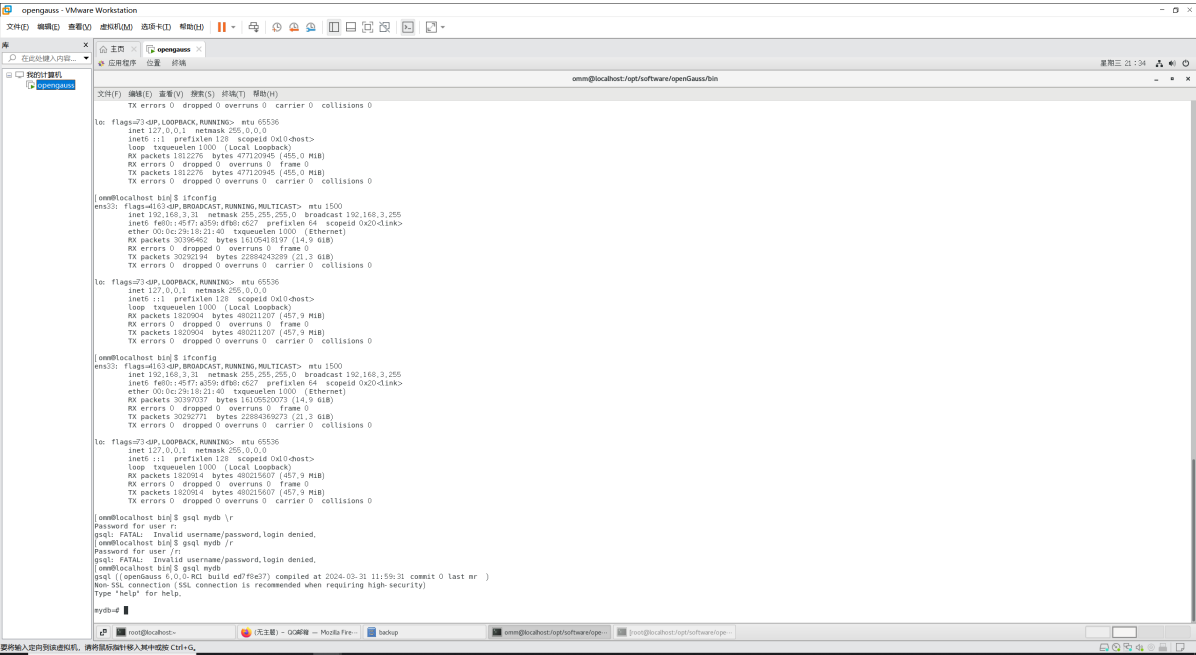


# 数据库系统实验

## 实验开始部分说明

本实验采用Opengauss6.0企业版，安装在VMWare本地虚拟机：



安装完成后进入gsql客户端：

```
[omm@localhost bin]$ gsql mydb
gsql ((openGauss 6.0.0-RC1 build ed7f8e37) compiled at 2024-03-31 11:59:31 commit 0 last mr )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

mydb=#
```

## 1、创建表和写入原始数据

### (1) 通过JDBC创建表和写入原始数据

#### (1) 代码

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Creat_table {
    public static void main(String[] args) {
        System.out.println("Hello, World!");

        // Database connection parameters
        String url = "jdbc:postgresql://192.168.3.26:5432/mydb";
        String user = "java";
        String password = "xyx2003.";

        Connection conn = null;
        try {
```

```

// Load the PostgreSQL driver (optional step in modern JDBC)
Class.forName("org.postgresql.Driver");

// Get a connection to the database
conn = DriverManager.getConnection(url, user, password);
System.out.println("Connection established successfully!");

// Create schema and tables, then insert data
createSchemaAndInsertData(conn);

} catch (SQLException e) {
    System.out.println("SQL Exception: " + e.getMessage());
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
    e.printStackTrace();
} finally {
    // Close the connection
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            System.out.println("Failed to close the connection: " +
e.getMessage());
        }
    }
}

}

// Method to create schema, tables and insert data
public static void createSchemaAndInsertData(Connection conn) throws
SQLException {
    Statement stmt = null;
    try {
        stmt = conn.createStatement();

        // Create new schema
        stmt.executeUpdate("CREATE SCHEMA IF NOT EXISTS gauss_db");

        // Drop existing tables if they exist in the new schema
        stmt.executeUpdate("DROP TABLE IF EXISTS gauss_db.SC");
        stmt.executeUpdate("DROP TABLE IF EXISTS gauss_db.S");
        stmt.executeUpdate("DROP TABLE IF EXISTS gauss_db.C");

        // Create student table in the new schema
        stmt.executeUpdate("CREATE TABLE gauss_db.S469 ("
            + "S# CHAR(10) PRIMARY KEY,"
            + "SNAME VARCHAR(50),"
            + "SEX CHAR(5),"
            + "BDATE DATE,"
            + "HEIGHT NUMERIC(4,2),"
            + "DORM CHAR(20)"
            + ")");

        // Create course table in the new schema

```

```

stmt.executeUpdate("CREATE TABLE gauss_db.C469 ("
    + "C# CHAR(10) PRIMARY KEY,"
    + "CNAME VARCHAR(100),"
    + "PERIOD SMALLINT,"
    + "CREDIT NUMERIC(3,1),"
    + "TEACHER VARCHAR(50)"
    + ")");

// Create selection table in the new schema
stmt.executeUpdate("CREATE TABLE gauss_db.SC469 ("
    + "S# CHAR(10),"
    + "C# CHAR(10),"
    + "GRADE NUMERIC(4,1),"
    + "PRIMARY KEY (S#, C#),"
    + "FOREIGN KEY (S#) REFERENCES gauss_db.S(S#),"
    + "FOREIGN KEY (C#) REFERENCES gauss_db.C(C#)"
    + ")");

// Insert data into student table
stmt.executeUpdate("INSERT INTO gauss_db.S (S#, SNAME, SEX, BDATE,
HEIGHT, DORM) 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')");

// Insert data into course table
stmt.executeUpdate("INSERT INTO gauss_db.C (C#, CNAME, PERIOD,
CREDIT, TEACHER) 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')");

// Insert data into selection table

```

```

stmt.executeUpdate("INSERT INTO gauss_db.SC (S#, C#, GRADE) 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)");

System.out.println("Tables created and data inserted successfully!");
ResultSet resultSet = conn.getMetaData().getTables(null, "gauss_db",
"s", null);
if (resultSet.next()) {
    System.out.println("Table S exists in gauss_db schema.");
} else {
    System.out.println("Table S does not exist in gauss_db schema.");
}

} finally {
    if (stmt != null) {
        stmt.close();
    }
}
}
}

```

通过JDBC中的executeUpdate实现创建对应表和把对应代码写入表中。

## (2) 数据类型分析

创建的表的各项数据类型：

```

stmt.executeUpdate("CREATE TABLE gauss_db.S ("
    + "S# CHAR(10) PRIMARY KEY,"

```

```

+ "SNAME VARCHAR(50),"
+ "SEX CHAR(5),"
+ "BDATE DATE,"
+ "HEIGHT NUMERIC(4,2),"
+ "DORM CHAR(20)"
+ ")");

// Create course table in the new schema
stmt.executeUpdate("CREATE TABLE gauss_db.C ("
+ "C# CHAR(10) PRIMARY KEY,"
+ "CNAME VARCHAR(100),"
+ "PERIOD SMALLINT,"
+ "CREDIT NUMERIC(3,1),"
+ "TEACHER VARCHAR(50)"
+ ")");

// Create selection table in the new schema
stmt.executeUpdate("CREATE TABLE gauss_db.SC ("
+ "S# CHAR(10),"
+ "C# CHAR(10),"
+ "GRADE NUMERIC(4,1),"
+ "PRIMARY KEY (S#, C#),"
+ "FOREIGN KEY (S#) REFERENCES gauss_db.S(S#),"
+ "FOREIGN KEY (C#) REFERENCES gauss_db.C(C#)"
+ ")");

```

## (2) 通过Navicat查看数据类型和录入数据

### s469表 (Student)

名	类型	长度	小数点	不是 null	键	注释
► s#	char	10	0	<input checked="" type="checkbox"/>	1	
sname	varchar	50	0	<input type="checkbox"/>		
sex	char	5	0	<input type="checkbox"/>		
bdate	timestamp	0	0	<input type="checkbox"/>		
height	numeric	4	2	<input type="checkbox"/>		
dorm	char	20	0	<input type="checkbox"/>		

s#	sname	sex	bdate	height	dorm
► 01032010	王涛	男	2003-04-05 00:00:00	1.72	东 6 舍 221
01032023	孙文	男	2004-06-10 00:00:00	1.80	东 6 舍 221
01032001	张晓梅	女	2004-11-17 00:00:00	1.58	东 1 舍 312
01032005	刘静	女	2003-01-10 00:00:00	1.63	东 1 舍 312
01032112	董蔚	男	2003-02-20 00:00:00	1.71	东 6 舍 221
03031011	王倩	女	2004-12-20 00:00:00	1.66	东 2 舍 104
03031014	赵思扬	男	2002-06-06 00:00:00	1.85	东 18 舍 421
03031051	周剑	男	2002-05-08 00:00:00	1.68	东 18 舍 422
03031009	田菲	女	2003-08-11 00:00:00	1.60	东 2 舍 104
03031033	蔡明明	男	2003-03-12 00:00:00	1.75	东 18 舍 423
03031056	曹子衿	女	2004-12-15 00:00:00	1.65	东 2 舍 305

c469表 (Course)

名	类型	长度	小数点	不是 null	键
c#	char	10	0	<input checked="" type="checkbox"/>	1
cname	varchar	100	0	<input type="checkbox"/>	
period	int2	16	0	<input type="checkbox"/>	
credit	numeric	3	1	<input type="checkbox"/>	
teacher	varchar	50	0	<input type="checkbox"/>	

c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-01	信号与系统	60	3.0	张明
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学与光子学	40	2.0	石韬

此处Cname定义的长度较长，是因为后续爬取的课程名较长，打到十几到几十个汉字，所以才设置为100.

SC469表 (选课记录表)

名	类型	长度	小数点	不是 null	键
s#	char	10	0	<input checked="" type="checkbox"/>	1
c#	char	10	0	<input checked="" type="checkbox"/>	2
grade	numeric	4	1	<input type="checkbox"/>	

s#	c#	grade
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

SC的外键:

字段	索引	外键	唯一键	检查	排除	规则	触发器	选项	注释	SQL 预览
名	字段	被引用的模式	被引用的表 (父)	被引用的字段	删除时	更新时	注释			
sc_c#_fkey	c#	gauss_db	c469	c#	NO ACTION	NO ACTION				
sc_s#_fkey	s#	gauss_db	s469	s#	NO ACTION	NO ACTION				

## 2、使用SQL语句进行查询和修改

### (1) SQL查询语句

((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。

```
SELECT C#, CNAME, CREDIT
FROM C469
WHERE C# LIKE 'EE-';
```

```

1 SELECT C#, CNAME, CREDIT
2 FROM C469
3 WHERE C# LIKE 'EE-%';

```

信息 摘要 结果 1

c#	cname	credit
EE-01	信号与系统	3.0
EE-02	数字逻辑电路	5.0
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。

```

SELECT S469.S#, SC469.C#, SC469.GRADE
FROM S469
JOIN SC469 ON S469.S# = SC469.S#
WHERE S469.SEX = '女' AND S469.S# NOT IN (
    SELECT S# FROM SC469 WHERE C# = 'CS-02'
);

```

```

1 SELECT S469.S#, SC469.C#, SC469.GRADE
2 FROM S469
3 JOIN SC469 ON S469.S# = SC469.S#
4 WHERE S469.SEX = '女' AND S469.S# NOT IN (
5     SELECT S# FROM SC469 WHERE C# = 'CS-02'
6 );

```

信息 摘要 结果 1

s#	c#	grade
03031011	EE-01	91.0
03031011	EE-02	86.0
03031009	EE-01	88.0
03031009	EE-02	78.5

使用EXISTS语法查询

```

SELECT S469.S#, SC469.C#, SC469.GRADE
FROM S469 JOIN SC469 ON S469.S#=SC469.S#
WHERE S469.SEX = '女' AND NOT EXISTS(
    SELECT 1
    FROM SC469 SC1
    WHERE SC1.C#='CS-02' AND SC1.S#=S469.S#
);

```



s#	c#	grade
03031011	EE-01	91.0
03031011	EE-02	86.0
03031009	EE-01	88.0
03031009	EE-02	78.5

### ((3)) 查询 2002 年 ~ 2003 年出生学生的基本信息。

```
SELECT * FROM S469
WHERE BDATE BETWEEN '2002-01-01' AND '2003-12-31';
```

```
1 SELECT * FROM S469
2 WHERE BDATE BETWEEN '2002-01-01' AND '2003-12-31';
```

信息	摘要	结果 1			
s#	sname	sex	bdate	height	dorm
01032010	王涛	男	2003-04-05 00:00:00	1.72	东 6 舍 221
01032005	刘静	女	2003-01-10 00:00:00	1.63	东 1 舍 312
01032112	董蔚	男	2003-02-20 00:00:00	1.71	东 6 舍 221
03031014	赵思扬	男	2002-06-06 00:00:00	1.85	东 18 舍 421
03031051	周剑	男	2002-05-08 00:00:00	1.68	东 18 舍 422
03031009	田菲	女	2003-08-11 00:00:00	1.60	东 2 舍 104
03031033	蔡明明	男	2003-03-12 00:00:00	1.75	东 18 舍 423

### ((4)) 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。

```
SELECT S469.S#, S469.SNAME, COALESCE(SUM(C469.CREDIT), 0) AS Totalcredit
FROM S469
LEFT JOIN SC469 ON S469.S# = SC469.S#
LEFT JOIN C469 ON SC469.C# = C469.C#
GROUP BY S469.S#, S469.SNAME;
```

2	FROM S469
3	LEFT JOIN SC469 ON S469.S# = SC469.S#
4	LEFT JOIN C469 ON SC469.C# = C469.C#
5	GROUP BY S469.S#, S469.SNAME;
6	

信息	摘要	结果 1
s#	sname	totalcredit
▶ 03031056	曹子衿	0
01032010	王涛	9.0
03031051	周剑	8.0
03031009	田菲	8.0
03031011	王倩	8.0
01032001	张晓梅	9.0
01032023	孙文	9.0
03031033	蔡明明	8.0
03031014	赵思扬	8.0
01032005	刘静	9.0
01032112	董蔚	11.0

分析：由于存在学生未选课，其获得学分为0，为此，使用COALESCE函数，返回第一个不为NULL的值，如果SUM计算为空，则返回0。

此外，使用左连接，保证没有任何选课记录的学生和选课成绩为空的学生也能被选中。

#### ((5)) 查询选修课程“CS-01”的学生中成绩第二高的学生学号。

```
SELECT S#
FROM SC469
WHERE C# = 'CS-01'
ORDER BY GRADE DESC
OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;
```

1	SELECT S#
2	FROM SC469
3	WHERE C# = 'CS-01'
4	ORDER BY GRADE DESC
5	OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;

信息	摘要	结果 1
s#		
▶ 01032010		

这样的查询语句有一个潜在的问题，即如果有多个成绩第二高的学生，理论上应该返回所有第二高学生的学号，但是这个语句只会返回一个学生的学号，为了解决这个问题，需要优化查询语句。

```

SELECT S#
FROM (
    SELECT S#, GRADE,
           DENSE_RANK() OVER (ORDER BY GRADE DESC) AS ranking
    FROM SC469
    WHERE C# = 'CS-01'
) AS ranked_students
WHERE ranking = 2;

```

```

1 SELECT S#
2 FROM (
3     SELECT S#, GRADE,
4           DENSE_RANK() OVER (ORDER BY GRADE DESC) AS ranking
5     FROM SC469
6     WHERE C# = 'CS-01'
7 ) AS ranked_students
8 WHERE ranking = 2;

```

信息 摘要 结果 1

s#

▶ 01032010

解释：

- `DENSE_RANK() OVER (ORDER BY GRADE DESC)` 将按成绩降序排列学生，并为每个学生分配一个密集的排名。
- 在外部查询中，我们选择排名为 2 的学生。这样即使有多个学生有相同的第二高成绩，也会显示所有这些学生的学号。

这个查询会返回所有选修课程 `CS-01` 并且成绩排名为第二的学生的学号。

**((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。**

```

SELECT S469.S#, S469.SNAME, AVG(SC469.GRADE) AS AvgGrade
FROM S469
JOIN SC469 ON S469.S# = SC469.S#
GROUP BY S469.S#, S469.SNAME
HAVING AVG(SC469.GRADE) > (
    SELECT AVG(GRADE)
    FROM S469
    JOIN SC469 ON S469.S# = SC469.S#
    WHERE S469.SNAME = '王涛'
)
ORDER BY S469.S# DESC;

```

```

4  GROUP BY S469.S#, S469.SNAME
5  HAVING AVG(SC469.GRADE) > (
6      SELECT AVG(GRADE)
7      FROM S469
8      JOIN SC469 ON S469.S# = SC469.S#
9      WHERE S469.SNAME = '王涛'
10 )
11 ORDER BY S469.S# DESC;
12

```

信息 摘要 结果 1

s#	sname	avggrade
03031033	蔡明明	91.0000000000000000
03031011	王倩	88.5000000000000000
01032112	董蔚	88.5000000000000000

((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。

```

WITH C_CS AS (
SELECT C469.C#
FROM C469
WHERE C469.C# LIKE 'CS-%'
)

SELECT S1.SNAME
FROM S469 S1
WHERE NOT EXISTS(
SELECT 1
FROM C_CS LEFT JOIN SC469 ON C_CS.C#=SC469.C# AND SC469.S#=S1.S#
WHERE SC469.S# IS NULL
)
GROUP BY S1.S#

```

```

7  SELECT S1.SNAME
8  FROM S469 S1
9  WHERE NOT EXISTS(
10     SELECT 1
11     FROM C_CS LEFT JOIN SC469 ON C_CS.C#=SC469.C# AND SC469.S#=S1.S#
12     WHERE SC469.S# IS NULL
13 )
14 GROUP BY S1.S#
15

```

信息 摘要 结果 1

sname
董蔚

**((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。**

```
WITH AvgGradeRank AS (  
    SELECT S469.S#, S469.SNAME,  
           AVG(SC469.GRADE) AS AvgGrade,  
           RANK() OVER (ORDER BY AVG(SC469.GRADE) DESC) AS rank  
    FROM S469  
    JOIN SC469 ON S469.S# = SC469.S#  
    GROUP BY S469.S#, S469.SNAME  
    HAVING COUNT(SC469.C#) >= 3  
)  
SELECT S#, SNAME  
FROM AvgGradeRank  
WHERE rank = 1;
```

1

WITH AvgGradeRank AS (  
2     SELECT S469.S#, S469.SNAME,  
3         AVG(SC469.GRADE) AS AvgGrade,  
4         RANK() OVER (ORDER BY AVG(SC469.GRADE) DESC) AS rank  
5     FROM S469  
6     JOIN SC469 ON S469.S# = SC469.S#  
7     GROUP BY S469.S#, S469.SNAME  
8     HAVING COUNT(SC469.C#) >= 3  
9     )  
10  SELECT S#, SNAME  
11  FROM AvgGradeRank  
12  WHERE rank = 1;

信息

摘要

结果 1

s#	sname
▶ 01032112	董蔚

**实现思路：**首先创建临时表AvgGradeRank，临时表中使用窗口函数RANK() OVER 进行排名,ORDER BY AVG(SC469.GRADE) DESC表示按平均成绩降序排名。之后再从临时表中选择rank为2的学生学号和名字，就能保证所有rank为2的学生都能被选出。

## (2) 修改表

分别在 S469和 C469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')。

插入前学生S469表：

s#	sname	sex	bdate	height	dorm
01032010	王涛	男	2003-04-05 00:00:00	1.72	东 6 舍 221
01032023	孙文	男	2004-06-10 00:00:00	1.80	东 6 舍 221
01032001	张晓梅	女	2004-11-17 00:00:00	1.58	东 1 舍 312
01032005	刘静	女	2003-01-10 00:00:00	1.63	东 1 舍 312
01032112	董蔚	男	2003-02-20 00:00:00	1.71	东 6 舍 221
03031011	王倩	女	2004-12-20 00:00:00	1.66	东 2 舍 104
03031014	赵思扬	男	2002-06-06 00:00:00	1.85	东 18 舍 421
03031051	周剑	男	2002-05-08 00:00:00	1.68	东 18 舍 422
03031009	田菲	女	2003-08-11 00:00:00	1.60	东 2 舍 104
03031033	蔡明明	男	2003-03-12 00:00:00	1.75	东 18 舍 423
03031056	曹子衿	女	2004-12-15 00:00:00	1.65	东 2 舍 305

插入前课程C469表:

c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-01	信号与系统	60	3.0	张明
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学与现代光通信	40	2.0	石韬

插入记录

插入学生记录

```
INSERT INTO S469 (S#, SNAME, SEX, BDATE, HEIGHT, DORM)
VALUES ('01032005', '刘竞', '男', '2003-12-10', 1.75, '东 14 舍 312');
```

1	INSERT INTO S469 (S#, SNAME, SEX, BDATE, HEIGHT, DORM)		
2	VALUES ('01032005', '刘竞', '男', '2003-12-10', 1.75, '东 14 舍 312');		
信息 摘要			
查询	信息	查询时间	
INSERT INTO S469 (S#, SNAME, SEX, BDATE, HEIGHT, DORM) VALUES ('01032005', '刘竞', '男', '2003-12-10', 1.75, '东 14 舍 312')	ERROR: duplicate key value violates unique constraint "s_pkey" DETAIL: Key ("s#")=(01032005 ) already exists.	0.074s	

错误的原因是已经有了学号为01032005的学生，学号作为CK不能重复。

01032005	刘静	女	2003-01-10 00:00:00	1.63	东 1 舍 312
----------	----	---	---------------------	------	-----------

插入课程记录

```
INSERT INTO C469 (c#, cname, period, credit, teacher)
VALUES ('CS-03', '离散数学', 64, 4, '陈建明');
```

```

1 INSERT INTO C469 (c#, cname, period, credit, teacher)
2 VALUES ('CS-03', '离散数学', 64, 4, '陈建明');

```

信息

摘要

```

INSERT INTO C469 (c#, cname, period, credit, teacher)
VALUES ('CS-03', '离散数学', 64, 4, '陈建明')

```

c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-01	信号与系统	60	3.0	张明
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学与光	40	2.0	石韬
CS-03	离散数学	64	4.0	陈建明

### (3) 将 S 表中已修分数大于 60 的学生记录删除。

首先确定是否有大于60学分的学生：

```

SELECT SNAME
FROM S469
JOIN SC469 ON S469.S# = SC469.S#
JOIN C469 ON SC469.C# = C469.C#
GROUP BY S469.S#
HAVING SUM(C469.CREDIT) > 60

```

```

1 SELECT SNAME
2 FROM S469
3 JOIN SC469 ON S469.S# = SC469.S#
4 JOIN C469 ON SC469.C# = C469.C#
5 GROUP BY S469.S#
6 HAVING SUM(C469.CREDIT) > 60
7 |

```

信息

摘要

结果 1

sname

(N/A)

可以看到没有大于60分记录的学生。

```
DELETE FROM S469
WHERE S# IN (
    SELECT S469.S#
    FROM S469
    JOIN SC469 ON S469.S# = SC469.S#
    JOIN C469 ON SC469.C# = C469.C#
    GROUP BY S469.S#
    HAVING SUM(C469.CREDIT) > 60
);
```

删除后再查询：

```
1 DELETE FROM S469
2 WHERE S# IN (
3     SELECT S469.S#
4     FROM S469
5     JOIN SC469 ON S469.S# = SC469.S#
6     JOIN C469 ON SC469.C# = C469.C#
7     GROUP BY S469.S#
8     HAVING SUM(C469.CREDIT) > 60
9 );
```

**(4) 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。**

调整前：

c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-01	信号与系统	60	3.0	张明
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学	40	2.0	石韬
CS-03	离散数学	64	4.0	陈建明

运行调整后

```
UPDATE C
SET PERIOD = 64, CREDIT = CREDIT + 1
WHERE CNAME = '信号与系统' AND TEACHER = '张明';
```



c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学与技术	40	2.0	石韬
CS-03	离散数学	64	4.0	陈建明
EE-01	信号与系统	64	4.0	张明

## (5) 建立视图：

(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。

```
CREATE VIEW MaleStudentsInEast18 AS
SELECT S#, SNAME, BDATE, HEIGHT
FROM S469
WHERE SEX = '男' AND DORM LIKE '东 18 舍%';
```

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



east18maleresidents

视图



opengauss



mydb



gauss\_db

OID

49371

所有者

java

可以看

定义

```
SELECT s."s#", s.sname, s.bdate, s.height
FROM gauss_db.s
WHERE s.dorm ~~ '东 18 舍%'::text AND s.sex = '男'::bpchar
```

ACL

--

注释

--

到视图创建成功

(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。

```
CREATE VIEW ZhangMingCourses AS
SELECT C469.C#, C469.CNAME, AVG(SC469.GRADE) AS AvgGrade
FROM C469
LEFT JOIN SC469 ON C469.C# = SC469.C#
WHERE C469.TEACHER = '张明'
GROUP BY C469.C#, C469.CNAME;
```

c#	cname	avggrade
EE-01	信号与系统	85.8000000000000000



zhangmingcourses

视图



opengauss



mydb



gauss\_db

OID

49376

所有者

java

定义

```
SELECT c."c#", c.cname, avg(sc.grade) AS avggrade
FROM gauss_db.c
LEFT JOIN gauss_db.sc ON c."c#" = sc."c#"
WHERE c.teacher::text = '张明'::text
GROUP BY c."c#", c.cname
```

ACL

--

注释

--

可以看到创建成功

(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。

```
CREATE VIEW AISTudents AS
SELECT S469.S#, S469.SNAME, SC469.GRADE
FROM S469
JOIN SC469 ON S469.S# = SC469.S#
WHERE SC469.C# = 'CS-04';
```

s#	sname	grade
01032023	孙文	76.0
01032001	张晓梅	83.0
01032005	刘静	82.0
01032112	董蔚	86.0
01032010	王涛	83.5



aistudents

视图

opengauss

mydb

gauss\_db

OID

49381

所有者

java

定义

```
SELECT s."s#", s.sname, sc.grade
FROM gauss_db.s
JOIN gauss_db.sc ON s."s#" = sc."s#"
WHERE sc."c#" = 'CS-04'::bpchar
```

ACL

--

注释

--

创建成功。

## 3、生成数据和爬取数据

3.1 S补充至约1000行，C补充约100行，SC补充约2000行，补充过程中删除数据。

### 3.1.1 补充S数据

#### 3.1.1.1 S数据爬取

使用python爬虫爬取数据，由于学生的学号，宿舍等信息不易爬取，学生信息中只选择爬取姓名，其他数据根据要求随机生成。爬取[西安交通大学电气工程学院复试名单](#)，网页内容如下：

准考证号	考生姓名	政治理论成绩	外语成绩	业务课1成绩	业务课2成绩	初试总成绩
106984110307091	韦铁	65	73	112	119	369
106984110407104	冯月	63	83	91	126	363
106984116207290	张为捷	63	65	109	127	364
106984116207292	王子彤	69	84	90	92	335
106984116207327	曹原齐	65	78	91	122	356
106984116207328	李欣尧	61	76	131	132	400
106984116207329	张景程	71	79	129	118	397
106984116207330	刘金亮	72	84	115	114	385
106984116207331	徐启昂	68	85	144	137	434
106984116207333	范哲伟	70	77	112	120	379
106984116207334	郑强	54	63	105	106	328
106984116207335	滕浩男	63	74	127	141	405
106984116207338	李杨	70	69	132	113	384
106984116307351	黄洁敏	77	86	119	111	393
106984116307353	蔡陆阳	71	72	138	93	374
106984121707443	刘海博	66	80	139	123	408
106984123207470	李峻峰	82	81	88	90	341
106984124607607	杨华蕊	63	72	115	116	366
106984124607609	李锋	75	83	121	130	409
106984130307660	薛建鹏	69	61	113	118	361

## 爬取代码

```
import openpyxl
from lxml import etree
# 创建一个新的excel工作簿
workbook=openpyxl.workbook()
# 获取默认的工作表
sheet=workbook.active
sheet['B1']='SNAME'
url="https://ee.xjtu.edu.cn/info/1383/12513.htm"
response=requests.get(url)
# 将html页面内容转换为xml文档对象
html=etree.HTML(response.content)
datalist=html.xpath('//tr')
for item in datalist:
    name=item.xpath('./td[2]/p/text()')
    if len(name)>0:
        sheet.append([' ',f'{name[0]}'])
workbook.save('D:\code\python\icourse\Sname.xlsx')
```

爬取后的表格：

origin_table >  Sname.xlsx	
	A
1	韦铁
2	冯月
3	张为捷
4	王子彤
5	曹原齐
6	李欣尧
7	张景程
8	刘金亮
9	徐启昂
10	范哲伟
11	郑强
12	滕浩男
13	李杨
14	黄洁敏
15	蔡陆阳
16	刘海博
17	李峻锋
18	杨华蕊
19	李铎
20	薛建鹏
21	张雷超
22	陈凯
23	岳林洋
24	曹光宇

### 3.1.1.2 完善表格随机生成其他数据

给学生分配学号，出生日期，身高，性别，宿舍号，分配时有以下几点需要注意：

- 1、学号作为CK不能重复。
- 2、女生身高应该平均低于男生身高，符合常理。身高整体应符合正态分布。
- 3、男女生不能分配到同一个宿舍，这里直接设置男女生分配到不同的宿舍楼,考虑到已有的学生信息，把1-5舍，11-15舍分配给女生，6-10舍，16-20舍分配给男生。
- 4、学生出生日期区别不能太大，此处设置为2002-01-01到2005-12-31。

```

import openpyxl
import random
from datetime import datetime, timedelta
import pandas as pd

def generate_student_id(existing_ids):
    exclude_ids = ['01032010', '01032023', '01032001', '01032005',
                   '01032112', '03031011', '03031014', '03031051',
                   '03031009', '03031033', '03031056']

    while True:
        student_id = ''.join(random.choices('0123456789', k=8))
        if student_id not in existing_ids and student_id not in exclude_ids:
            return student_id

def generate_gender():
    return random.choice(['男', '女'])

def generate_birthdate():
    start_date = datetime.strptime('2002-01-01', '%Y-%m-%d')
    end_date = datetime.strptime('2005-12-31', '%Y-%m-%d')
    random_date = start_date + timedelta(days=random.randint(0, (end_date -
start_date).days))
    return random_date.strftime('%Y-%m-%d')

def generate_height(gender):
    if gender == '男':
        return round(random.uniform(1.6, 1.9), 2)
    elif gender == '女':
        return round(random.uniform(1.55, 1.8), 2)
    else:
        return None

def generate_dorm(assigned_dorms, gender):
    while True:
        building = random.choice(['东', '西'])

        if gender == '女':
            # 女生只能分配到楼层 1-5 和 11-15
            floor = random.choice([1, 2, 3, 4, 5, 11, 12, 13, 14, 15])
        else:
            # 男生可以分配到除了 1-5 和 11-15 外的其他楼层
            floor = random.randint(6, 10) # 楼层范围从6到10

        fls = str(random.randint(1, 9)) # 确保是两位数, 如09
        room = str(random.randint(1, 40)).zfill(2) # 房间号, 确保是两位数, 如05

        room_number = f'{building} {floor} 舍 {fls}{room}'

        # 获取已分配的宿舍情况
        male_dorms = [dorm['dorm'] for dorm in assigned_dorms if dorm['gender']
== '男']
        female_dorms = [dorm['dorm'] for dorm in assigned_dorms if dorm['gender']
== '女']

```

```

# 如果只有男性宿舍或只有女性宿舍，则后续只能分配相应性别
if room_number in female_dorms and gender == '女':
    continue
elif room_number in male_dorms and gender == '男':
    continue

# 如果没有冲突的宿舍分配，则返回宿舍号和性别
break

return room_number, gender

def main():
    # 打开Excel文件
    wb =
openpyxl.load_workbook(r'D:\code\python\icourse\origin_table\Sname.xlsx')
    ws = wb.active

    existing_ids = set()
    assigned_dorms = []

    # 生成学号、性别、出生日期、身高和宿舍号并写入Excel文件
    for idx, row in enumerate(ws.iter_rows(min_row=1, max_col=1,
values_only=True), start=1):
        student_name = row[0]
        student_id = generate_student_id(existing_ids)
        gender = generate_gender()
        birthdate = generate_birthdate()
        height = generate_height(gender)
        dorm, dorm_gender = generate_dorm(assigned_dorms, gender)
        existing_ids.add(student_id)
        assigned_dorms.append({'dorm': dorm, 'gender': dorm_gender})

    # 写入Excel文件，第一列为学号，第二列为性别，依次类推
    ws.cell(row=idx, column=1, value=student_id)
    ws.cell(row=idx, column=2, value=student_name) # 调换了第一列和第二列的位置
    ws.cell(row=idx, column=3, value=gender)
    ws.cell(row=idx, column=4, value=birthdate)
    ws.cell(row=idx, column=5, value=height)
    ws.cell(row=idx, column=6, value=dorm)


wb.save(r'D:\\code\\python\\icourse\\processed_table\\New_Processed_Sname_table
.xlsx')

if __name__ == "__main__":
    main()

```

补充后的表格：



processed_table >  New_Processed_Sname_table.xlsx						
	A	B	C	D	E	F
1	韦铁	900785909	男	2005-07-02	1.73	东 7 舍 335
2	冯月	619438820	男	2004-07-04	1.61	东 3 舍 325
3	张为捷	764255372	女	2003-11-03	1.60	东 12 舍 301
4	王子彤	458575988	男	2003-10-12	1.78	西 17 舍 134
5	曹原齐	204377367	男	2005-03-20	1.67	西 13 舍 924
6	李欣尧	929040624	男	2004-04-03	1.64	东 19 舍 439
7	张景程	057338353	女	2004-12-07	1.75	西 6 舍 236
8	刘金亮	704074967	男	2005-08-28	1.88	东 7 舍 639
9	徐启昂	459919732	男	2005-12-15	1.68	西 19 舍 614
10	范哲伟	848332154	男	2002-03-28	1.62	西 5 舍 216
11	郑强	413818708	男	2005-01-28	1.63	西 13 舍 126
12	滕浩男	243375659	男	2004-11-21	1.65	西 1 舍 905
13	李杨	398750179	女	2004-02-07	1.67	西 6 舍 528
14	黄洁敏	892619479	男	2002-05-21	1.63	西 1 舍 817
15	蔡陆阳	887175170	男	2004-09-11	1.77	西 13 舍 522
16	刘海博	525462772	女	2004-05-26	1.69	西 18 舍 705
17	李峻锋	773426195	男	2005-06-04	1.70	西 9 舍 216
18	杨华蕊	370196670	男	2002-10-03	1.68	西 17 舍 813
19	李铎	935809174	女	2002-02-25	1.55	东 4 舍 618
20	薛建鹏	674777595	女	2005-12-12	1.64	西 6 舍 234
21	张雷超	812687661	男	2003-07-05	1.71	西 5 舍 432
22	陈凯	513794214	男	2002-07-15	1.69	东 1 舍 116
23	岳林洋	589252956	女	2005-09-02	1.59	东 4 舍 622
24	曹光宇	733693904	男	2002-10-23	1.69	西 7 舍 517

### 3.1.2 使用JDBC写入生成数据到数据库

```
package com.sxt;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Insert_new_Student {
    public static void main(String[] args) {
        System.out.println("Hello, world!");

        // Database connection parameters
        String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
        String user = "java";
        String password = "xyx2003.";
    }
}
```

```

Connection conn = null;
try {
    // Load the PostgreSQL driver (optional step in modern JDBC)
    Class.forName("org.postgresql.Driver");

    // Get a connection to the database
    conn = DriverManager.getConnection(url, user, password);
    System.out.println("Connection established successfully!");

    // Read Excel file and insert or update data into the database
    String excelFilePath =
"D:\\code\\python\\icourese\\processed_table\\New_Processed_Sname_table.xlsx";
    readAndInsertOrUpdateData(conn, excelFilePath);

} catch (SQLException e) {
    System.out.println("SQL Exception: " + e.getMessage());
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
    e.printStackTrace();
} catch (IOException e) {
    System.out.println("IOException: " + e.getMessage());
    e.printStackTrace();
} finally {
    // Close the connection
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            System.out.println("Failed to close the connection: " +
e.getMessage());
        }
    }
}

}

public static void readAndInsertOrUpdateData(Connection conn, String
excelFilePath) throws IOException, SQLException {
    try (FileInputStream fis = new FileInputStream(excelFilePath);
        Workbook workbook = new XSSFWorkbook(fis)) {

        Sheet sheet = workbook.getSheetAt(0);

        // SQL语句, 注意不包含ON CONFLICT子句
        String sql = "INSERT INTO gauss_db.s (S#, SNAME, SEX, BDATE, HEIGHT,
DORM) "
            + "VALUES (?, ?, ?, ?, ?, ?)";

        try (PreparedStatement pstmtInsert = conn.prepareStatement(sql)) {
            for (Row row : sheet) {
                if (row.getRowNum() == 0) {
                    // Skip header row
                    continue;
                }
            }
        }
    }
}

```

```

        String sNumber = getCellValue(row.getCell(0));
        String sName = getCellValue(row.getCell(1));
        String sex = getCellValue(row.getCell(2));
        String bDate = getCellValue(row.getCell(3));
        String height = getCellValue(row.getCell(4));
        String dorm = getCellValue(row.getCell(5));

        System.out.println("S#: " + sNumber + ", SName: " + sName +
            ", Sex: " + sex + ", BDate: " + bDate + ", Height: " + height + ", Dorm: " +
            dorm);

        // 设置参数
        pstmtInsert.setString(1, sNumber);
        pstmtInsert.setString(2, sName);
        pstmtInsert.setString(3, sex);
        pstmtInsert.setString(4, bDate);
        pstmtInsert.setString(5, height);
        pstmtInsert.setString(6, dorm);

        try {
            pstmtInsert.executeUpdate();
        } catch (SQLException ex) {
            // 如果发生主键冲突, 执行更新操作
            if (ex.getSQLState().equals("23505")) { // 23505 是主键冲突
                System.out.println("Duplicate key found, updating
                    record...");

                String updateSql = "UPDATE gauss_db.s "
                    + "SET SNAME = ?, "
                    + "SEX = ?, "
                    + "BDATE = ?, "
                    + "HEIGHT = ?, "
                    + "DORM = ? "
                    + "WHERE S# = ?";

                try (PreparedStatement pstmtUpdate =
                    conn.prepareStatement(updateSql)) {
                    pstmtUpdate.setString(1, sName);
                    pstmtUpdate.setString(2, sex);
                    pstmtUpdate.setString(3, bDate);
                    pstmtUpdate.setString(4, height);
                    pstmtUpdate.setString(5, dorm);
                    pstmtUpdate.setString(6, sNumber);

                    pstmtUpdate.executeUpdate();
                }
            } else {
                throw ex; // 如果不是主键冲突, 抛出异常
            }
        }
    }
}

```

的SQL状态码

```

private static String getCellValue(Cell cell) {
    if (cell == null) {
        return "";
    }
    switch (cell.getCellTypeEnum()) {
        case STRING:
            return cell.getStringCellValue();
        case NUMERIC:
            if (DateUtil.isCellDateFormatted(cell)) {
                return cell.getDateCellValue().toString();
            } else {
                return Double.toString(cell.getNumericCellValue());
            }
        case BOOLEAN:
            return Boolean.toString(cell.getBooleanCellValue());
        case FORMULA:
            return cell.getCellFormula();
        case BLANK:
            return "";
        default:
            return "";
    }
}
}
}

```

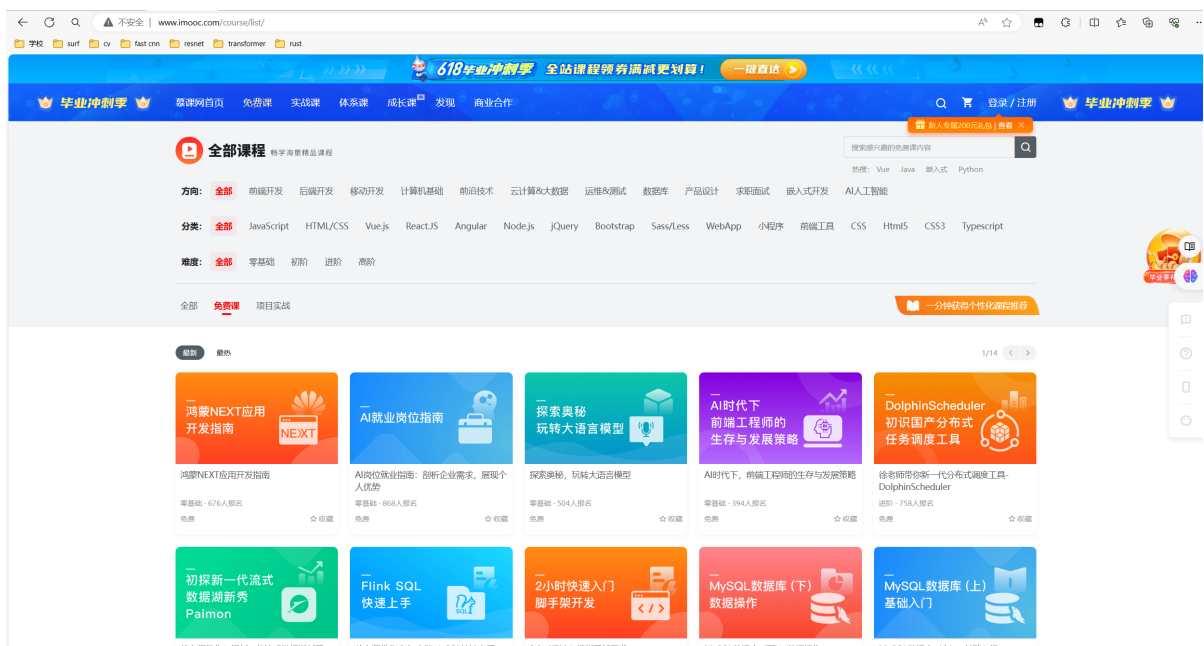
写入后:

s#	sname	sex	bdate	height	dorm
01032023	周文	男	2004-06-10 00:00:00	1.80	东 6 舍 221
01032001	张翰梅	女	2004-11-17 00:00:00	1.58	东 1 舍 312
01032005	刘静	女	2003-01-10 00:00:00	1.63	东 1 舍 312
01032112	曹静	男	2003-02-20 00:00:00	1.71	东 6 舍 221
03031011	王倩	女	2004-12-20 00:00:00	1.66	东 2 舍 104
03031014	赵思扬	男	2002-06-06 00:00:00	1.85	东 18 舍 421
03031009	田菲	女	2003-08-11 00:00:00	1.60	东 2 舍 104
03031033	蔡明明	男	2003-03-12 00:00:00	1.75	东 18 舍 423
03031056	曹子铃	女	2004-12-15 00:00:00	1.65	东 2 舍 305
03031051	周剑	男	2002-05-08 00:00:00	1.68	东 18 舍 422
01032010	王海	男	2003-04-05 00:00:00	1.72	东 6 舍 221
09710964	冯月	男	2003-11-29 00:00:00	1.61	西 10 舍 918
92188103	张为建	女	2005-03-29 00:00:00	1.70	西 14 舍 737
08362874	王子彭	女	2004-03-28 00:00:00	1.75	西 14 舍 822
82743236	曹国齐	女	2005-08-04 00:00:00	1.73	西 3 舍 714
36094232	李欣尧	男	2005-07-18 00:00:00	1.84	西 8 舍 940
18064528	张豪程	女	2002-09-12 00:00:00	1.69	西 3 舍 710
77746176	刘金英	女	2002-02-16 00:00:00	1.59	东 4 舍 309
54091277	徐松昂	男	2004-08-10 00:00:00	1.75	东 6 舍 315
76020571	范明伟	女	2003-09-11 00:00:00	1.66	东 4 舍 421
15429790	郑强	男	2003-05-10 00:00:00	1.69	西 9 舍 110
21102338	熊伟男	男	2005-06-05 00:00:00	1.65	西 9 舍 221
03105636	李彬	男	2003-01-09 00:00:00	1.64	东 6 舍 730
33172909	蔡洪敏	男	2004-06-19 00:00:00	1.69	东 7 舍 431
72904686	蔡林阳	男	2004-10-20 00:00:00	1.63	东 8 舍 309
10718620	刘阳博	女	2003-12-13 00:00:00	1.57	西 5 舍 728
31173336	李敏峰	女	2004-07-21 00:00:00	1.56	西 2 舍 916
01350228	纪华瑞	女	2002-06-05 00:00:00	1.58	东 15 舍 808
08256750	李群	女	2003-09-09 00:00:00	1.64	东 1 舍 621
87385945	薛建鹏	男	2004-01-31 00:00:00	1.66	西 10 舍 433
85459463	张鑫超	男	2005-11-18 00:00:00	1.84	西 7 舍 312
10957595	陈凯	女	2003-09-12 00:00:00	1.61	西 5 舍 612
31253254	陈松涛	男	2003-03-05 00:00:00	1.70	西 9 舍 311
90809408	曹光宇	男	2002-04-11 00:00:00	1.89	东 8 舍 323
61021196	梁琪	女	2004-08-15 00:00:00	1.77	西 2 舍 437
17504501	王瑞琳	女	2002-11-29 00:00:00	1.64	西 5 舍 903

## 3.1.2 补充C数据

### 3.1.2.1 C数据爬取

只爬取课程名，其他信息通过补全生成。使用python爬虫爬取数据，爬取网站为[中国大学mooc](http://www.mooc.cn)一个子页面，页面如下：



代码：

```
from bs4 import BeautifulSoup
import requests
import os
from openpyxl import workbook, load_workbook

# 文件路径
file_path = 'D:\\code\\python\\icourse\\cname.xlsx'

# 创建或加载Excel文件
if os.path.exists(file_path):
    wb = load_workbook(file_path)
    ws = wb.active
else:
    wb = workbook()
    ws = wb.active
    # 写入表头（如果需要）
    ws.append(['Course Name'])

# 请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36',
```

```

        'cookie': 'imooc_uuid=86ab3b90-3f49-41aa-ab5e-52d53e209d85; imooc_isnew=1;
imooc_isnew_ct=1717230404; tgw_l7_route=126bb7f6409ec5ff7744080c81a70225;
sajssdk_2015_cross_new_user=1;
sensorsdata2015jssdkcross=%7B%22distinct_id%22%3A%2218fd2e889f68f0-
02cdb6893c548f8-26001c51-2359296-
18fd2e889f75a8%22%2C%22first_id%22%3A%22%22%2C%22props%22%3A%7B%22%24latest_traffic_source_type%22%3A%22%E7%9B%B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest_search_keyword%22%3A%22%E6%9C%AA%E5%8F%96%E5%88%B0%E5%80%BC_%E7%9B%B4%E6%8E%A5%E6%89%93%E5%BC%80%22%2C%22%24latest_referrer%22%3A%22%22%7D%2C%22identities%22%3A%22eyJkawRlbnRpdHlfY29va2l1x2lkijoimThmZDJlODg5ZjY4ZjAtMDJjZGI2ODkzYzU0OGY4LTI2MDAxYzUxLTIzNTkyOTYtMThmZDJlODg5Zjc1YTgifiQ%3D%3D%22%2C%22history_login_id%22%3A%7B%22name%22%3A%22%22%2C%22value%22%3A%22%22%7D%2C%22%24device_id%22%3A%2218fd2e889f68f0-02cdb6893c548f8-26001c51-2359296-18fd2e889f75a8%22%7D;
Hm_lvt_f0cfcccd7b1393990c78efdeebff3968=1717230406;
Hm_lpvtf0cfcccd7b1393990c78efdeebff3968=1717230412; cvde=665adb44bd037-4'
    }
}

```

# 循环访问第2页到第10页的内容

```
for page in range(1, 11):
```

```
    # 构造URL
```

```
    url = f'http://www.imooc.com/course/list/?page={page}'
```

```
    # 发送HTTP请求获取网页内容
```

```
    response = requests.get(url, headers=headers)
```

```
    # 创建BeautifulSoup对象
```

```
    soup = BeautifulSoup(response.text, 'html.parser')
```

```
    # 提取符合条件的<a>标签
```

```
    a_tags = soup.find_all('a', class_="item free")
```

```
    # 提取并写入data-title属性的值
```

```
    for tag in a_tags:
```

```
        data_title = tag.get('data-title')
```

```
        if data_title:
```

```
            print(data_title)
```

```
            ws.append([data_title])
```

```
# 保存Excel文件
```

```
wb.save(file_path)
```

```
print('数据爬取完成!')
```

爬取到的课程名示例：

	A	Y
1		
2	鸿蒙NEXT应用开发指南	
3	AI岗位就业指南：剖析企业需求，展现个人优势	
4	探索奥秘，玩转大语言模型	
5	AI时代下，前端工程师的生存与发展策略	
6	徐老师带你新一代分布式调度工具-DolphinScheduler	
7	徐老师带你入门新一代流式数据湖新秀--Apache Paimon	
8	Course Name	
9	2小时极速入门脚手架开发	
10		
11	MySQL数据库（上）- 基础入门	
12	Express 基础入门	
13	Calcite数据管理与SQL优化实战	
14	TypeScript极速入门	
15	LLM行业领军大佬 带你转型大语言模型-应对市场变革	
16	从Docker到K8S，容器技术演进之路	
17	全面解读大热行业 物联网/嵌入式前景与就业	
18	《前端高级工程师》体验课	
19	《DBA数据库工程师》体验课	
20	《大数据开发2023》体验课	
21	《产品经理》体验课	
22	《Java高级工程师》体验课	
23	手把手带你前端快速入门	
24	vue3 从0到1 全流程封装基础组件	

### 3.1.2.2 C数据补充

使用python语言补充表格。

补充时的设计：

(1) 学时不能太少也不能太多，统一设置为24-80区间，且能被4整除（考虑到一节课2h），这样设置的学时比较合理。

(2) 学分credit应该和学时成正相关，即学时越多学分应该相应更高。使用学时除以20，当小数点后小于0.5时向下取整，大于等于0.5时保留到0.5。

```
import pandas as pd
```

```

import random
import openpyxl
from faker import Faker
from math import floor

# 创建一个新的Excel文件
file_path = r'D:\code\python\icourese\processed_table\Processed_Course.xlsx'
wb = openpyxl.Workbook()
ws = wb.active
ws.title = 'Sheet1'

# 生成AA-BB
AA_values = ["CS", "EE", "ME", "CE", "BE", "AE", "EC", "FN", "MA", "MT", "PH",
             "CH", "BI", "AR", "DE", "PS", "ST", "GE", "HI", "LS"]
excluded_codes = ["CS-01", "CS-02", "CS-04", "CS-05", "EE-02", "EE-03", "CS-03",
                  "EE-01"]
generated_codes = set(excluded_codes)

while len(generated_codes) < 100 + len(excluded_codes):
    AA = random.choice(AA_values)
    BB = f"{random.randint(1, 99):02}"
    code = f"{AA}-{BB}"
    if code not in generated_codes:
        generated_codes.add(code)

# 去掉排除的代码，只保留生成的100个代码
generated_codes -= set(excluded_codes)
generated_codes = list(generated_codes)[:100]

# 写入第一列
for i, code in enumerate(generated_codes, start=2):
    ws.cell(row=i, column=1, value=code)

# 读取现有Excel文件中的数据并写入新文件的第二列
origin_file_path = r'D:\code\python\icourese\origin_table\Cname.xlsx'
origin_data = pd.read_excel(origin_file_path)
second_column_data = origin_data.iloc[1:101, 0].tolist() # 读取第2行到第101行的数据

for i, data in enumerate(second_column_data, start=2):
    ws.cell(row=i, column=2, value=data)

# 随机生成指定范围内的数字，并写入第三列和第四列
for i in range(2, 102):
    num = random.choice(range(24, 101, 4))
    decimal_part = num % 20 / 20.0 # 计算小数部分
    if decimal_part >= 0.5:
        adjusted_num = floor(num / 20.0) + 0.5
    else:
        adjusted_num = floor(num / 20.0)
    ws.cell(row=i, column=3, value=num)
    ws.cell(row=i, column=4, value=adjusted_num)

# 随机生成中文名字并写入第五列
faker = Faker('zh_CN')
for i in range(2, 102):
    name = faker.name()

```



```
ws.cell(row=i, column=5, value=name)
```

```
# 保存Excel文件
```

```
wb.save(file_path)
```

补充后的S表：

c#	cname	period	credit	teacher
CS-01	数据结构	60	3.0	张军
CS-02	计算机组成原理	80	4.0	王亚伟
CS-04	人工智能	40	2.0	李蕾
CS-05	深度学习	40	2.0	崔昀
EE-01	信号与系统	60	3.0	张明
EE-02	数字逻辑电路	100	5.0	胡海东
EE-03	光电子学与光子学	40	2.0	石韬
ME-44	AI岗位就业指南：剖析1	28	1.0	黄兰英
ME-45	探索奥秘，玩转大语言1	56	2.0	李冬梅
GE-77	AI时代下，前端工程师1	100	5.0	蔡明
AR-57	徐老师带你新一代分布1	28	1.0	梁萍
AR-58	徐老师带你入门新一代1	88	4.0	蒋冬梅
EE-14	徐老师带你 2小时 Flink	44	2.0	张磊
BE-14	2小时极速入门脚手架开	52	2.0	张帆
DE-32	MySQL数据库（下）-参	40	2.0	杜楠
ST-43	MySQL数据库（上）-参	32	1.0	蔡秀兰
CS-72	Express 基础入门	52	2.0	郝阳
BE-89	Calcite数据管理与SQL	92	4.0	梁明
BE-86	TypeScript极速入门	100	5.0	曾秀华
LS-09	LLM行业领军大佬 带你	64	3.0	吴玉
ME-65	从Docker到K8S，容器	64	3.0	刘丽丽
DE-07	全面解读大热行业 物联	28	1.0	金玉兰
AE-72	《前端高级工程师》体	96	4.0	杨林
CE-09	《DBA数据库工程师》1	92	4.0	康桂芳
BE-78	《大数据开发2023》体	96	4.0	杨桂芳
CE-59	《产品经理》体验课	68	3.0	杨倩
BI-36	《Java高级工程师》体	72	3.0	刘桂珍
PS-01	手把手带你前端快速入	48	2.0	李玲
AR-49	vue3 从0到1 全流程封	76	3.0	万文
EE-09	《前端共学会》之图表	56	2.0	黄桂花
BI-34	清华大学语音实验室大	96	4.0	苏婷婷
HI-48	Gin最新版入门与案例实	72	3.0	王华
BI-35	RN实现原生扫码功能	76	3.0	刘帆
BE-05	G1—Java新一代垃圾回	48	2.0	吕刚
AR-56	CSS3布局样式与案例	84	4.0	魏佳
LS-42	Sharding-JDBC 入门与	68	3.0	颜博

### 3.1.2.3 使用JDBC写入生成数据到数据库

```
package com.sxt;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Insert_Course {
    public static void main(String[] args) {
        System.out.println("Hello, World!");

        // Database connection parameters
        String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
        String user = "java";
        String password = "xyx2003.";

        Connection conn = null;
        try {
            // Load the PostgreSQL driver (optional step in modern JDBC)
            Class.forName("org.postgresql.Driver");

            // Get a connection to the database
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("Connection established successfully!");

            // Read Excel file and insert data into the database
            String excelFilePath =
"D:\\code\\python\\icourese\\processed_table\\Processed_Course.xlsx";
            insertDataFromExcel(conn, excelFilePath);

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("IOException: " + e.getMessage());
            e.printStackTrace();
        } finally {
            // Close the connection
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
```

```

        System.out.println("Failed to close the connection: " +
e.getMessage());
    }
}

// Method to read data from Excel and insert it into the database
public static void insertDataFromExcel(Connection conn, String excelFilePath)
throws IOException, SQLException {
    FileInputStream fis = new FileInputStream(excelFilePath);
    workbook workbook = new XSSFWorkbook(fis);
    Sheet sheet = workbook.getSheetAt(0);

    String sql = "INSERT INTO gauss_db.c (c#, cname, period, credit, teacher)
VALUES (?, ?, ?, ?, ?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    for (Row row : sheet) {
        if (row.getRowNum() == 0) {
            // skip header row
            continue;
        }

        String cNum = getCellValue(row.getCell(0));
        String cName = getCellValue(row.getCell(1));
        int period = (int) row.getCell(2).getNumericCellValue(); // Ensuring
period is an integer
        int credit = (int) row.getCell(3).getNumericCellValue(); // Ensuring
credit is an integer
        String teacher = getCellValue(row.getCell(4));

        pstmt.setString(1, cNum);
        pstmt.setString(2, cName);
        pstmt.setInt(3, period);
        pstmt.setInt(4, credit);
        pstmt.setString(5, teacher);

        pstmt.addBatch();
    }

    pstmt.executeBatch();
    workbook.close();
    fis.close();
}

// Helper method to get cell value as string
private static String getCellValue(Cell cell) {
    if (cell == null) {
        return "";
    }
    switch (cell.getCellTypeEnum()) {
        case STRING:
            return cell.getStringCellValue();
        case NUMERIC:

```

```

        return String.valueOf((int) cell.getNumericCellValue()); //
Assuming numeric values can be cast to int
    case BOOLEAN:
        return String.valueOf(cell.getBooleanCellValue());
    case FORMULA:
        return cell.getCellFormula();
    default:
        return "";
    }
}
}

```

### 3.1.3 补充SC数据并随机删除200条grade低于60分的数据。

```

package com.sxt;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Random;
import java.util.Set;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class CourseSelectionManager {

    public static void main(String[] args) {
        System.out.println("Hello, World!");

        // 数据库连接参数
        String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
        String user = "java";
        String password = "xyx2003.";

        Connection conn = null;
        try {
            // 加载PostgreSQL驱动（在现代JDBC中是可选的）
            Class.forName("org.postgresql.Driver");

            // 获取数据库连接
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("连接成功建立!");

            // 创建一个ExecutorService来管理线程
            ExecutorService executor = Executors.newFixedThreadPool(10);

            // 第一步：在一个单独的线程中为学生分配课程
            executor.submit(() -> {
                try {
                    assignCoursesToStudents(conn);
                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        } catch (SQLException e) {
            e.printStackTrace();
        }
    });

    // 第二步: 在一个单独的线程中删除随机低分条目
    executor.submit(() -> {
        try {
            deleteRandomLowGrades(conn);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    });

    // 关闭executor并等待任务完成
    executor.shutdown();
    executor.awaitTermination(1, TimeUnit.HOURS);

} catch (SQLException e) {
    System.out.println("SQL异常: " + e.getMessage());
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    System.out.println("找不到PostgreSQL JDBC驱动: " + e.getMessage());
    e.printStackTrace();
} catch (InterruptedException e) {
    System.out.println("线程中断: " + e.getMessage());
    e.printStackTrace();
} finally {
    // 关闭连接
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            System.out.println("关闭连接失败: " + e.getMessage());
        }
    }
}
}
}

```

```

private static void assignCoursesToStudents(Connection conn) throws
SQLException {
    Statement stmt = null;
    ResultSet rs = null;
    List<String> studentIds = new ArrayList<>();
    List<String> courseIds = new ArrayList<>();
    Random random = new Random();

    try {
        stmt = conn.createStatement();
        // 获取所有学生ID
        rs = stmt.executeQuery("SELECT S# FROM gauss_db.S");
        while (rs.next()) {
            studentIds.add(rs.getString("S#"));
        }
        rs.close();
    }
}

```

```

// 获取所有课程ID
rs = stmt.executeQuery("SELECT C# FROM gauss_db.C");
while (rs.next()) {
    courseIds.add(rs.getString("C#"));
}
rs.close();

// 创建一个线程池
ExecutorService executor = Executors.newFixedThreadPool(10);

// 为每个学生分配20门课程
for (String studentId : studentIds) {
    executor.submit(() -> {
        Set<String> assignedCourses = new HashSet<>();
        int coursesAssigned = 0;
        while (coursesAssigned < 20) {
            String courseId =
courseIds.get(random.nextInt(courseIds.size()));
            if (!assignedCourses.contains(courseId)) {
                double grade = generateGrade(random);
                try (Statement localStmt = conn.createStatement()) {
                    localStmt.executeUpdate("INSERT INTO gauss_db.SC
(S#, C#, GRADE) VALUES "
                                + "(" + studentId + ", " + courseId +
                                ", " + grade + ")");

                    assignedCourses.add(courseId);
                    coursesAssigned++;
                } catch (SQLException e) {
                    if (e.getSQLState().equals("23505")) {
                        // 主键重复，跳过此课程并尝试另一个
                        continue;
                    } else {
                        throw new RuntimeException(e);
                    }
                }
            }
        }
    });
}

executor.shutdown();
executor.awaitTermination(1, TimeUnit.HOURS);
System.out.println("课程成功分配给学生！");

} catch (InterruptedException e) {
    System.out.println("线程中断: " + e.getMessage());
    e.printStackTrace();
} finally {
    if (stmt != null) {
        stmt.close();
    }
    if (rs != null) {
        rs.close();
    }
}
}

```

```

private static double generateGrade(Random random) {
    // 生成平均分为80的成绩
    return 70 + random.nextGaussian() * 10;
}

private static void deleteRandomLowGrades(Connection conn) throws
SQLException {
    Statement stmt = null;
    ResultSet rs = null;
    List<String> lowGradeEntries = new ArrayList<>();
    Random random = new Random();

    try {
        stmt = conn.createStatement();
        // 获取所有成绩低于60的条目
        rs = stmt.executeQuery("SELECT S#, C# FROM gauss_db.SC WHERE GRADE <
60");

        while (rs.next()) {
            lowGradeEntries.add(rs.getString("S#") + "," +
rs.getString("C#"));
        }
        rs.close();

        // 随机删除200个低分条目
        ExecutorService executor = Executors.newFixedThreadPool(10);

        for (int i = 0; i < 200 && !lowGradeEntries.isEmpty(); i++) {
            executor.submit(() -> {
                int index = random.nextInt(lowGradeEntries.size());
                String[] ids = lowGradeEntries.remove(index).split(",");
                try (Statement localStmt = conn.createStatement()) {
                    localStmt.executeUpdate("DELETE FROM gauss_db.SC WHERE S#
= '" + ids[0] + "' AND C# = '" + ids[1] + "'");
                } catch (SQLException e) {
                    throw new RuntimeException(e);
                }
            });
        }

        executor.shutdown();
        executor.awaitTermination(1, TimeUnit.HOURS);
        System.out.println("200个低分条目成功删除!");

    } catch (InterruptedException e) {
        System.out.println("线程中断: " + e.getMessage());
        e.printStackTrace();
    } finally {
        if (stmt != null) {
            stmt.close();
        }
        if (rs != null) {
            rs.close();
        }
    }
}

```



```
}
```

### 关键点说明：

#### 关键代码：

```
// 创建一个ExecutorService来管理线程
ExecutorService executor = Executors.newFixedThreadPool(10);

// 第一步：在一个单独的线程中为学生分配课程
executor.submit(() -> {
    try {
        assignCoursesToStudents(conn);
    } catch (SQLException e) {
        e.printStackTrace();
    }
});

// 第二步：在一个单独的线程中删除随机低分条目
executor.submit(() -> {
    try {
        deleteRandomLowGrades(conn);
    } catch (SQLException e) {
        e.printStackTrace();
    }
});

// 关闭executor并等待任务完成
executor.shutdown();
executor.awaitTermination(1, TimeUnit.HOURS);
```

1. **并行执行**：使用 `ExecutorService` 将 `assignCoursesToStudents` 和 `deleteRandomLowGrades` 两个任务并行提交，确保它们可以同时运行。
2. **线程安全**：每个任务使用自己的 `Statement` 对象来避免并发问题。
3. **关闭和等待终止**：在提交任务后，调用 `shutdown()` 来停止接受新任务，并调用 `awaitTermination()` 来等待所有任务完成，确保所有操作在给定时间内（此处为1小时）完成。

生成后各表行数：



**C**  
表

opengauss  
mydb  
gauss\_db

OID  
49419

所有者  
java

行  
107



**S**  
表

opengauss  
mydb  
gauss\_db

OID  
49414

所有者  
java

行  
1,068



**SC**  
表

opengauss  
mydb  
gauss\_db

OID  
49424

所有者  
java

行  
21,186

### 3.2 . 在 S中补充至 5000 行，在 C表中补充数据至约 1000行，在 SC表中补充数据至约 200000 行。

由于数据太多不方便爬取，故此处S补充数据和C补充数据均通过python代码随机生成。

#### 3.2.1 S数据生成：

```
import openpyxl
import random
from datetime import datetime, timedelta
from faker import Faker

def generate_student_id(existing_ids):
    exclude_ids = ['01032010', '01032023', '01032001', '01032005',
                   '01032112', '03031011', '03031014', '03031051',
```

```

'03031009', '03031033', '03031056']

while True:
    student_id = ''.join(random.choices('0123456789', k=8))
    if student_id not in existing_ids and student_id not in exclude_ids:
        return student_id

def generate_gender():
    return random.choice(['男', '女'])

def generate_birthdate():
    start_date = datetime.strptime('2002-01-01', '%Y-%m-%d')
    end_date = datetime.strptime('2005-12-31', '%Y-%m-%d')
    random_date = start_date + timedelta(days=random.randint(0, (end_date -
start_date).days))
    return random_date.strftime('%Y-%m-%d')

def generate_height(gender):
    if gender == '男':
        return round(random.uniform(1.6, 1.9), 2)
    elif gender == '女':
        return round(random.uniform(1.55, 1.8), 2)
    else:
        return None

def generate_dorm(assigned_dorms, gender):
    while True:
        building = random.choice(['东', '西'])

        if gender == '女':
            # 女生只能分配到楼层 1-5 和 11-15
            floor = random.choice([1, 2, 3, 4, 5, 11, 12, 13, 14, 15])
        else:
            # 男生可以分配到除了 1-5 和 11-15 外的其他楼层
            floor = random.randint(6, 10) # 楼层范围从6到10

        fls = str(random.randint(1, 9)) # 确保是两位数, 如09
        room = str(random.randint(1, 40)).zfill(2) # 房间号, 确保是两位数, 如05

        room_number = f'{building} {floor} 舍 {fls}{room}'

        # 获取已分配的宿舍情况
        male_dorms = [dorm['dorm'] for dorm in assigned_dorms if dorm['gender']
== '男']
        female_dorms = [dorm['dorm'] for dorm in assigned_dorms if dorm['gender']
== '女']

        # 如果只有男性宿舍或只有女性宿舍, 则后续只能分配相应性别
        if room_number in female_dorms and gender == '女':
            continue
        elif room_number in male_dorms and gender == '男':
            continue

        # 如果没有冲突的宿舍分配, 则返回宿舍号和性别
        break

```

```

        return room_number, gender

def main():
    # 打开Excel文件
    wb =
openpyxl.load_workbook(r'D:\code\python\icourese\processed_table\New_Processed_Sn
ame_table.xlsx')
    ws = wb.active

    existing_ids = set()
    assigned_dorms = []

    # 读取已有的学号并添加到exclude_ids中
    for row in ws.iter_rows(min_row=2, max_col=1, values_only=True):
        existing_ids.add(row[0])

    # 使用faker生成中文名
    faker = Faker(locale='zh_CN')

    # 生成4000个条目
    for idx in range(1, 4001):
        student_name = faker.name()
        student_id = generate_student_id(existing_ids)
        gender = generate_gender()
        birthdate = generate_birthdate()
        height = generate_height(gender)
        dorm, dorm_gender = generate_dorm(assigned_dorms, gender)
        existing_ids.add(student_id)
        assigned_dorms.append({'dorm': dorm, 'gender': dorm_gender})

    # 写入Excel文件，第一列为学号，第二列为姓名，依次类推
    ws.cell(row=len(existing_ids), column=1, value=student_id)
    ws.cell(row=len(existing_ids), column=2, value=student_name)
    ws.cell(row=len(existing_ids), column=3, value=gender)
    ws.cell(row=len(existing_ids), column=4, value=birthdate)
    ws.cell(row=len(existing_ids), column=5, value=height)
    ws.cell(row=len(existing_ids), column=6, value=dorm)

    # 保存到新的Excel文件
    wb.save(r'D:\code\python\icourese\last_work\Sname.xlsx')

if __name__ == "__main__":
    main()

```

### 3.2.2 C数据生成

已经存在100条的数据，之前爬取到的数据有接近500条，直接按之前爬取到的数据的名称后缀上加上'I'和'II'，使课程名加倍。

```

import os
import openpyxl

```

```

# 定义读取Excel文件路径和生成的Excel文件路径
input_file = r'D:\code\python\icourese\origin_table\Cname.xlsx'
output_folder = r'D:\code\python\icourese\temp'
output_file = os.path.join(output_folder, 'Course.xlsx')

# 打开并读取Excel文件
wb = openpyxl.load_workbook(input_file)
sheet = wb.active

# 获取第一列的课程名数据
courses = [sheet.cell(row=i, column=1).value for i in range(1, sheet.max_row + 1)]

# 创建一个新的Excel文件
wb_output = openpyxl.Workbook()
sheet_output = wb_output.active
sheet_output.title = 'Courses'

# 写入Excel文件
row_index = 1
for course in courses:
    sheet_output.cell(row=row_index, column=1, value=f'{course}-I')
    sheet_output.cell(row=row_index + 1, column=1, value=f'{course}-II')
    row_index += 2 # 每个课程名的两个变体占据两行

# 确保目标文件夹存在, 如果不存在则创建
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# 保存Excel文件
wb_output.save(output_file)
print(f"课程名变体已成功保存到 {output_file}")

```

完善课程信息:

```

import random
import openpyxl
from math import floor
from faker import Faker

# 创建一个新的Excel文件
file_path = r'D:\code\python\icourese\last_work\Cname.xlsx'
wb = openpyxl.Workbook()
ws = wb.active
ws.title = 'Sheet1'

# 生成AA-BB
AA_values = ["CS", "EE", "ME", "CE", "BE", "AE", "EC", "FN", "MA", "MT", "PH",
             "CH", "BI", "AR", "DE", "PS", "ST", "GE", "HI", "LS",
             "MU", "LT", "ET", "IT", "SE", "AI", "CI", "EN", "LE", "FE", "RE",
             "UE", "IE", "OE", "YE", "WE", "NE", "XE", "TE"]
excluded_codes = ["CS-01", "CS-02", "CS-04", "CS-05", "EE-02", "EE-03", "CS-03",
                  "EE-01"]

```

```

# 读取现有Excel文件中的课程代码并添加到排除列表中
existing_codes_wb =
openpyxl.load_workbook(r'D:\code\python\icourse\processed_table\Processed_Course
.xlsx')
existing_codes_ws = existing_codes_wb.active
existing_codes = [existing_codes_ws.cell(row=i, column=1).value for i in range(2,
existing_codes_ws.max_row + 1)]
excluded_codes += existing_codes

# 读取课程名的Excel文件
course_file = r'D:\code\python\icourse\temp\Course.xlsx'
course_wb = openpyxl.load_workbook(course_file)
course_ws = course_wb.active

# 获取课程名列的数据
generated_courses = [course_ws.cell(row=i, column=1).value for i in range(1,
course_ws.max_row + 1)]

# 创建 Faker 实例
faker = Faker('zh-CN')

# 写入第一列（课程代码）
generated_codes = set()
while len(generated_codes) < 900:
    AA = random.choice(AA_values)
    BB = f"{random.randint(1, 99):02}"
    code = f"{AA}-{BB}"
    if code not in generated_codes and code not in excluded_codes:
        generated_codes.add(code)

# 写入第二列（课程名）
for i, code in enumerate(generated_codes, start=2):
    ws.cell(row=i, column=1, value=code)
    ws.cell(row=i, column=2, value=generated_courses[i - 2])

# 随机生成指定范围内的数字，并写入第三列和第四列
for i in range(2, 902):
    num = random.choice(range(24, 101, 4))
    decimal_part = num % 20 / 20.0 # 计算小数部分
    if decimal_part >= 0.5:
        adjusted_num = floor(num / 20.0) + 0.5
    else:
        adjusted_num = floor(num / 20.0)
    ws.cell(row=i, column=3, value=num)
    ws.cell(row=i, column=4, value=adjusted_num)

# 随机生成中文名字并写入第五列
for i in range(2, 902):
    name = faker.name()
    ws.cell(row=i, column=5, value=name)

# 保存Excel文件
wb.save(file_path)
print(f"文件已保存到 {file_path}")

```

### 3.2.3 S和C数据写入

#### S数据写入

```
package com.sxt;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Insert_Student2 {
    public static void main(String[] args) {
        System.out.println("Hello, world!");

        // Database connection parameters
        String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
        String user = "java";
        String password = "xyx2003.";

        Connection conn = null;
        try {
            // Load the PostgreSQL driver (optional step in modern JDBC)
            Class.forName("org.postgresql.Driver");

            // Get a connection to the database
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("Connection established successfully!");

            // Read Excel file and insert or update data into the database
            String excelFilePath =
"D:\\code\\python\\icourese\\last_work\\Sname.xlsx";
            readAndInsertOrUpdateData(conn, excelFilePath);

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("IOException: " + e.getMessage());
            e.printStackTrace();
        } finally {
            // Close the connection
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {

```

```

        System.out.println("Failed to close the connection: " +
e.getMessage());
    }
}

}

    public static void readAndInsertOrUpdateData(Connection conn, String
excelFilePath) throws IOException, SQLException {
        try (FileInputStream fis = new FileInputStream(excelFilePath);
            Workbook workbook = new XSSFWorkbook(fis)) {

            Sheet sheet = workbook.getSheetAt(0);

            // SQL语句, 注意不包含ON CONFLICT子句
            String sql = "INSERT INTO gauss_db.s (S#, SNAME, SEX, BDATE, HEIGHT,
DORM) "

                + "VALUES (?, ?, ?, ?, ?, ?)";

            try (PreparedStatement pstmtInsert = conn.prepareStatement(sql)) {
                for (Row row : sheet) {
                    if (row.getRowNum() == 0) {
                        // skip header row
                        continue;
                    }

                    String sNumber = getCellValue(row.getCell(0));
                    String sName = getCellValue(row.getCell(1));
                    String sex = getCellValue(row.getCell(2));
                    String bDate = getCellValue(row.getCell(3));
                    String height = getCellValue(row.getCell(4));
                    String dorm = getCellValue(row.getCell(5));

                    System.out.println("S#: " + sNumber + ", SName: " + sName +
", Sex: " + sex + ", BDate: " + bDate + ", Height: " + height + ", Dorm: " +
dorm);

                    // 设置参数
                    pstmtInsert.setString(1, sNumber);
                    pstmtInsert.setString(2, sName);
                    pstmtInsert.setString(3, sex);
                    pstmtInsert.setString(4, bDate);
                    pstmtInsert.setString(5, height);
                    pstmtInsert.setString(6, dorm);

                    try {
                        pstmtInsert.executeUpdate();
                        System.err.println("succeed insert");
                    } catch (SQLException ex) {
                        // 如果发生主键冲突, 执行更新操作
                        if (ex.getSQLState().equals("23505")) { // 23505 是主键冲突
                            System.out.println("Duplicate key found, updating
record...");

                            String updateSql = "UPDATE gauss_db.s "

```



```

        + "SET SNAME = ?, "
        + "SEX = ?, "
        + "BDATE = ?, "
        + "HEIGHT = ?, "
        + "DORM = ? "
        + "WHERE S# = ?";

        try (PreparedStatement pstmtUpdate =
conn.prepareStatement(updatesql)) {
            pstmtUpdate.setString(1, sName);
            pstmtUpdate.setString(2, sex);
            pstmtUpdate.setString(3, bDate);
            pstmtUpdate.setString(4, height);
            pstmtUpdate.setString(5, dorm);
            pstmtUpdate.setString(6, sNumber);

            pstmtUpdate.executeUpdate();
        }
    } else {
        throw ex; // 如果不是主键冲突，抛出异常
    }
}
}
}
}

private static String getCellValue(Cell cell) {
    if (cell == null) {
        return "";
    }
    switch (cell.getCellTypeEnum()) {
        case STRING:
            return cell.getStringCellValue();
        case NUMERIC:
            if (DateUtil.isCellDateFormatted(cell)) {
                return cell.getDateCellValue().toString();
            } else {
                return Double.toString(cell.getNumericCellValue());
            }
        case BOOLEAN:
            return Boolean.toString(cell.getBooleanCellValue());
        case FORMULA:
            return cell.getCellFormula();
        case BLANK:
            return "";
        default:
            return "";
    }
}
}
}

```

C数据写入:

```

package com.sxt;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Random;
import java.util.Set;

public class Courseselection2 {

    public static void main(String[] args) {
        System.out.println("Hello, world!");

        // Database connection parameters
        String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
        String user = "java";
        String password = "xyx2003.";

        Connection conn = null;
        try {
            // Load the PostgreSQL driver (optional step in modern JDBC)
            Class.forName("org.postgresql.Driver");

            // Get a connection to the database
            conn = DriverManager.getConnection(url, user, password);
            System.out.println("Connection established successfully!");

            // Assign courses to students
            assignCoursesToStudents(conn);

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
            e.printStackTrace();
        } finally {
            // Close the connection
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
                    System.out.println("Failed to close the connection: " +
e.getMessage());
                }
            }
        }
    }
}

```

```

private static void assignCoursesToStudents(Connection conn) throws
SQLException {
    Statement stmt = null;
    ResultSet rs = null;
    List<String> studentIds = new ArrayList<>();
    List<String> courseIds = new ArrayList<>();
    Random random = new Random();

    try {
        stmt = conn.createStatement();
        // Get all student IDs
        rs = stmt.executeQuery("SELECT S# FROM gauss_db.S");
        while (rs.next()) {
            studentIds.add(rs.getString("S#"));
        }
        rs.close();

        // Get all course IDs
        rs = stmt.executeQuery("SELECT C# FROM gauss_db.C");
        while (rs.next()) {
            courseIds.add(rs.getString("C#"));
        }
        rs.close();

        // For each student, assign courses
        for (String studentId : studentIds) {
            // Check if the student already has course records
            rs = stmt.executeQuery("SELECT COUNT(*) AS course_count FROM
gauss_db.SC WHERE S# = '" + studentId + "'");
            rs.next();
            int existingCourses = rs.getInt("course_count");
            rs.close();

            int coursesToAssign = existingCourses > 0 ? 20 : 40;
            System.err.println("course="+ coursesToAssign);
            Set<String> assignedCourses = new HashSet<>();
            int coursesAssigned = 0;

            while (coursesAssigned < coursesToAssign) {
                String courseId =
courseIds.get(random.nextInt(courseIds.size()));
                System.out.println("courseId is "+courseId);
                System.out.println(coursesAssigned+"/"+coursesToAssign);
                // Check if the course already has 300 students
                rs = stmt.executeQuery("SELECT COUNT(*) AS student_count FROM
gauss_db.SC WHERE C# = '" + courseId + "'");
                rs.next();
                int studentCount = rs.getInt("student_count");
                rs.close();

                if (studentCount < 300 &&
!assignedCourses.contains(courseId)) {
                    double grade = generateGrade(random);
                    try {
                        stmt.executeUpdate("INSERT INTO gauss_db.SC (S#, C#,
GRADE) VALUES "

```



```

// Database connection parameters
String url = "jdbc:postgresql://192.168.3.28:5432/mydb";
String user = "java";
String password = "xyx2003.";

Connection conn = null;
try {
    // Load the PostgreSQL driver (optional step in modern JDBC)
    Class.forName("org.postgresql.Driver");

    // Get a connection to the database
    conn = DriverManager.getConnection(url, user, password);
    System.out.println("Connection established successfully!");

    // Assign courses to students
    assignCoursesToStudents(conn);

} catch (SQLException e) {
    System.out.println("SQL Exception: " + e.getMessage());
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    System.out.println("PostgreSQL JDBC Driver not found: " +
e.getMessage());
    e.printStackTrace();
} finally {
    // Close the connection
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            System.out.println("Failed to close the connection: " +
e.getMessage());
        }
    }
}

private static void assignCoursesToStudents(Connection conn) throws
SQLException {
    Statement stmt = null;
    ResultSet rs = null;
    List<String> studentIds = new ArrayList<>();
    List<String> courseIds = new ArrayList<>();
    Random random = new Random();

    try {
        stmt = conn.createStatement();
        // Get all student IDs
        rs = stmt.executeQuery("SELECT S# FROM gauss_db.S");
        while (rs.next()) {
            studentIds.add(rs.getString("S#"));
        }
        rs.close();

        // Get all course IDs

```

```

rs = stmt.executeQuery("SELECT C# FROM gauss_db.C");
while (rs.next()) {
    courseIds.add(rs.getString("C#"));
}
rs.close();

// For each student, assign courses
for (String studentId : studentIds) {
    // Check if the student already has course records
    rs = stmt.executeQuery("SELECT COUNT(*) AS course_count FROM
gauss_db.SC WHERE S# = '" + studentId + "'");
    rs.next();
    int existingCourses = rs.getInt("course_count");
    rs.close();

    int coursesToAssign = existingCourses > 0 ? 20 : 40;
    System.err.println("course="+ coursesToAssign);
    Set<String> assignedCourses = new HashSet<>();
    int coursesAssigned = 0;

    while (coursesAssigned < coursesToAssign) {
        String courseId =
courseIds.get(random.nextInt(courseIds.size()));
        System.out.println("courseId is "+courseId);
        System.out.println(coursesAssigned+"/"+coursesToAssign);
        // Check if the course already has 300 students
        rs = stmt.executeQuery("SELECT COUNT(*) AS student_count FROM
gauss_db.SC WHERE C# = '" + courseId + "'");
        rs.next();
        int studentCount = rs.getInt("student_count");
        rs.close();

        if (studentCount < 300 &&
!assignedCourses.contains(courseId)) {
            double grade = generateGrade(random);
            try {
                stmt.executeUpdate("INSERT INTO gauss_db.SC (S#, C#,
GRADE) VALUES "
                                + "(" + studentId + ", '" + courseId + "',
" + grade + ")");

                assignedCourses.add(courseId);
                coursesAssigned++;
            } catch (SQLException e) {
                if (e.getSQLState().equals("23505")) {
                    // Duplicate key, skip this course and try
another

                    continue;
                } else {
                    throw e;
                }
            }
        }
    }
}

System.out.println("Courses assigned to students successfully!");

```


```

    } finally {
        if (stmt != null) {
            stmt.close();
        }
        if (rs != null) {
            rs.close();
        }
    }
}

private static double generateGrade(Random random) {
    // Generate a grade with a mean of 80
    return 70 + random.nextGaussian() * 10;
}
}

```

### 3.2.5 插入后的展示

名	行
 c	1003
 s	5068
 SC	192886

### 3.2.6 SQL 语句实现，并分析其运行效率

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。

原查询：

```

1  SELECT S.S#, S.SNAME
2  FROM S
3  JOIN SC ON S.S# = SC.S#
4  GROUP BY S.S#, S.SNAME
5  HAVING COUNT(SC.C#) >= 3
6  ORDER BY AVG(SC.GRADE) DESC
7  FETCH FIRST 1 ROW ONLY;

```

信息	摘要	结果 1
SELECT S.S#, S.SNAME FROM S JOIN SC ON S.S# = SC.S# GROUP BY S.S#, S.SNAME HAVING COUNT(SC.C#) >= 3 ORDER BY AVG(SC.GRADE) DESC FETCH FIRST 1 ROW ONLY > OK > 查询时间：0.196s		

优化后的查询：

```

1 WITH FilteredStudents AS (
2     SELECT S.S#, S.SNAME, AVG(SC.GRADE) AS AvgGrade
3     FROM S
4     JOIN SC ON S.S# = SC.S#
5     GROUP BY S.S#, S.SNAME
6     HAVING COUNT(SC.C#) >= 3
7 )
8 SELECT S#, SNAME
9 FROM FilteredStudents
10 ORDER BY AvgGrade DESC
11 FETCH FIRST 1 ROW ONLY;
12

```

信息 摘要 结果 1

```

WITH FilteredStudents AS (
    SELECT S.S#, S.SNAME, AVG(SC.GRADE) AS AvgGrade
    FROM S
    JOIN SC ON S.S# = SC.S#
    GROUP BY S.S#, S.SNAME
    HAVING COUNT(SC.C#) >= 3
)
SELECT S#, SNAME
FROM FilteredStudents
ORDER BY AvgGrade DESC
FETCH FIRST 1 ROW ONLY
> OK
> 查询时间: 0.181s

```

优化后的查询思路:

#### 数据过滤和聚合:

- 使用 `WITH` 子句定义一个名为 `FilteredStudents` 的临时结果集。
- 在

FilteredStudents

中:

- 从表 `S` 和 `SC` 进行连接, 目的是获取学生信息以及他们的课程成绩。
- 对每个学生 (`S.S#` 和 `S.SNAME` 组合) 进行分组。
- 计算每个学生的平均成绩 `AVG(SC.GRADE)`。
- 使用 `HAVING` 子句过滤出至少选修了3门课程的学生 (`HAVING COUNT(SC.C#) >= 3`)。

#### 结果排序和选择:

- 从 `FilteredStudents` 临时结果集中选择学生编号 (`S#`) 和学生姓名 (`SNAME`)。
- 按照平均成绩 (`AvgGrade`) 降序排列学生。
- 使用 `FETCH FIRST 1 ROW ONLY` 限制结果集只返回一行, 即平均成绩最高的那位学生。

#### 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): 使用原查询语句:



```

1 SELECT S.S#, S.SNAME, SC.C#, SC.GRADE
2 FROM S
3 JOIN SC ON S.S# = SC.S#
4 WHERE S.SEX = '女' AND S.S# NOT IN (
5     SELECT S# FROM SC WHERE C# = 'CS-02'
6 );

```

信息	摘要	结果 1
----	----	------

```

SELECT S.S#, S.SNAME, SC.C#, SC.GRADE
FROM S
JOIN SC ON S.S# = SC.S#
WHERE S.SEX = '女' AND S.S# NOT IN (
    SELECT S# FROM SC WHERE C# = 'CS-02'
)
> OK
> 查询时间: 0.231s

```

(2) : 使用修改后的sql语句:

```

1 SELECT S.S#, S.SNAME, SC.C#, SC.GRADE
2 FROM S
3 JOIN SC ON S.S# = SC.S#
4 LEFT JOIN SC SC2 ON S.S# = SC2.S# AND SC2.C# = 'CS-02'
5 WHERE S.SEX = '女' AND SC2.S# IS NULL;

```

信息	摘要	结果 1
----	----	------

```

SELECT S.S#, S.SNAME, SC.C#, SC.GRADE
FROM S
JOIN SC ON S.S# = SC.S#
LEFT JOIN SC SC2 ON S.S# = SC2.S# AND SC2.C# = 'CS-02'
WHERE S.SEX = '女' AND SC2.S# IS NULL
> OK
> 查询时间: 0.147s

```

效率提高原因:

**NOT IN 子查询:** NOT IN 子查询在处理大数据集时效率低, 因为它需要对外层查询的每一行都进行子查询。这种操作在数据量大时会非常耗时。

**LEFT JOIN 和 IS NULL:** 使用 LEFT JOIN 加上 IS NULL 的方式, 数据库可以通过一次性扫描和连接操作来完成查询, 而不是对每一行进行子查询。LEFT JOIN 会首先连接两个表, 并在连接后的结果集中查找 SC2.S# IS NULL 的行。

**减少子查询次数:** 这种方法避免了对每个 S 表记录进行子查询, 减少了查询次数。

### 3.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

原始查询:

```

6  SELECT S1.SNAME
7  FROM S S1
8  WHERE NOT EXISTS (
9      SELECT 1
10     FROM C_CS
11     LEFT JOIN SC ON C_CS.C# = SC.C# AND SC.S# = S1.S#
12     WHERE SC.S# IS NULL
13 )
14 GROUP BY S1.S#;

```

信息	摘要	结果 1
----	----	------

```

WITH C_CS AS (
    SELECT C#
    FROM C
    WHERE C.C# LIKE 'CS-%'
)
SELECT S1.SNAME
FROM S S1
WHERE NOT EXISTS (
    SELECT 1
    FROM C_CS
    LEFT JOIN SC ON C_CS.C# = SC.C# AND SC.S# = S1.S#
    WHERE SC.S# IS NULL
)
GROUP BY S1.S#
> OK
> 查询时间: 0.522s

```

优化后的查询:

```

1  SELECT S1.SNAME
2  FROM S S1
3  JOIN SC ON S1.S# = SC.S#
4  JOIN C ON SC.C# = C.C#
5  WHERE C.C# LIKE 'CS-%'
6  GROUP BY S1.S#, S1.SNAME
7  HAVING COUNT(DISTINCT C.C#) = (SELECT COUNT(*) FROM C WHERE C.C# LIKE 'CS-%');
8  |

```

信息	摘要	结果 1
----	----	------

```

SELECT S1.SNAME
FROM S S1
JOIN SC ON S1.S# = SC.S#
JOIN C ON SC.C# = C.C#
WHERE C.C# LIKE 'CS-%'
GROUP BY S1.S#, S1.SNAME
HAVING COUNT(DISTINCT C.C#) = (SELECT COUNT(*) FROM C WHERE C.C# LIKE 'CS-%')
> OK
> 查询时间: 0.034s

```

分析:

**JOIN 替换 LEFT JOIN 和 NOT EXISTS:**

- 通过内连接 JOIN 替代 LEFT JOIN 和 NOT EXISTS, 确保只查询存在的匹配记录。
- 这样做减少了数据库引擎处理左连接的开销。

HAVING COUNT：

- 使用 HAVING 子句来进行分组计数并对比，这样可以一次性获取符合条件的学生。
- COUNT(DISTINCT C.C#) 确保每个学生选修的独特课程数量与所有 CS- 开头的课程数量一致。

## 4、数据库的备份

和好友申程宇预先协商，都通过openGauss提供的gs\_dumpall方式备份各自数据库，并发送对方进行恢复。

### 进行数据库的备份

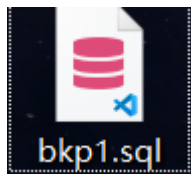
```
mkdir -p /opt/software/openGauss/backup

/opt/software/openGauss/bin/gs_dumpall -f /opt/software/openGauss/backup/bkp1.sql
```

创建backup目录，然后使用gs\_dumpall方式对数据库进行备份，保存为bkp1.sql文件。

```
omm@localhost root]$ mkdir -p /opt/software/openGauss/backup
omm@localhost root]$ /opt/software/openGauss/bin/gs_dumpall -f /opt/software/openGauss/backup/bkp1.sql
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: Begin scanning database.
Progress: [=====] 100% (38/37, cur_step/total_step). finish scanning database
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: Finish scanning database.
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: Start dumping objects
Progress: [=====] 100% (4876/4876, dumpObjNums/totalObjNums). dump objects
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: Finish dumping objects
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: dump database dbname= finance' successfully
js_dump[port=5432][dbname= finance'] [2024-06-19 15:50:40]: total time: 406 ms
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:40]: Begin scanning database.
Progress: [=====] 100% (38/37, cur_step/total_step). finish scanning database
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:40]: Finish scanning database.
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:41]: Start dumping objects
Progress: [=====] 100% (4875/4875, dumpObjNums/totalObjNums). dump objects
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:41]: Finish dumping objects
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:41]: dump database dbname= mydb' successfully
js_dump[port=5432][dbname= mydb'] [2024-06-19 15:50:41]: total time: 561 ms
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: Begin scanning database.
Progress: [=====] 100% (38/37, cur_step/total_step). finish scanning database
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: Finish scanning database.
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: Start dumping objects
Progress: [=====] 100% (4828/4828, dumpObjNums/totalObjNums). dump objects
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: Finish dumping objects
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: dump database dbname= postgres' successfully
js_dump[port=5432][dbname= postgres'] [2024-06-19 15:50:41]: total time: 299 ms
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:41]: Begin scanning database.
Progress: [=====] 100% (38/37, cur_step/total_step). finish scanning database
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:41]: Finish scanning database.
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:41]: Start dumping objects
Progress: [=====] 100% (4862/4862, dumpObjNums/totalObjNums). dump objects
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:41]: Finish dumping objects
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:42]: dump database dbname= school' successfully
js_dump[port=5432][dbname= school'] [2024-06-19 15:50:42]: total time: 365 ms
js_dumpall[port=5432][2024-06-19 15:50:42]: dumpall operation successful
js_dumpall[port=5432][2024-06-19 15:50:42]: total time: 1952 ms
omm@localhost root]$ firefox
```

保存的文件：



### 对搭档的数据库进行恢复

```
gsq1 -d postgres -p 5432 -f /opt/software/openGauss/backup/bkp_scy.sql
```

gsq1：这是 OpenGauss 数据库系统的命令行客户端工具，用于连接和操作数据库。

-d postgres：

- `-d` 选项指定要连接的数据库名称。
- `postgres` 是要连接的数据库名称。OpenGauss 安装中，默认数据库名称为 `postgres`。

`-p 5432` :

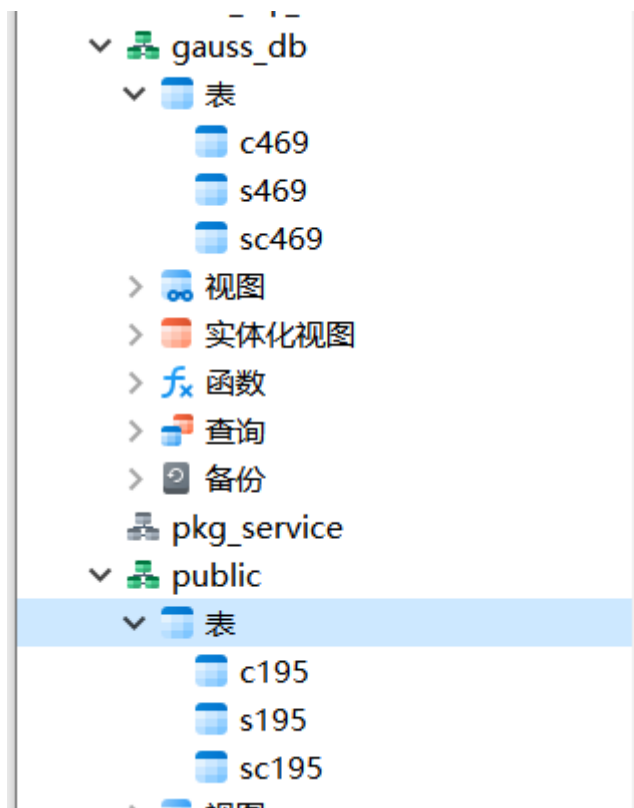
- `-p` 选项指定数据库服务器监听的端口号。
- `5432` 设置端口号为5432

`-f /opt/software/openGauss/backup/bkp_scy.sql` :

- `-f` 选项指定包含 SQL 命令的文件路径。
- `/opt/software/openGauss/backup/bkp_scy.sql` 保存的搭档的.sql文件的路径。

```
omm@localhost root]$ gsql -d postgres -p 5432 -f /opt/software/openGauss/backup/bkp_scy.sql
SET
SET
gsql:/opt/software/openGauss/backup/bkp_scy.sql:12: ERROR:  role "omm" already exists
gsql:/opt/software/openGauss/backup/bkp_scy.sql:13: ERROR:  Permission denied to change privilege of the initial account.
gsql:/opt/software/openGauss/backup/bkp_scy.sql:14: NOTICE:  Using encrypted password directly now and it is not recommended.
HINT:  The role can't be used if you don't know the original password before encrypted.
CREATE ROLE
ALTER ROLE
gsql:/opt/software/openGauss/backup/bkp_scy.sql:26: ERROR:  database "mydb" already exists
REVOKE
REVOKE
GRANT
GRANT
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "mydb" as user "omm".
SET
SET
SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
CREATE VIEW
ALTER VIEW
CREATE VIEW
ALTER VIEW
CREATE VIEW
ALTER VIEW
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
REVOKE
REVOKE
GRANT
GRANT
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "postgres" as user "omm".
SET
SET
SET
SET
SET
SET
SET
COMMENT
CREATE SCHEMA
ALTER SCHEMA
REVOKE
REVOKE
```

恢复效果展示



其中gauss\_db模式下为原本自己的3个表， public下为搭档的3个表。

搭档表的合理性及数据质量

表的设计

名	类型	长度	小数点	不是 null	键	注释
▶ c#	char	10	0	<input checked="" type="checkbox"/>	🔑 1	
cname	varchar	30	0	<input type="checkbox"/>		
period	int4	32	0	<input type="checkbox"/>		
credit	numeric	3	1	<input type="checkbox"/>		
teacher	varchar	30	0	<input type="checkbox"/>		

名	类型	长度	小数点	不是 null	键
▶ s#	char	10	0	<input checked="" type="checkbox"/>	🔑 1
sname	varchar	30	0	<input type="checkbox"/>	
sex	char	3	0	<input type="checkbox"/>	
bdate	timestamp	0	0	<input type="checkbox"/>	
height	numeric	3	2	<input type="checkbox"/>	
dorm	varchar	20	0	<input type="checkbox"/>	

名	类型	长度	小数点	不是 null	键
▶ s#	char	10	0	<input checked="" type="checkbox"/>	🔑 1
c#	char	10	0	<input checked="" type="checkbox"/>	🔑 2
grade	numeric	4	1	<input type="checkbox"/>	

搭档表中的每个条目的数据类型设计基本合理。

例如C表中的cname，虽然当前表内所有名字不会超过9个byte(3个汉字)，但是考虑到可能有少数民族或英文名等较长的人名，设置为varchar[30]合理。

### 可以优化的地方

C表中的period学时属性可以设置为int2，16字节，可以表示 $\pm 2^{15}$ ，约为 $\pm 32000$ ，表示课时已经足够了。

C表中的c#使用了char[10]，课程编号格式都为"XX-YY"，其中X和Y都可以用ASCII编码表示，所以理论上只需要char[5]就能表示所有的课程号，这里这样设计可能是预留几位提高鲁棒性。

## 数据分析

C表：

c#	cname	period	credit	teacher
SO-06	并行计算	40	4.0	张勇
SE-69	软件工程	48	3.0	王静
SO-77	微积分	32	5.0	陈建明
FI-99	网络安全	60	3.5	王芳
BI-63	网络编程	72	3.5	刘强
AC-63	线性代数	72	5.0	李敏
SE-56	经济学	120	3.0	石韬
AC-96	移动通信	80	4.5	李敏
SO-33	软件工程	72	3.5	李明
ED-22	计算机网络	64	4.5	李勇
BA-25	神经网络	90	3.5	张杰
PH-78	网络安全	40	2.0	张娟
ME-13	嵌入式系统	100	2.5	李蕾
SE-06	图形学	90	5.0	李敏
BA-93	软件测试	50	4.5	李洋
ED-75	化学	50	3.0	胡海东
CE-32	机器学习	90	4.0	张杰
EE-90	机器学习	90	3.5	王丽
CH-99	离散数学	80	2.0	李洋
CH-04	软件工程	80	4.5	张明
HR-46	数据挖掘	90	5.0	王静
CE-85	大数据	90	4.5	石韬
ME-09	机器视觉	120	4.5	李华
FI-54	网络安全	60	2.0	王军
AE-74	大数据	32	3.5	张明
SE-96	模式识别	72	4.0	王亚伟
CS-22	云计算	60	2.0	王明
SO-41	模式识别	64	5.0	李杰
FI-47	图形学	32	3.5	李伟
CE-56	并行计算	48	2.5	王娟
PS-41	物联网	90	4.5	张涛
MM-48	线性代数	60	5.0	李勇
CH-82	深度学习	72	4.5	李艳
SE-84	网络安全	120	4.0	陈建明
EE-07	微机原理	40	4.0	张娟
CE-74	嵌入式系统设计	50	4.0	李洋
ED-56	操作系统	100	3.0	陈建明
BI-41	数据结构	60	3.5	李明
ED-84	数据挖掘	30	3.5	李敏
SC-02	微积分	30	2.5	李丽
FI-50	光电子学与光子学	40	3.5	胡海东
CH-06	生物学	100	3.5	崔昀
MM-90	网络安全	50	4.5	张娟
SC-04	哲学	48	3.0	张娟

优点：

课时period分布均匀。课程名称规整，一名老师可以代多门课，符合实际。

缺点：

课时与学分分布不合理。大部分情况下学分与课时数应该成一定程度的正相关，这里出现了高学时低学分的情况，有些不合理

SE-56	经济学	120	3.0	石韬
BA-93	软件测试	50	4.5	李洋

此外课程名复用的情况较多，一般情况下是可能有多门不同的课课程名相同，但这种情况较少。



S表：

s#	sname	sex	bdate	height	dorm
03039883	林蓝蓝	男	2004-04-12 00:00:00	1.67	西8舍806
06039852	刘绍伟	女	2001-04-07 00:00:00	1.66	西2舍412
05038522	宋晓婷	男	2005-08-09 00:00:00	1.70	西19舍829
04039643	卢石生	男	2000-04-12 00:00:00	1.76	西9舍423
02030906	余晓东	男	2002-11-09 00:00:00	1.76	西9舍421
02039112	徐卫云	女	2002-06-19 00:00:00	1.72	西2舍830
06039155	廖世亮	男	2004-05-09 00:00:00	1.76	西19舍302
07031545	刘林春	男	2001-04-29 00:00:00	1.73	西19舍524
08032804	李德渠	女	2002-04-22 00:00:00	1.60	西2舍429
03031494	何玉海	男	2000-02-24 00:00:00	1.77	西6舍505
03035550	祁兴华	男	2001-09-25 00:00:00	1.72	西10舍706
03032075	游乐天	男	2002-02-27 00:00:00	1.80	西19舍428
03031893	韦志宏	男	2004-10-25 00:00:00	1.81	西19舍329
04033655	瞿研	男	2001-10-19 00:00:00	1.68	西16舍314
01031760	宋文钦	男	2004-01-13 00:00:00	1.74	西9舍603
07033758	封源	男	2004-08-21 00:00:00	1.78	西18舍827
06032218	曾为金	男	2001-06-08 00:00:00	1.66	西19舍230
02034142	任晓培	男	2003-07-07 00:00:00	1.77	西9舍309
05030814	徐力中	男	2003-10-18 00:00:00	1.75	西18舍616
06030157	于善	男	2005-04-29 00:00:00	1.75	西6舍819
04035893	邓利军	女	2000-02-03 00:00:00	1.60	西12舍409
07030632	付林森	女	2001-11-29 00:00:00	1.69	西12舍729
05030119	林维坚	男	2001-04-08 00:00:00	1.87	西17舍109
06036103	黄华新	女	2002-02-03 00:00:00	1.66	西11舍708
05039329	徐兆亮	男	2002-04-17 00:00:00	1.81	西6舍502
05035253	福敏	男	2005-11-14 00:00:00	1.76	西20舍325
03037626	高广路	女	2002-09-26 00:00:00	1.62	西11舍321
05031718	兰念	男	2004-08-09 00:00:00	1.75	西6舍506
06031353	闵海涛	男	2003-08-19 00:00:00	1.75	西9舍728
06035503	蒋世军	女	2000-06-15 00:00:00	1.63	西3舍724
05031447	李之云	男	2003-06-09 00:00:00	1.82	西19舍229
04037024	许肯	男	2002-02-06 00:00:00	1.78	西18舍724
06033998	王克龙	女	2000-02-24 00:00:00	1.67	西14舍215
07032242	柳亚军	男	2004-04-05 00:00:00	1.71	西9舍425
02037912	卢梦玲	女	2004-06-21 00:00:00	1.71	西5舍815
05032067	杜天泽	男	2000-09-25 00:00:00	1.78	西8舍707
03039294	美璇	女	2004-05-16 00:00:00	1.68	西3舍503
08037888	牛燕燕	男	2004-10-15 00:00:00	1.74	西18舍322
05035301	曾玲	男	2004-08-25 00:00:00	1.70	西8舍802
01037290	陈庆玲	女	2003-01-07 00:00:00	1.60	西1舍321
03034820	单薇	女	2001-01-07 00:00:00	1.60	西2舍529
04035339	龙娜	男	2004-06-17 00:00:00	1.75	西10舍203

表中的各项数据均合理。

学号设置初始位为0，长度与一开始提供的已有的学号的学号相同都是8位，且第一位都为0。

姓名合理，符合人名规范。

出生日期合理，基本都是2000-2005出生，符合当前在校大学生年纪。

身高合理。女生平均身高低于男生平均身高，平均身高合理，身高分布合理，推测是使用的正态分布。

宿舍分配合理，男女住在不同的宿舍楼。

SC表

s#	c#	grade
04035878	AC-32	73.5
02034479	SO-95	78.0
03034147	PS-45	91.5
05031110	CS-54	81.0
05037791	HR-08	92.5
02037938	AR-47	83.5
03033998	BA-42	88.0
05031056	MA-22	81.5
01033946	AR-38	67.0
05037647	ED-08	99.0
01032862	HR-66	91.5
06031355	SC-40	97.5
03033442	HR-64	64.5
04039391	AE-32	97.0
04037950	FI-54	73.0
04032943	ED-36	75.0
08030021	MM-36	62.5
08030059	FI-04	87.0
07035464	CH-98	71.0
02034677	HR-23	55.0
03034167	AR-24	(Null)
03037217	AE-42	64.0
01036706	EE-65	99.0
02032282	CH-44	73.5
04031633	MA-15	68.5
08031867	SC-88	100.0
02036404	PS-45	98.5
05033834	BI-89	76.5
06036789	BI-75	82.0
07033628	MA-74	(Null)
01037784	HR-82	100.0
07031785	AC-22	98.0
08036751	CE-50	75.5
06034432	HR-64	73.5
08037549	PH-65	79.5
08030055	PH-48	60.5
06032956	CH-89	75.0
06036451	BA-36	96.0
01039163	PS-20	82.5
02039928	CH-08	84.0
06039980	AR-21	98.0
03039778	ED-96	90.5
01030893	CE-92	100.0

成绩合理，最多保留到0.5符合实际。

## 5、实验心得

---

在本次数据库实验中，我完成了在本地安装OpenGauss数据库，并进行了相关数据管理的操作。首先，通过详细的文档和指南，我成功地在本地环境中安装并配置了OpenGauss数据库。这一步骤让我深入了解了数据库安装的各个环节，以及如何配置数据库参数以优化性能。随后，我使用Python编写了爬虫程序，从指定的网站爬取数据，并将这些数据存储到OpenGauss数据库的表中。这个过程让我掌握了爬虫技术和数据清洗的基本方法，以及使用JDBC将外部数据导入数据库。在实验中，我还设计了多种SQL查询，进一步增强了对SQL语法和数据库操作的理解。总体而言，这次实验不仅提升了我对数据库技术的实际操作能力，也加深了对数据管理和处理的认识，为以后的数据相关工作打下了坚实的基础。