  
59 |
```

插入结果使用下面的SQL语句查询：

```
1 | SELECT COUNT(*) FROM S058;  
2 | SELECT COUNT(*) FROM C058;  
3 | SELECT COUNT(*) FROM SC058;
```

结果依次为：

	count
1	5070

	count
1	1586

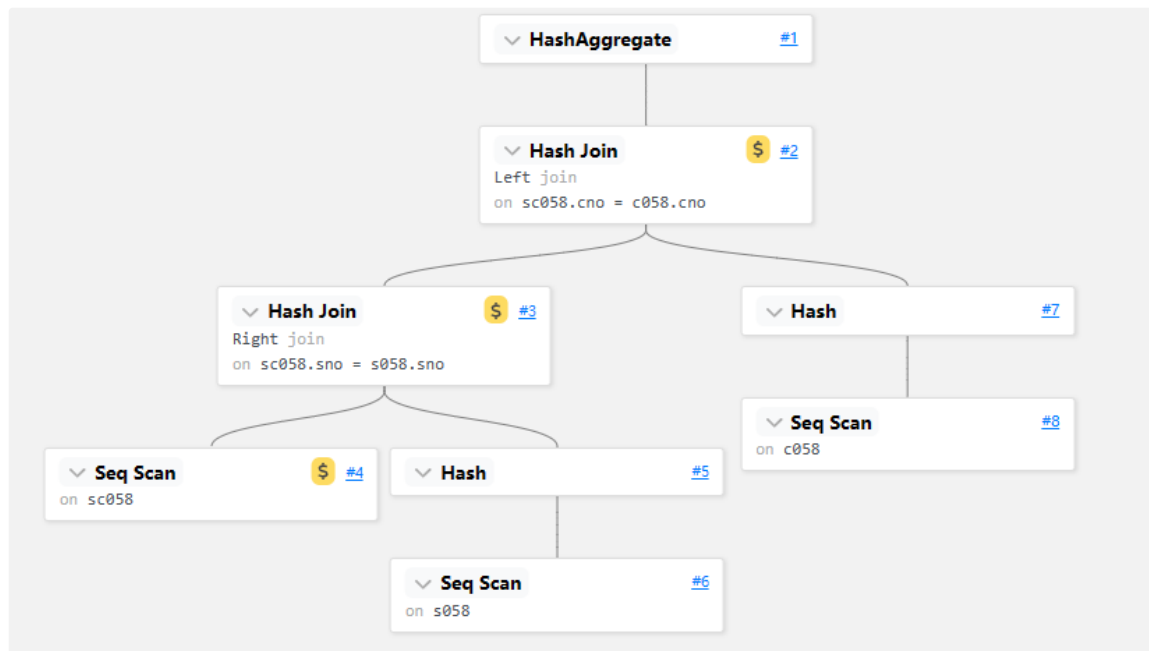
	count
1	253527

重写前面的查询

1. 查询每位学生的学号、学生姓名及其已选修课程的学分总数。

```
1 | EXPLAIN SELECT S058.Sno, S058.Sname, COALESCE(SUM(C058.Credit), 0) AS  
  | TotalCredits  
2 | FROM S058  
3 | LEFT JOIN SC058 ON S058.Sno = SC058.Sno -- 左连接  
4 | LEFT JOIN C058 ON SC058.Cno = C058.Cno -- 左连接  
5 | GROUP BY S058.Sno;
```

1	HashAggregate (cost=12619.66..12670.36 rows=5070 width=54)
2	Group By Key: s058.sno
3	-> Hash Left Join (cost=218.76..11352.02 rows=253527 width=22)
4	Hash Cond: (sc058.cno = c058.cno)
5	-> Hash Right Join (cost=167.07..7814.34 rows=253527 width=24)
6	Hash Cond: (sc058.sno = s058.sno)
7	-> Seq Scan on sc058 (cost=0.00..4161.27 rows=253527 width=16)
8	-> Hash (cost=103.70..103.70 rows=5070 width=17)
9	-> Seq Scan on s058 (cost=0.00..103.70 rows=5070 width=17)
10	-> Hash (cost=31.86..31.86 rows=1586 width=12)
11	-> Seq Scan on c058 (cost=0.00..31.86 rows=1586 width=12)

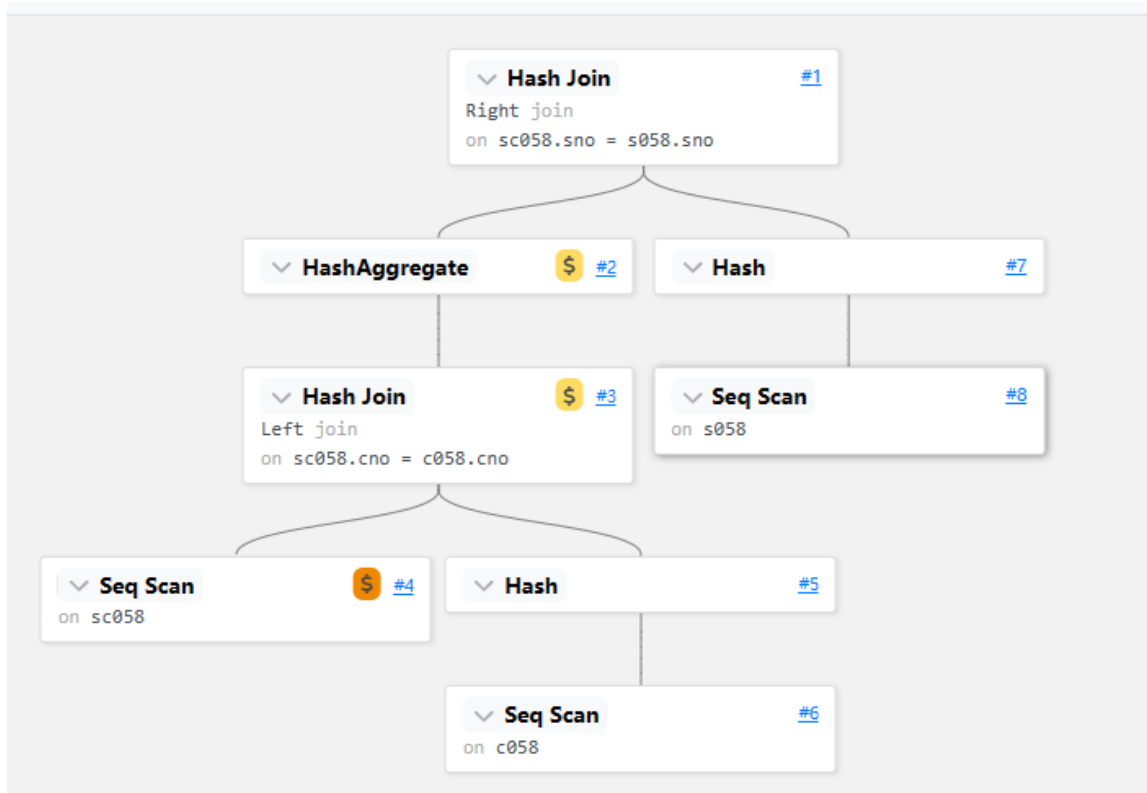


可以看到，开销最大的是HashAggregate，其次是Hash Left Join，最小的是Seq Scan on c058。原因是HashAggregate需要对结果进行分组，而Hash Left Join需要对两个表进行连接。

我们可以先计算学生的学分总数，然后再与学生表进行连接，这样可以减少连接的次数。

```
1 | SELECT S058.Sno, S058.Sname, COALESCE(TotalCredits, 0) AS TotalCredits
2 | FROM S058
3 | LEFT JOIN (
4 |     SELECT SC058.Sno, SUM(C058.Credit) AS TotalCredits
5 |     FROM SC058
6 |     LEFT JOIN C058 ON SC058.Cno = C058.Cno
7 |     GROUP BY SC058.Sno
8 | ) AS SubQuery ON S058.Sno = SubQuery.Sno;
```


	QUERY PLAN
1	Hash Right Join (cost=9133.66..9293.67 rows=5070 width=49)
2	Hash Cond: (sc058.sno = s058.sno)
3	-> HashAggregate (cost=8966.59..9014.00 rows=4741 width=46)
4	Group By Key: sc058.sno
5	-> Hash Left Join (cost=51.69..7698.95 rows=253527 width=14)
6	Hash Cond: (sc058.cno = c058.cno)
7	-> Seq Scan on sc058 (cost=0.00..4161.27 rows=253527 width=16)
8	-> Hash (cost=31.86..31.86 rows=1586 width=12)
9	-> Seq Scan on c058 (cost=0.00..31.86 rows=1586 width=12)
10	-> Hash (cost=103.70..103.70 rows=5070 width=17)
11	-> Seq Scan on s058 (cost=0.00..103.70 rows=5070 width=17)



```

mydb.public> SELECT S058.Sno, S058.Sname, COALESCE(SUM(C058.Credit), 0) AS TotalCredits
FROM S058
LEFT JOIN SC058 ON S058.Sno = SC058.Sno -- 左连接
LEFT JOIN C058 ON SC058.Cno = C058.Cno -- 左连接
GROUP BY S058.Sno
[2024-06-11 22:38:17] 在 238 ms (execution: 202 ms, fetching: 36 ms) 内检索到从 1 开始的 500 行
mydb.public> SELECT S058.Sno, S058.Sname, COALESCE(TotalCredits, 0) AS TotalCredits
FROM S058
LEFT JOIN (
    SELECT SC058.Sno, SUM(C058.Credit) AS TotalCredits
    FROM SC058
    LEFT JOIN C058 ON SC058.Cno = C058.Cno
    GROUP BY SC058.Sno
) AS SubQuery ON S058.Sno = SubQuery.Sno
[2024-06-11 22:38:17] 在 195 ms (execution: 165 ms, fetching: 30 ms) 内检索到从 1 开始的 500 行

```

可以看到，使用子查询的方法比原来的方法更快。

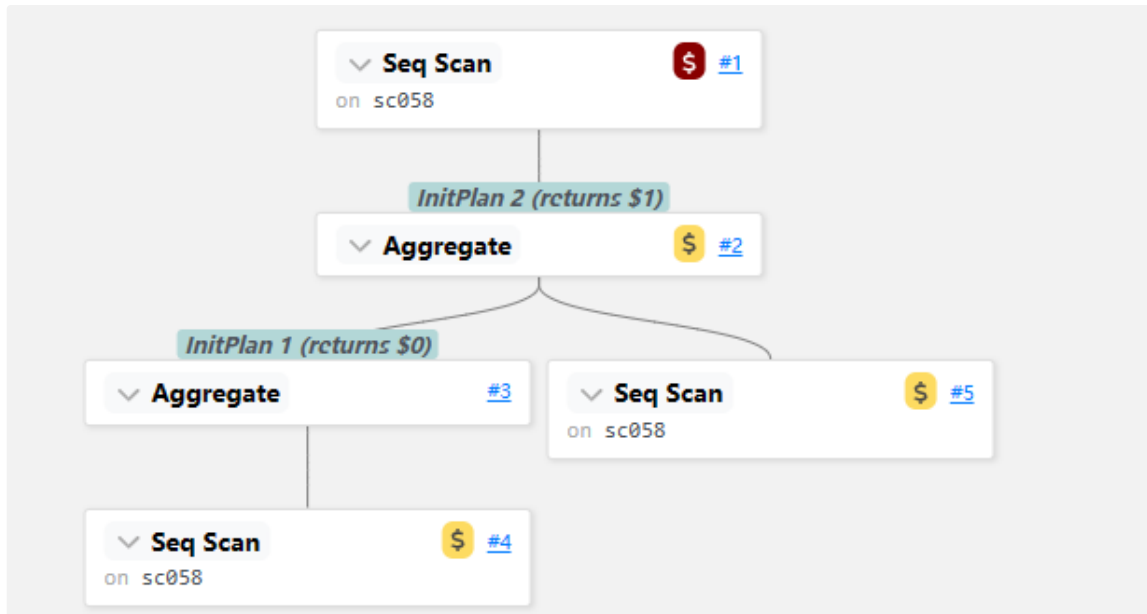
2. 查询选修课程“CS-01”的学生中成绩第二高的学生学号。

```

1 EXPLAIN SELECT SC058.Sno
2 FROM SC058
3 WHERE SC058.Cno = 'CS-01' AND SC058.Grade = (
4     SELECT MAX(SC058.Grade)
5     FROM SC058
6     WHERE SC058.Cno = 'CS-01' AND SC058.Grade < (
7         SELECT MAX(SC058.Grade)
8         FROM SC058
9         WHERE SC058.Cno = 'CS-01'
10    )
11 );

```

QUERY PLAN
1 Seq Scan on sc058 (cost=10224.53..15653.43 rows=1 width=9)
2 Filter: ((cno = 'CS-08'::bpchar) AND (grade = \$1))
3 InitPlan 2 (returns \$1)
4 -> Aggregate (cost=10224.52..10224.53 rows=1 width=38)
5 InitPlan 1 (returns \$0)
6 -> Aggregate (cost=4795.47..4795.48 rows=1 width=38)
7 -> Seq Scan on sc058 (cost=0.00..4795.09 rows=155 width=6)
8 Filter: (cno = 'CS-08'::bpchar)
9 -> Seq Scan on sc058 (cost=0.00..5428.90 rows=52 width=6)
10 Filter: ((grade < \$0) AND (cno = 'CS-08'::bpchar))



可以看到，最耗时的是Seq Scan on sc058，原因是需要对表进行扫描。

下面给出一个优化的方法：使用窗口函数。此窗口函数的作用是对成绩进行排序，然后找出第二高的成绩。同分数的学生会被视为同一名次。

这个优化的查询将计算选修课程数量和平均成绩的工作放在CTE中完成，然后在主查询中进行连接和排序，从而提高查询的效率和可读性。

```

1 WITH RankedGrades AS (
2   SELECT SC058.Sno, SC058.Cno, SC058.Grade,
3         ROW_NUMBER() OVER (PARTITION BY SC058.Cno ORDER BY SC058.Grade
4                             DESC) AS Rank
5   FROM SC058
6   WHERE SC058.Cno = 'CS-08'
7 )
8 SELECT RankedGrades.Sno
9 FROM RankedGrades
10 WHERE RankedGrades.Cno = 'CS-08' AND RankedGrades.Rank = 2;

```

```

mydb.public> SELECT SC058.Sno
FROM SC058
WHERE SC058.Cno = 'CS-08' AND SC058.Grade = (
  SELECT MAX(SC058.Grade)
  FROM SC058
  WHERE SC058.Cno = 'CS-08' AND SC058.Grade < (
    SELECT MAX(SC058.Grade)
    FROM SC058
    WHERE SC058.Cno = 'CS-08'
  )
)
[2024-06-11 23:03:36] 在 153 ms (execution: 123 ms, fetching: 30 ms) 内检索到从 1 开始的 1 行
mydb.public> WITH RankedGrades AS (
  SELECT SC058.Sno, SC058.Cno, SC058.Grade,
        ROW_NUMBER() OVER (PARTITION BY SC058.Cno ORDER BY SC058.Grade DESC) AS Rank
  FROM SC058
  WHERE SC058.Cno = 'CS-08'
)
SELECT RankedGrades.Sno
FROM RankedGrades
WHERE RankedGrades.Cno = 'CS-08' AND RankedGrades.Rank = 2
[2024-06-11 23:03:36] 在 69 ms (execution: 39 ms, fetching: 30 ms) 内检索到从 1 开始的 1 行

```

可以看到，使用窗口函数的方法比原来的方法更快。

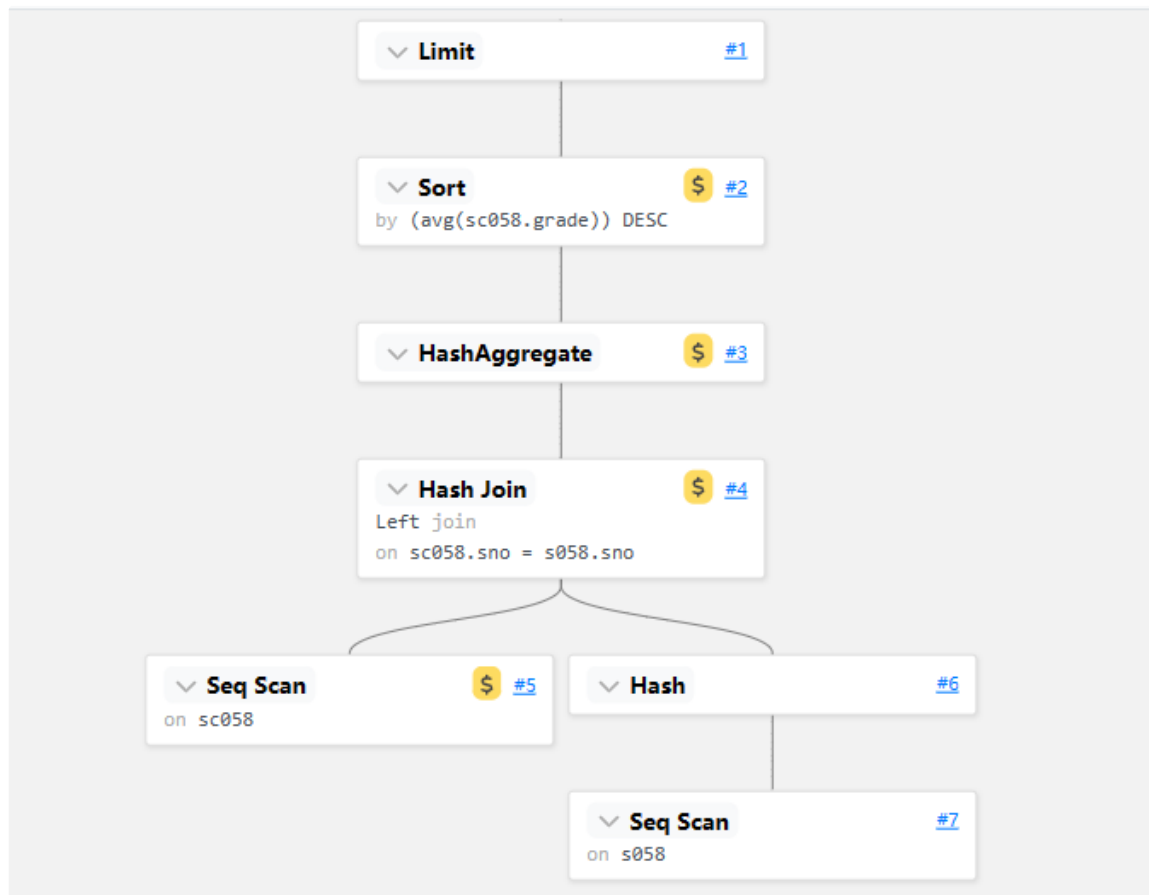
3. 查询选修了 3 门以上课程（包括 3 门）的学生中平均成绩最高的同学学号及姓名。

```

1 SELECT SC058.Sno, S058.Sname
2 FROM SC058
3 LEFT JOIN S058 ON SC058.Sno = S058.Sno
4 GROUP BY SC058.Sno, S058.Sname
5 HAVING COUNT(SC058.Cno) ≥ 3
6 ORDER BY AVG(SC058.Grade) DESC
7 LIMIT 1;

```

1	Limit (cost=15420.15..15420.15 rows=1 width=70)
2	-> Sort (cost=15420.15..16053.97 rows=253527 width=70)
3	Sort Key: (avg(sc058.grade)) DESC
4	-> HashAggregate (cost=10349.61..14152.52 rows=253527 width=70)
5	Group By Key: sc058.sno, s058.sname
6	Filter: (count(sc058.cno) >= 3)
7	-> Hash Left Join (cost=167.07..7814.34 rows=253527 width=30)
8	Hash Cond: (sc058.sno = s058.sno)
9	-> Seq Scan on sc058 (cost=0.00..4161.27 rows=253527 width=22)
10	-> Hash (cost=103.70..103.70 rows=5070 width=17)
11	-> Seq Scan on s058 (cost=0.00..103.70 rows=5070 width=17)



为了优化这个查询，我们可以采取以下措施：

减少连接次数：首先计算符合条件的学生，再进行连接，以避免不必要的连接操作。

使用公用表表达式（CTE）：将计算部分分离出来，使主查询更简洁。

```

1 WITH StudentCourseStats AS (
2     SELECT SC058.Sno, COUNT(SC058.Cno) AS CourseCount, AVG(SC058.Grade)
3     AS AvgGrade
4     FROM SC058
5     GROUP BY SC058.Sno
6     HAVING COUNT(SC058.Cno) ≥ 3
7 )
8 SELECT S058.Sno, S058.Sname
9 FROM StudentCourseStats
10 JOIN S058 ON StudentCourseStats.Sno = S058.Sno
11 ORDER BY StudentCourseStats.AvgGrade DESC
12 LIMIT 1;

```

```

mydb.public> SELECT SC058.Sno, S058.Sname
               FROM SC058
               LEFT JOIN S058 ON SC058.Sno = S058.Sno
               GROUP BY SC058.Sno, S058.Sname
               HAVING COUNT(SC058.Cno) >= 3
               ORDER BY AVG(SC058.Grade) DESC
               LIMIT 1
[2024-06-11 23:22:45] 在 266 ms (execution: 242 ms, fetching: 24 ms) 内检索到从 1 开始的 1 行
mydb.public> WITH StudentCourseStats AS (
               SELECT SC058.Sno, COUNT(SC058.Cno) AS CourseCount, AVG(SC058.Grade) AS AvgGrade
               FROM SC058
               GROUP BY SC058.Sno
               HAVING COUNT(SC058.Cno) >= 3
               )
               SELECT S058.Sno, S058.Sname
               FROM StudentCourseStats
               JOIN S058 ON StudentCourseStats.Sno = S058.Sno
               ORDER BY StudentCourseStats.AvgGrade DESC
               LIMIT 1
[2024-06-11 23:22:45] 在 194 ms (execution: 168 ms, fetching: 26 ms) 内检索到从 1 开始的 1 行

```

可以看到，使用CTE的方法比原来的方法更快。

总结

1. 通过本次实验，我学会了如何使用OpenGauss数据库，以及如何使用SQL语言进行数据库操作。
2. 通过本次实验，我学会了如何使用JDBC连接数据库，以及如何使用Java程序进行数据库操作。
3. 通过本次实验，我学会了如何使用Selenium库进行网页爬虫，以及如何使用正则表达式进行数据提取。
4. 通过本次实验，我学会了如何使用使用JAVA进行多线程编程，以及如何使用多线程提高程序的效率。

附件

1. 第四题使用的爬虫程序：

```

1  import requests
2  import re
3  from selenium import webdriver
4  from selenium.webdriver.common.by import By
5  import time
6  import os
7
8  def get_html(url):
9      # 发送HTTP GET请求到指定URL
10     response = requests.get(url)
11     # 检查请求是否成功
12     if response.status_code == 200:
13         return response.content.decode('utf-8')
14     else:
15         print(f"请求失败, 状态码: {response.status_code}")
16         return None
17
18  def get_stu_list():
19     # 数据来源: XJTU计算机学院夏令营优营班学生名单
20     # 23年
21     url = "http://www.cs.xjtu.edu.cn/info/1233/3149.htm"
22     html = get_html(url)

```

```

23     if html is None:
24         return None
25     content = re.findall(r'<span style="font-size:12.0pt;font-family:
宋体;color:black">(.*?)</span></p></td>', html)
26     # 22年
27     url1 = "http://www.cs.xjtu.edu.cn/info/1233/2899.htm"
28     html1 = get_html(url1)
29     content += re.findall(r'<span style="font-size:12.0pt;font-family:
宋体;color:black">(.*?)</span></p></td>', html1)
30     # 20年
31     url2 = "http://www.cs.xjtu.edu.cn/info/1233/2469.htm"
32     html2 = get_html(url2)
33     content += re.findall(
34         r'<td width="105" nowrap height="27" style="border-top: none;
border-left: none; border-bottom: 1px solid '
35         r'black; border-right: 1px solid black; padding: 1px 1px
0px;"><p '
36         r'style="text-align:center;vertical-align:middle"><span
style="font-size:16px;font-family:宋体;color:black">('
37         r'.*?)</span></p></td>',
38         html2)
39     url3 = "http://www.cs.xjtu.edu.cn/info/1233/2466.htm"
40     html3 = get_html(url3)
41     content += re.findall(
42         r'<td nowrap height="33" style="border-top: none; border-left:
none; border-bottom: 1px solid black; '
43         r'border-right: 1px solid black; padding: 1px 1px 0px;"><p '
44         r'style="text-align:center;vertical-align:middle"><span
style="font-size:16px;font-family:宋体;color:black">('
45         r'.*?)</span></p></td>',
46         html3)
47     # 删除三个表头
48     content = content[3:]
49     # 将content写入文件, 如果文件存在则覆盖
50     with open('student_list.txt', mode='w', encoding='utf-8') as f:
51         for student in content:
52             f.write(student + '\n')
53     return content
54
55 def get_course_list():
56     if not os.path.exists('course_crawler.txt'):
57         # 学号与密码
58         NetID = '*****'
59         password = '*****'
60
61         # 创建WebDriver对象
62         driver = webdriver.Edge()
63
64         # 打开本科教务平台
65         url = 'http://ehall.xjtu.edu.cn/new/index.html?browser=no'
66         driver.get(url)
67
68         # 点击登录
69         login_button = driver.find_element(By.ID, 'ampHasNoLogin')

```

```
70     login_button.click()
71
72     # 输入学号和密码
73     user_box = driver.find_element(By.NAME, 'username')
74     password_box = driver.find_element(By.NAME, 'pwd')
75     user_box.send_keys(NetID)
76     password_box.send_keys(password)
77
78     # 统一身份认证网关
79     login_button = driver.find_element(By.CSS_SELECTOR,
80     '.loginState > #account_login')
81     driver.execute_script('arguments[0].click();', login_button)
82
83     # 全校课表查询
84     time.sleep(3)
85     app_list = driver.find_element(By.CSS_SELECTOR, '.amp-aside-
86     box-mini-item > .icon-liebiao1')
87     driver.execute_script('arguments[0].click();', app_list)
88     course_table = driver.find_element(By.CSS_SELECTOR,
89     '.appFlag:nth-child(10) > .amp-str-cut')
90     driver.execute_script('arguments[0].click();', course_table)
91
92     # 切换页面
93     new_window = driver.window_handles[-1]
94     driver.switch_to.window(new_window)
95
96     # 获取课程信息
97     with open('course_crawler.txt', mode='w') as f:
98         for i in range(400):
99             time.sleep(3)
100             for j in range(10):
101                 cno1 = f'id("row{j}qxkcb-index-
102                 table")/TD[3]/SPAN[1]'
103                 cno2 = f'id("row{j}qxkcb-index-
104                 table")/TD[5]/SPAN[1]'
105                 cname = f'id("row{j}qxkcb-index-
106                 table")/TD[4]/SPAN[1]'
107                 college = f'id("row{j}qxkcb-index-
108                 table")/TD[6]/SPAN[1]'
109                 period = f'id("row{j}qxkcb-index-
110                 table")/TD[8]/SPAN[1]'
111                 credit = f'id("row{j}qxkcb-index-
112                 table")/TD[9]/SPAN[1]'
113                 teacher = f'id("row{j}qxkcb-index-
114                 table")/TD[10]/SPAN[1]'
115                 f.write(driver.find_element(By.XPATH,
116                 cno1).get_attribute('title') + '; ')
117                 f.write(driver.find_element(By.XPATH,
118                 cno2).get_attribute('title') + '; ')
119                 f.write(driver.find_element(By.XPATH,
120                 cname).get_attribute('title') + '; ')
121                 f.write(driver.find_element(By.XPATH,
122                 college).get_attribute('title') + '; ')
```

```

109         f.write(driver.find_element(By.XPATH,
period).get_attribute('title') + '; ')
110         f.write(driver.find_element(By.XPATH,
credit).get_attribute('title') + '; ')
111         f.write(driver.find_element(By.XPATH,
teacher).get_attribute('title') + '\n')
112         next_page = driver.find_element(By.CSS_SELECTOR,
'#pagerqxkcb-index-table .icon-keyboardarrowright')
113         driver.execute_script('arguments[0].click();',
next_page)
114
115         # 关闭浏览器
116         driver.quit()
117
118         # 数据清洗
119         with open('course_crawler.txt', mode='r') as f:
120             content = f.readlines()
121             course_list = []
122             for line in content:
123                 course = line.split('; ')
124                 course_list.append([course[2], course[4], course[5],
course[6]])
125             # 删除同名课程, 只保留第一次出现的课程
126             course_set = set()
127             course_list_clean = []
128             for course in course_list:
129                 if course[0] not in course_set:
130                     course_set.add(course[0])
131                     # 老师字段有多个时, 只保留第一个老师, 同时删除换行符
132                     course[3] = course[3].split(',')[0]
133                     course[3] = course[3].replace('\n', '')
134                     course_list_clean.append(course)
135             # 将content写入文件, 如果文件存在则覆盖
136             with open('course_list.txt', mode='w', encoding='utf-8') as f:
137                 for course in course_list_clean:
138                     f.write(course[0] + '; ' + course[1] + '; ' + course[2] +
'; ' + course[3] + '\n')
139             return course_list_clean
140
141 if __name__ == '__main__':
142
143     students = get_stu_list()
144     for i in range(10):
145         print(students[i])
146     course = get_course_list()
147     for i in range(10):
148         print(course[i])
149

```

2. 第四题使用的Java程序:

```

1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4 import java.sql.*;

```



```

5  import java.util.ArrayList;
6  import java.util.Collections;
7  import java.util.List;
8
9  public class Database {
10
11      // 读取txt文件, 按行存储为二维数组
12      public static List<String[]> readTxtTo2DArray(String filePath)
13      throws IOException {
14          //使用ArrayList来存储每行的数据
15          List<String[]> data = new ArrayList<>();
16          try (BufferedReader br = new BufferedReader(new
17      FileReader(filePath))) {
18              String line;
19              // 逐行读取
20              while ((line = br.readLine()) != null) {
21                  // 以逗号为分隔符, 将行拆分成数组
22                  String[] values = line.split(",");
23                  // 添加到二维数组
24                  data.add(values);
25              }
26          }
27          return data;
28      }
29
30      // 插入学生数据,根据len判断插入的数据量
31      private static void insertStudentData(Connection conn,
32      List<String[]> studentData) throws SQLException {
33          // SQL插入语句模板
34          String insertStudent = "INSERT INTO S058 (Sno, Sname, Sex,
35      Bdate, Height,Dorm) VALUES (?, ?, ?, ?, ?, ?)";
36          // 调用Connection的prepareStatement方法创建预编译语句对象。
37          try (PreparedStatement pstmt =
38      conn.prepareStatement(insertStudent)) {
39              for (String[] row : studentData) {
40                  // 设置参数 set+数据类型(参数位置, 参数值)
41                  pstmt.setString(1, row[0]); // Sno
42                  pstmt.setString(2, row[1]); // Sname
43                  pstmt.setString(3, row[2]); // Sex
44                  pstmt.setString(4, row[3]); // Bdate
45                  pstmt.setDouble(5, Double.parseDouble(row[4])); //
46      Height
47                  pstmt.setString(6, row[5]); // Dorm
48                  // 将预编译语句添加到批处理命令中
49                  pstmt.addBatch();
50              }
51              pstmt.executeBatch(); // 执行批量插入
52          }
53      }
54
55      // 插入课程数据
56      //解析同上
57      private static void insertCourseData(Connection conn,
58      List<String[]> courseData) throws SQLException {

```

```

52         String insertCourse = "INSERT INTO C058 (Cno, Cname, Period,
Credit, Teacher) VALUES (?, ?, ?, ?, ?)";
53         try (PreparedStatement pstmt =
conn.prepareStatement(insertCourse)) {
54             for (String[] row : courseData) {
55                 pstmt.setString(1, row[0]); // Cno
56                 pstmt.setString(2, row[1]); // CNAME
57                 pstmt.setInt(3, Integer.parseInt(row[2])); // PERIOD
58                 pstmt.setDouble(4, Double.parseDouble(row[3])); //
CREDIT
59                 pstmt.setString(5, row[4]); // TEACHER
60                 pstmt.addBatch();
61             }
62             pstmt.executeBatch(); // 执行批量插入
63         }
64     }
65
66     // 插入学生课程数据
67     private static void insertStudentCourseData(Connection conn,
List<String[]> studentCourseData) throws SQLException {
68         String insertStudentCourse = "INSERT INTO SC058 (Sno, Cno,
Grade) VALUES (?, ?, ?)";
69         try (PreparedStatement pstmt =
conn.prepareStatement(insertStudentCourse)) {
70             for (String[] row : studentCourseData) {
71                 pstmt.setString(1, row[0]); // Sno
72                 pstmt.setString(2, row[1]); // Cno
73                 pstmt.setDouble(3, Double.parseDouble(row[2])); //
Grade
74                 pstmt.addBatch();
75             }
76             pstmt.executeBatch(); // 执行批量插入
77         }
78     }
79
80     // 随机删除SC表中成绩低于60分的项: 筛选出成绩低于60分的学生, 然后随机排序, 删除
前200个, 主键为sno+cno
81     public static void deleteSCData(Connection conn) throws
SQLException {
82         String deleteSC = "DELETE FROM SC058 WHERE Sno = ? AND Cno =
?";
83         String selectSC = "SELECT Sno, Cno FROM SC058 WHERE Grade <
60";
84         try (PreparedStatement pstmt =
conn.prepareStatement(deleteSC);
85             PreparedStatement selectPstmt =
conn.prepareStatement(selectSC)) {
86             // 获取成绩低于60分的学生课程数据
87             List<String[]> scData = new ArrayList<>();
88             selectPstmt.execute();
89             try (ResultSet rs = selectPstmt.getResultSet()) {
90                 while (rs.next()) {
91                     String[] row = new String[2];
92                     row[0] = rs.getString(1);

```

```

93         row[1] = rs.getString(2);
94         scData.add(row);
95     }
96 }
97 // 随机排序
98 Collections.shuffle(scData);
99 // 删除前200个
100 for (int i = 0; i < 200; i++) {
101     pstmt.setString(1, scData.get(i)[0]);
102     pstmt.setString(2, scData.get(i)[1]);
103     pstmt.addBatch();
104 }
105 pstmt.executeBatch();
106 }
107 }
108
109 //从user.txt中读取url,username,password
110 public static String[] readUser() throws IOException {
111     String[] user = new String[3];
112     try (BufferedReader br = new BufferedReader(new
113 FileReader("./src/user.txt"))) {
114         String line;
115         int i = 0;
116         while ((line = br.readLine()) != null) {
117             user[i] = line;
118             i++;
119         }
120     }
121     return user;
122 }
123
124 public static void main(String[] args) {
125     // 数据库连接信息
126     String[] user = new String[3];
127     try {
128         user = readUser();
129     } catch (IOException e) {
130         System.err.println("Error reading user file: " +
131 e.getMessage());
132     }
133     String url = user[0]; // 数据库URL
134     String username = user[1]; // 数据库用户名
135     String password = user[2]; // 数据库密码
136
137     try (Connection conn = DriverManager.getConnection(url,
138 username, password)) {
139         // 读取数据
140         List<String[]> studentData =
141 readTxtTo2DArray("./src/S058.txt");
142         List<String[]> courseData =
143 readTxtTo2DArray("./src/C058.txt");
144         List<String[]> studentCourseData =
145 readTxtTo2DArray("./src/SC058.txt");
146

```

```

141         // 创建多线程插入
142         Thread thread1 = new Thread() → {
143             try {
144                 insertStudentData(conn, studentData.subList(0,
2000));
145             } catch (SQLException e) {
146                 System.err.println("Error inserting student data: "
+ e.getMessage());
147             }
148         });
149
150         Thread thread2 = new Thread() → {
151             try {
152                 insertStudentData(conn, studentData.subList(2000,
studentData.size()));
153             } catch (SQLException e) {
154                 System.err.println("Error inserting student data: "
+ e.getMessage());
155             }
156         });
157
158         thread1.start();
159         thread2.start();
160
161         thread1.join(); // 等待线程1结束
162         thread2.join(); // 等待线程2结束
163
164         // 插入课程数据, 双线程
165         Thread thread8 = new Thread() → {
166             try {
167                 insertCourseData(conn, courseData.subList(0,
700));
168             } catch (SQLException e) {
169                 System.err.println("Error inserting course data: "
+ e.getMessage());
170             }
171         });
172
173         Thread thread9 = new Thread() → {
174             try {
175                 insertCourseData(conn, courseData.subList(700,
courseData.size()));
176             } catch (SQLException e) {
177                 System.err.println("Error inserting course data: "
+ e.getMessage());
178             }
179         });
180
181         thread8.start();
182         thread9.start();
183
184         thread8.join();
185         thread9.join();
186
187         // 插入学生课程数据, 5个线程, 删除低于60分的学生数据, 1个线程

```

```

187         Thread thread3 = new Thread(() → {
188             try {
189                 insertStudentCourseData(conn,
studentCourseData.subList(0, 50000));
190             } catch (SQLException e) {
191                 System.err.println("Error inserting student course
data: " + e.getMessage());
192             }
193         });
194         Thread thread4 = new Thread(() → {
195             try {
196                 insertStudentCourseData(conn,
studentCourseData.subList(50000, 100000));
197             } catch (SQLException e) {
198                 System.err.println("Error inserting student course
data: " + e.getMessage());
199             }
200         });
201         Thread thread5 = new Thread(() → {
202             try {
203                 insertStudentCourseData(conn,
studentCourseData.subList(100000, 150000));
204             } catch (SQLException e) {
205                 System.err.println("Error inserting student course
data: " + e.getMessage());
206             }
207         });
208         Thread thread6 = new Thread(() → {
209             try {
210                 insertStudentCourseData(conn,
studentCourseData.subList(150000, 200000));
211             } catch (SQLException e) {
212                 System.err.println("Error inserting student course
data: " + e.getMessage());
213             }
214         });
215         Thread thread7 = new Thread(() → {
216             try {
217                 insertStudentCourseData(conn,
studentCourseData.subList(200000, studentCourseData.size()));
218             } catch (SQLException e) {
219                 System.err.println("Error inserting student course
data: " + e.getMessage());
220             }
221         });
222
223         Thread thread10 = new Thread(() → {
224             try {
225                 deleteSCData(conn);
226             } catch (SQLException e) {
227                 System.err.println("Error deleting student data: "
+ e.getMessage());
228             }
229         });

```

```
230
231         thread3.start();
232         thread4.start();
233         thread5.start();
234         thread6.start();
235         thread7.start();
236         thread10.start();
237
238         thread3.join();
239         thread4.join();
240         thread5.join();
241         thread6.join();
242         thread7.join();
243         thread10.join();
244
245         System.out.println("Data insertion complete.");
246
247         } catch (IOException e) {
248             System.err.println("Error reading file: " +
249 e.getMessage());
250         } catch (SQLException | InterruptedException e) {
251             System.err.println("Error: " + e.getMessage());
252         }
253     }
254 }
255
```