《数据库系统》课内实验

一、在 openGauss 中创建 MYDB 数据库，并在 MYDB 中创建学生、课程、选课三个表。

各表包含属性如下：

S500（S#，SNAME，SEX，BDATE，HEIGHT，DORM）

C500（C#，CNAME，PERIOD，CREDIT，TEACHER）

SC500（S#，C#，GRADE）其中 S#、C#均为外键

①创建 S500 表：

```
my_db=> CREATE TABLE S500 (
my_db(>     SNO VARCHAR ( 8 ) NOT NULL,
my_db(>     SNAME VARCHAR ( 15 ) NOT NULL,
my_db(>     SEX VARCHAR ( 3 ) NOT NULL,
my_db(>     BDATE DATE NOT NULL,
my_db(>     HEIGHT DEC ( 3, 2 ) NOT NULL,
my_db(>     DORM VARCHAR ( 20 ) NOT NULL,
my_db(> PRIMARY KEY ( SNO )
my_db(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "s500_pkey" for table "s500"
CREATE TABLE
my_db=> \d+ S500
                                    Table "public.s500"
 Column |             Type             | Modifiers | Storage  | Stats target | Description
--------+------------------------------+-----------+----------+--------------+-------------
 sno    | character varying(8)         | not null  | extended |              |
 sname  | character varying(15)        | not null  | extended |              |
 sex    | character varying(3)         | not null  | extended |              |
 bdate  | timestamp(0) without time zone | not null  | plain    |              |
 height | numeric(3,2)                 | not null  | main     |              |
 dorm   | character varying(20)        | not null  | extended |              |
Indexes:
    "s500_pkey" PRIMARY KEY, btree (sno) TABLESPACE pg_default
Has OIDs: no
Options: orientation=row, compression=no
```

②创建 C500 表：

```
my_db=> CREATE TABLE C500 (
my_db(>     CNO VARCHAR ( 10 ) NOT NULL,
my_db(>     CNAME VARCHAR ( 50 ) NOT NULL,
my_db(>     PERIOD INT NOT NULL,
my_db(>     CREDIT DEC ( 3, 1 ) NOT NULL,
my_db(>     TEACHER VARCHAR ( 15 ) NOT NULL,
my_db(> PRIMARY KEY ( CNO )
my_db(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "c500_pkey" for table "c500"
CREATE TABLE
my_db=> \d+ C500
                                Table "public.c500"
 Column  |         Type          | Modifiers | Storage  | Stats target | Description
---------+-----------------------+-----------+----------+--------------+-------------
 cno     | character varying(10) | not null  | extended |              |
 cname   | character varying(50) | not null  | extended |              |
 period  | integer               | not null  | plain    |              |
 credit  | numeric(3,1)          | not null  | main     |              |
 teacher | character varying(15) | not null  | extended |              |
Indexes:
    "c500_pkey" PRIMARY KEY, btree (cno) TABLESPACE pg_default
Has OIDs: no
Options: orientation=row, compression=no
```

③创建 SC500 表：

```
my_db=> CREATE TABLE SC500 (
my_db(>     SNO VARCHAR ( 8 ) NOT NULL,
my_db(>     CNO VARCHAR ( 10 ) NOT NULL,
my_db(>     GRADE DEC ( 4, 1 ) DEFAULT NULL,
my_db(>     PRIMARY KEY ( SNO, CNO ),
my_db(>     FOREIGN KEY ( SNO ) REFERENCES S500 ON DELETE CASCADE,
my_db(>     FOREIGN KEY ( CNO ) REFERENCES C500 ON DELETE RESTRICT,
my_db(> CHECK ( ( GRADE IS NULL ) OR ( GRADE BETWEEN 0 AND 100 ) )
my_db(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "sc500_pkey" for table "sc500"
CREATE TABLE
my_db=> \d+ SC500
                                    Table "public.sc500"
 Column |         Type          |       Modifiers       | Storage  | Stats target | Description
--------+-----------------------+-----------------------+----------+--------------+-------------
 sno    | character varying(8)  | not null              | extended |              |
 cno    | character varying(10) | not null              | extended |              |
 grade  | numeric(4,1)          | default NULL::numeric | main     |              |
Indexes:
    "sc500_pkey" PRIMARY KEY, btree (sno, cno) TABLESPACE pg_default
Check constraints:
    "sc500_grade_check" CHECK (grade IS NULL OR grade >= 0::numeric AND grade <= 100::numeric)
Foreign-key constraints:
    "sc500_cno_fkey" FOREIGN KEY (cno) REFERENCES c500(cno) ON DELETE RESTRICT
    "sc500_sno_fkey" FOREIGN KEY (sno) REFERENCES s500(sno) ON DELETE CASCADE
Has OIDs: no
Options: orientation=row, compression=no
```

二、将数据加入相应的表中。

①将数据加入 S500 表：

```
my_db=> INSERT INTO S500 ( SNO, SNAME, SEX, BDATE, HEIGHT, DORM )
my_db-> VALUES
my_db->     ( '01032010', '王涛', '男', '2002-4-5', 1.72, '东14舍221' ),
my_db->     ( '01032023', '孙文', '男', '2003-6-10', 1.80, '东14舍221' ),
my_db->     ( '01032001', '张晓梅', '女', '2003-11-17', 1.58, '东1舍312' ),
my_db->     ( '01032005', '刘静', '女', '2002-1-10', 1.63, '东1舍312' ),
my_db->     ( '01032112', '董蔚', '男', '2002-2-20', 1.71, '东14舍221' ),
my_db->     ( '03031011', '王倩', '女', '2003-12-20', 1.66, '东2舍104' ),
my_db->     ( '03031014', '赵思扬', '男', '2001-6-6', 1.85, '东18舍421' ),
my_db->     ( '03031051', '周剑', '男', '2001-5-8', 1.68, '东18舍422' ),
my_db->     ( '03031009', '田婷', '女', '2002-8-11', 1.60, '东2舍104' ),
my_db->     ( '03031033', '蔡明明', '男', '2002-3-12', 1.75, '东18舍423' ),
my_db->     ( '03031056', '曹子衿', '女', '2003-12-15', 1.65, '东2舍305' );
INSERT 0 11
my_db=> SELECT * FROM S500;
   sno    | sname  | sex |        bdate        | height |   dorm
----------+--------+-----+---------------------+--------+-----------
 01032010 | 王涛   | 男  | 2002-04-05 00:00:00 |   1.72 | 东14舍221
 01032023 | 孙文   | 男  | 2003-06-10 00:00:00 |   1.80 | 东14舍221
 01032001 | 张晓梅 | 女  | 2003-11-17 00:00:00 |   1.58 | 东1舍312
 01032005 | 刘静   | 女  | 2002-01-10 00:00:00 |   1.63 | 东1舍312
 01032112 | 董蔚   | 男  | 2002-02-20 00:00:00 |   1.71 | 东14舍221
 03031011 | 王倩   | 女  | 2003-12-20 00:00:00 |   1.66 | 东2舍104
 03031014 | 赵思扬 | 男  | 2001-06-06 00:00:00 |   1.85 | 东18舍421
 03031051 | 周剑   | 男  | 2001-05-08 00:00:00 |   1.68 | 东18舍422
 03031009 | 田婷   | 女  | 2002-08-11 00:00:00 |   1.60 | 东2舍104
 03031033 | 蔡明明 | 男  | 2002-03-12 00:00:00 |   1.75 | 东18舍423
 03031056 | 曹子衿 | 女  | 2003-12-15 00:00:00 |   1.65 | 东2舍305
(11 rows)
```

②将数据加入 C500 表：

```
my_db=> INSERT INTO C500 ( CNO, CNAME, PERIOD, CREDIT, TEACHER )
my_db-> VALUES
my_db->     ( 'CS-01', '数据结构', 60, 3, '张军' ),
my_db->     ( 'CS-02', '计算机组成原理', 80, 4, '王亚伟' ),
my_db->     ( 'CS-04', '人工智能', 40, 2, '李蕾' ),
my_db->     ( 'CS-05', '深度学习', 40, 2, '崔均' ),
my_db->     ( 'EE-01', '信号与系统', 60, 3, '张明' ),
my_db->     ( 'EE-02', '数字逻辑电路', 100, 5, '胡海东' ),
my_db->     ( 'EE-03', '光电子学与光子学', 40, 2, '石韬' );
INSERT 0 7
my_db=> SELECT * FROM C500;
  cno  |      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 | 崔均
 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
(7 rows)
```

③将数据加入 SC500 表：

```
my_db=> INSERT INTO SC500 ( SNO, CNO, GRADE )
my_db-> VALUES
my_db->     ( '01032010', 'CS-01', 82.0 ),
my_db->     ( '01032010', 'CS-02', 91.0 ),
my_db->     ( '01032010', 'CS-04', 83.5 ),
my_db->     ( '01032001', 'CS-01', 77.5 ),
my_db->     ( '01032001', 'CS-02', 85.0 ),
my_db->     ( '01032001', 'CS-04', 83.0 ),
my_db->     ( '01032005', 'CS-01', 62.0 ),
my_db->     ( '01032005', 'CS-02', 77.0 ),
my_db->     ( '01032005', 'CS-04', 82.0 ),
my_db->     ( '01032023', 'CS-01', 55.0 ),
my_db->     ( '01032023', 'CS-02', 81.0 ),
my_db->     ( '01032023', 'CS-04', 76.0 ),
my_db->     ( '01032112', 'CS-01', 88.0 ),
my_db->     ( '01032112', 'CS-02', 91.5 ),
my_db->     ( '01032112', 'CS-04', 86.0 ),
my_db->     ( '01032112', 'CS-05', NULL ),
my_db->     ( '03031033', 'EE-01', 93.0 ),
my_db->     ( '03031033', 'EE-02', 89.0 ),
my_db->     ( '03031009', 'EE-01', 88.0 ),
my_db->     ( '03031009', 'EE-02', 78.5 ),
my_db->     ( '03031011', 'EE-01', 91.0 ),
my_db->     ( '03031011', 'EE-02', 86.0 ),
my_db->     ( '03031051', 'EE-01', 78.0 ),
my_db->     ( '03031051', 'EE-02', 58.0 ),
my_db->     ( '03031014', 'EE-01', 79.0 ),
my_db->     ( '03031014', 'EE-02', 71.0 );
INSERT 0 26
```

```
my_db=> SELECT * FROM SC500;
   sno     |  cno   | 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 |
 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
(26 rows)
```

三、完成以下操作，将相应 SQL 语句及其执行结果截屏保存，并写入实验报告中。

1．在上述基本表上完成以下查询：

(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。

SQL 语句解释：通过 LIKE 语句和通配符%查询以“EE-”开头的课程编号。

```
my_db=> SELECT
my_db->     CNO,
my_db->     CNAME,
my_db->     CREDIT
my_db-> FROM
my_db->     C500
my_db-> WHERE
my_db->     CNO LIKE 'EE-%';
  cno   |       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
(3 rows)
```

(2) 查询未选修课程“CS-01”的女生学号及其已选各课程编号、成绩。

SQL 语句解释：先通过 SELECT 子查询获得选修课程“CS-01”的学生学号，再使用 NOT IN 语句得到最终结果。

```
my_db=> SELECT
my_db->     SC500.SNO,
my_db->     CNO,
my_db->     GRADE
my_db-> FROM
my_db->     SC500,
my_db->     S500
my_db-> WHERE
my_db->     SC500.SNO = S500.SNO
my_db->     AND SEX = '女'
my_db->     AND SC500.SNO NOT IN ( SELECT SNO FROM SC500 WHERE CNO = 'CS-01' );
   sno     |  cno   | grade
-----------+--------+-------
 03031009 | EE-01 |  88.0
 03031009 | EE-02 |  78.5
 03031011 | EE-01 |  91.0
 03031011 | EE-02 |  86.0
(4 rows)
```

(3) 查询 2001 年～2002 年出生学生的基本信息。

SQL 语句解释：使用 BETWEEN AND 语句确定出生日期范围。

```
my_db=> SELECT
my_db->     *
my_db-> FROM
my_db->     S500
my_db-> WHERE
my_db->     BDATE BETWEEN '2001-01-01' AND '2001-12-31';
   sno     | sname  | sex |        bdate        | height |   dorm
-----------+--------+-----+---------------------+--------+-----------
 03031014 | 赵思扬 | 男  | 2001-06-06 00:00:00 |   1.85 | 东18舍421
 03031051 | 周剑   | 男  | 2001-05-08 00:00:00 |   1.68 | 东18舍422
(2 rows)
```

(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。

SQL 语句解释：学生只有在成绩及格后才能获得学分，因此使用 SUM 函数统计已选修课程的学分总数时需加上条件判断。此外，S500 表中的部分学生未选修任何课程，在 SC500 表中无相应记录，因此不能使用等值连接，而应使用外连接将两张表连接起来。

```
my_db=> SELECT
my_db->     S500.SNO,
my_db->     S500.SNAME,
my_db->     SUM (CASE WHEN COALESCE (GRADE, 0) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END)
my_db->     AS SUM_CREDIT
my_db-> FROM
my_db->     S500
my_db->     LEFT JOIN SC500 ON S500.SNO = SC500.SNO
my_db->     LEFT JOIN C500 ON C500.CNO = SC500.CNO
my_db-> GROUP BY
my_db->     S500.SNO,
my_db->     S500.SNAME;
   sno     | sname  | sum_credit
-----------+--------+------------
 03031051 | 周剑   |        3.0
 01032001 | 张晓梅 |        9.0
 01032005 | 刘静   |        9.0
 03031011 | 王倩   |        8.0
 03031009 | 田婷   |        8.0
 01032112 | 董蔚   |        9.0
 03031056 | 曹子衿 |          0
 01032010 | 王涛   |        9.0
 03031014 | 赵思扬 |        8.0
 01032023 | 孙文   |        6.0
 03031033 | 蔡明明 |        8.0
(11 rows)
```

(5) 查询选修课程“CS-02”的学生中成绩第二高的学生学号。

SQL 语句解释：先通过 SELECT 子查询获得最高成绩，再使用 NOT IN 语句查询第二高的成绩，最终查询取得该成绩的学生学号。之所以这么做是因为可能存在多名学生成绩相同，不能将学生按照成绩高低排序后取第二个。

```
my_db=> SELECT
my_db->     SNO
my_db-> FROM
my_db->     SC500
my_db-> WHERE
my_db->     CNO = 'CS-02'
my_db->     AND GRADE = (
my_db(>         SELECT
my_db(>             MAX ( GRADE )
my_db(>         FROM
my_db(>             SC500
my_db(>         WHERE
my_db(>             CNO = 'CS-02'
my_db(>             AND GRADE NOT IN ( SELECT MAX ( GRADE ) FROM SC500 WHERE CNO = 'CS-02' )
my_db(>     );
   sno
----------
 01032010
(1 row)
```

(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。

SQL 语句解释：先通过 SELECT 子查询获得“王涛”同学的平均成绩，这里大于号后面的 ALL 是考虑到可能有重名的情况导致子查询结果有多个。

```
my_db=> SELECT
my_db->     S500.SNO,
my_db->     SNAME,
my_db->     AVG ( GRADE ) AS AVG_GRADE
my_db-> FROM
my_db->     S500,
my_db->     SC500
my_db-> WHERE
my_db->     S500.SNO = SC500.SNO
my_db-> GROUP BY
my_db->     S500.SNO,
my_db->     SNAME
my_db-> HAVING
my_db->     AVG ( GRADE ) > ALL (
my_db(>         SELECT
my_db(>             AVG ( GRADE )
my_db(>         FROM
my_db(>             S500,
my_db(>             SC500
my_db(>         WHERE
my_db(>             S500.SNO = SC500.SNO
my_db(>             AND SNAME = '王涛'
my_db(>         GROUP BY
my_db(>             S500.SNO )
my_db-> ORDER BY
my_db->     S500.SNO DESC;
   sno    | sname  |       avg_grade
----------+--------+--------------------
 03031033 | 蔡明明 | 91.0000000000000000
 03031011 | 王倩   | 88.5000000000000000
 01032112 | 董蔚   | 88.5000000000000000
(3 rows)
```

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

SQL 语句解释：与（4）类似，统计学分总数时应考虑成绩是否及格，此外，这里通过 SELECT 子查询获得计算机专业的课程总数，并在 HAVING 子句中判断学生是否选修了计算机专业全部课程。

```
my_db=> SELECT
my_db->     SNAME,
my_db->     SUM (CASE WHEN COALESCE (GRADE, 0) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END)
my_db->     AS SUM_CREDIT
my_db-> FROM
my_db->     S500,
my_db->     C500,
my_db->     SC500
my_db-> WHERE
my_db->     C500.CNO LIKE 'CS-%'
my_db->     AND C500.CNO = SC500.CNO
my_db->     AND S500.SNO = SC500.SNO
my_db-> GROUP BY
my_db->     S500.SNO,
my_db->     SNAME
my_db-> HAVING
my_db->     COUNT ( * ) = ( SELECT COUNT ( CNO ) FROM C500 WHERE CNO LIKE 'CS-%' );
 sname | sum_credit
-------+------------
 董蔚  |        9.0
(1 row)
```

(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。

SQL 语句解释：先通过 SELECT 子查询获得选修了 3 门以上课程（包括 3 门）的所有学生的平均成绩，再通过>=ALL 查询其中平均成绩最高的同学。

```
my_db=> SELECT
my_db->     S500.SNO,
my_db->     SNAME
my_db-> FROM
my_db->     S500,
my_db->     SC500
my_db-> WHERE
my_db->     S500.SNO = SC500.SNO
my_db-> GROUP BY
my_db->     S500.SNO,
my_db->     SNAME
my_db-> HAVING
my_db->     COUNT ( * ) >= 3
my_db->     AND AVG ( GRADE ) >= ALL (
my_db(>         SELECT
my_db(>             AVG ( GRADE )
my_db(>         FROM
my_db(>             SC500
my_db(>         GROUP BY
my_db(>             SNO
my_db(>         HAVING
my_db(>             COUNT ( * ) >= 3 );
   sno    | sname
----------+-------
 01032112 | 董蔚
(1 row)
```

2．分别在 S500 和 C500 表中加入记录(‘01032005’，‘刘竞’，‘男’，‘2003-12-10’，1.75，‘东 14 舍 312’)及(‘CS-03’，“离散数学”，64，4，‘陈建明’)。

①由于主键已经存在，故插入失败。

```
my_db=> INSERT INTO S500 ( SNO, SNAME, SEX, BDATE, HEIGHT, DORM )
my_db-> VALUES
my_db->     ( '01032005', '刘竞', '男', '2003-12-10', 1.75, '东14舍312' );
ERROR:  duplicate key value violates unique constraint "s500_pkey"
DETAIL:  Key (sno)=(01032005) already exists.
```

②成功在 C500 表中插入一条“CS-03”的记录。

```
my_db=> INSERT INTO C500 ( CNO, CNAME, PERIOD, CREDIT, TEACHER )
my_db-> VALUES
my_db->     ( 'CS-03', '离散数学', 64, 4, '陈建明' );
INSERT 0 1
my_db=> SELECT * FROM C500;
  cno   |      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 | 崔均
 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
 CS-03 | 离散数学          |     64 |    4.0 | 陈建明
(8 rows)
```

3．将 S500 表中已修学分数大于 60 的学生记录删除。

SQL 语句解释：先通过 SELECT 子查询获得已修学分数大于 60 的学生学号，再通过 IN 语句判断 S500 表中是否存在这样的学号。由于 S500 表中并没有满足条件的记录，因此未删除任何数据。

```
my_db=> DELETE
my_db-> FROM
my_db->     S500
my_db-> WHERE
my_db->     SNO IN (
my_db(>         SELECT
my_db(>             SNO
my_db(>         FROM
my_db(>             SC500,
my_db(>             C500
my_db(>         WHERE
my_db(>             C500.CNO = SC500.CNO
my_db(>         GROUP BY
my_db(>             SNO
my_db(>         HAVING
my_db(>             SUM ( CREDIT ) > 60 );
DELETE 0
```

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。

```
my_db=> UPDATE
my_db->     C500
my_db-> SET
my_db->     PERIOD = 64,
my_db->     CREDIT = CREDIT + 1
my_db-> WHERE
my_db->     CNAME = '信号与系统'
my_db->     AND TEACHER = '张明';
UPDATE 1
my_db=> SELECT * FROM C500;
  cno   |       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 | 崔均
 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
 CS-03  | 离散数学           |     64 |    4.0 | 陈建明
 EE-01  | 信号与系统         |     64 |    4.0 | 张明
(8 rows)
```

5．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。

```
my_db=> CREATE VIEW East_18_Male AS
my_db-> SELECT
my_db->     SNO,
my_db->     SNAME,
my_db->     BDATE,
my_db->     HEIGHT
my_db-> FROM
my_db->     S500
my_db-> WHERE
my_db->     DORM LIKE '东18舍%'
my_db->     AND SEX = '男';
CREATE VIEW
my_db=> SELECT * FROM East_18_Male;
   sno    | sname  |        bdate        | height
----------+--------+---------------------+--------
 03031014 | 赵思扬 | 2001-06-06 00:00:00 |   1.85
 03031051 | 周剑   | 2001-05-08 00:00:00 |   1.68
 03031033 | 蔡明明 | 2002-03-12 00:00:00 |   1.75
(3 rows)
```

(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。

```
my_db=> CREATE VIEW ZhangMing_Course AS
my_db-> SELECT
my_db->     C500.CNO,
my_db->     CNAME,
my_db->     AVG ( GRADE ) AS AVG_GRADE
my_db-> FROM
my_db->     C500,
my_db->     SC500
my_db-> WHERE
my_db->     C500.CNO = SC500.CNO
my_db->     AND TEACHER = '张明'
my_db-> GROUP BY
my_db->     C500.CNO,
my_db->     CNAME;
CREATE VIEW
my_db=> SELECT * FROM ZhangMing_Course;
  cno   |   cname    |       avg_grade
--------+------------+--------------------
 EE-01 | 信号与系统 | 85.8000000000000000
(1 row)
```

(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。

```
my_db=> CREATE VIEW AI_Student AS
my_db-> SELECT
my_db->     S500.SNO,
my_db->     SNAME,
my_db->     GRADE
my_db-> FROM
my_db->     SC500,
my_db->     S500,
my_db->     C500
my_db-> WHERE
my_db->     S500.SNO = SC500.SNO
my_db->     AND C500.CNO = SC500.CNO
my_db->     AND CNAME = '人工智能';
CREATE VIEW
my_db=> SELECT * FROM AI_Student;
   sno    | sname  | grade
----------+--------+-------
 01032010 | 王涛   |  83.5
 01032001 | 张晓梅 |  83.0
 01032005 | 刘静   |  82.0
 01032023 | 孙文   |  76.0
 01032112 | 董蔚   |  86.0
(5 rows)
```

四、完成以下操作，将相应结果截屏保存，并写入实验报告中。

1．在 S500 表中补充数据至约 1000 行，在 C500 表中补充数据至约 100 行，在 SC500 表中补充数据至约 20000 行。在向 SC500 表中补充数据的过程中，随机选择成绩低于 60 分的 200 行选课记录删除。以上过程不得在同一程序中串行完成。

（1）数据生成

①S500 表：由于学号是主键，因此要避免随机生成的学号重复，具体做法是

每次随机生成学号后先判断是否重复，若重复则重新生成，否则将其加入列表中。然后按照男女比例 7：3 生成学生信息，姓名通过 Python 的 Faker 库按照性别随机生成，出生日期限制在 19-23 年前。根据性别不同随机生成不同的身高并分配不同的宿舍，男生身高在 160-190cm，宿舍为东 14-18 舍，女生身高在 150-180cm，宿舍为东 1-5 舍。此外，限制每栋宿舍楼 8 层，每层 25 间，每间 4 人。

②C500 表：使用 Python 爬虫从学校本科教务网站的全校课表查询中获取课程信息，由于获取的课程数据格式与 C500 表不符，因此需要进行数据处理，具体做法是截取课程编号的前两位字母并将后六位数字映射到 01-99，从而组成 C500 表的 CNO 字段值，与 S500 表中的主键处理相同，需要对 CNO 进行去重。学校课程的授课教师可能有多个，这里只取第一位作为课程教师。学时与学分字段与学校课程相同。

③SC500 表：由于 SC500 表中的 SNO 与 CNO 字段既是主键又是外键，因此需要随机从生成的 S500 表与 C500 表中各选取一个 SNO 和 CNO 组合，并利用 Python 的字典避免重复，然后随机生成一个成绩。此外，限制了每名学生最多可选修 100 门课程。

（具体代码见附录部分）

（2）数据插入

将三张表的插入分别在三个 Java 程序中实现，为了在向 SC500 表中补充数据的同时，随机选择成绩低于 60 分的 200 行选课记录删除，使用两个线程对 SC500 表进行更新，一个线程用于插入数据，另一个线程用于删除数据。

（具体代码见附录部分）

```
 02564489 | 康建军 | 男   | 2002-05-14 00:00:00 |   1.70 | 东16舍 317
 02497432 | 平芳   | 男   | 2002-12-20 00:00:00 |   1.73 | 东17舍 611
 01371119 | 刘刚   | 女   | 2002-06-10 00:00:00 |   1.64 | 东1舍 307
 01634104 | 石丽丽 | 女   | 2003-10-10 00:00:00 |   1.52 | 东1舍 518
 02162088 | 黄芳   | 女   | 2002-02-02 00:00:00 |   1.68 | 东5舍 518
 01067322 | 萧雪   | 男   | 2002-06-19 00:00:00 |   1.66 | 东18舍 721
 01277960 | 徐建   | 男   | 2004-03-02 00:00:00 |   1.62 | 东17舍 406
 02461187 | 马海燕 | 男   | 2003-05-09 00:00:00 |   1.75 | 东18舍 202
 01878832 | 孙强   | 男   | 2004-02-14 00:00:00 |   1.87 | 东17舍 107
my_db=> SELECT COUNT(*) FROM S500;
 count
-------
  1011
(1 row)
```

```
 PH-74 | 逻辑与批判性思维             |     32 |    2.0 | 王伟
 BA-83 | 分子生物学实验方法与技术     |     32 |    1.0 | 杨旭东
 CS-07 | 面向对象与数据分析 (C#)      |     40 |    2.0 | 齐琪
 CH-09 | 大学化学实验                 |     32 |    1.0 | 范修军
 CH-27 | 有机化学实验-1               |     32 |    1.0 | 朱敏
 CH-68 | 医用近代仪器分析             |     32 |    2.0 | 潘爱钊
 EN-75 | 综合英语-2                   |     48 |    3.0 | 黄平安
 BA-47 | 神经生物学                   |     32 |    2.0 | 陈新林
 MA-71 | 矩阵分析                     |     48 |    3.0 | 徐康丽
 LI-95 | 辞赋与汉唐文化               |     32 |    2.0 | 刘祥
my_db=> SELECT COUNT(*) FROM C500;
 count
-------
   108
(1 row)
```

```
 02971838 | FI-10 |  76.5
 02461027 | PH-15 |  64.0
 02528210 | ST-95 |  62.5
 02469764 | BA-47 |   1.0
 02698537 | PH-64 |  28.5
 02898029 | BI-20 |  85.5
 01978450 | MA-27 |  36.5
 01688622 | PH-17 |  72.0
 02447127 | JZ-75 |  13.5
 01486922 | PH-15 |  51.0
my_db=> SELECT COUNT(*) FROM SC500;
 count
-------
 19826
(1 row)
```

2.在 S500 表中补充数据至约 5000 行，在 C500 表中补充数据至约 1000 行，在 SC500 表中补充数据至约 200000 行。尝试为三、1.中的部分查询（不少于 3 个）编写不同的 SQL 语句实现，分析其运行效率。如果可能，请尝试给出可提高查询效率的改进方法。

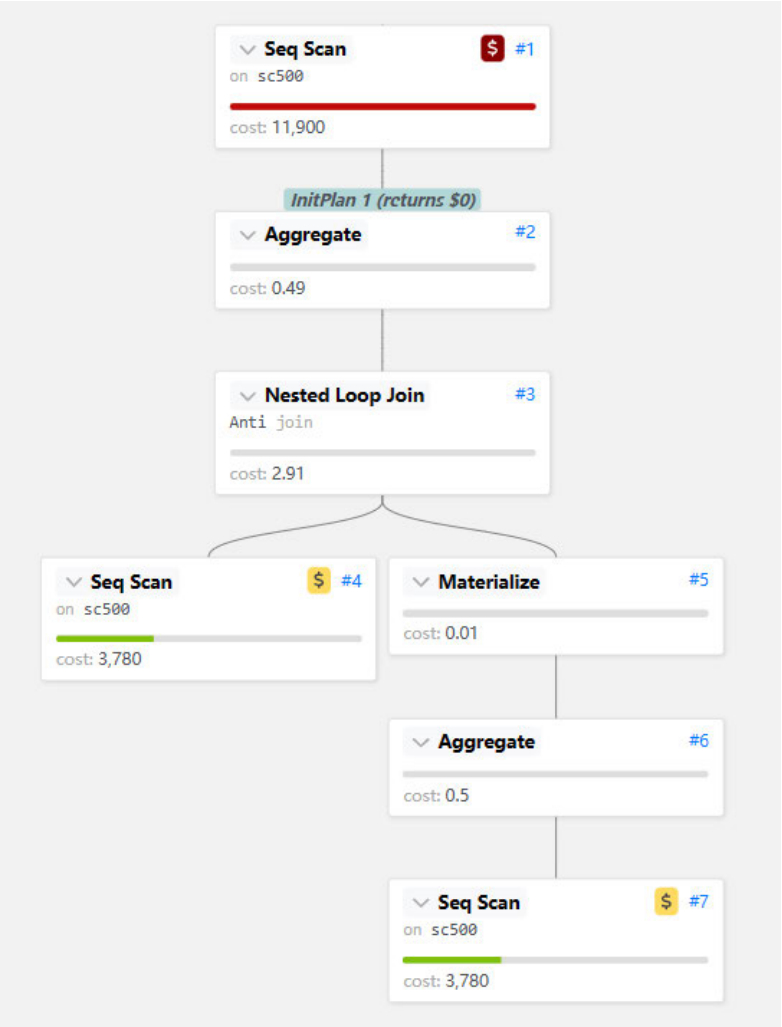
```
my_db=> SELECT COUNT(*) FROM S500;
 count
-------
  5011
(1 row)

my_db=> SELECT COUNT(*) FROM C500;
 count
-------
  1007
(1 row)

my_db=> SELECT COUNT(*) FROM SC500;
 count
--------
 200026
(1 row)
```

①查询选修课程“CS-02”的学生中成绩第二高的学生学号。

写法 1：



分析：写法 1 的查询时间主要耗费在三次顺序扫描上，第一次顺序扫描用于查询课程“CS-02”的最高成绩，第二次顺序扫描用于查询第二高的成绩，最后一次顺序扫描用于查询取得第二高成绩的学生学号。

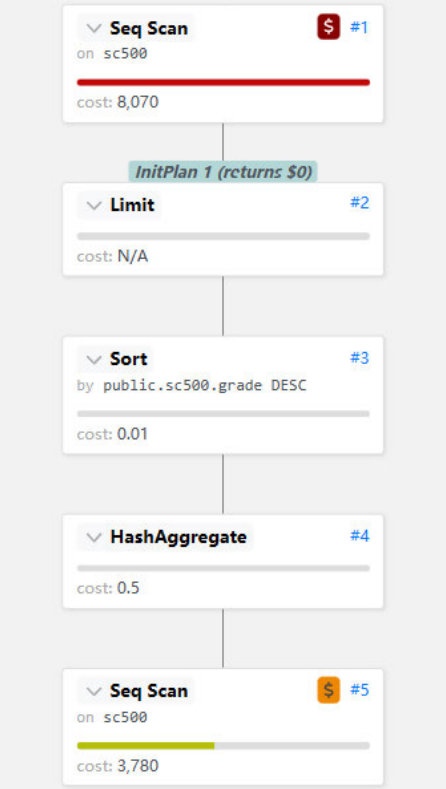
写法 2：

openGauss | my_db | public | 运行 | 停止 | 解释

```
1  SELECT
2      SNO
3  FROM
4      SC500
5  WHERE
6      CNO = 'CS-02'
7      AND GRADE = (
8          SELECT
9              DISTINCT GRADE
10         FROM
11             SC500
12         WHERE
13             CNO = 'CS-02'
14         ORDER BY
15             GRADE DESC
16         LIMIT 1, 1
17     );
```

消息 | 摘要 | 解释 1 | 结果 1

sno
01032010



分析：写法 2 只使用了两次顺序扫描，第一次顺序扫描时将成绩降序排列，且使用 DISTINCT 去除了重复的成绩，再用 LIMIT 取得课程“CS-02”的第二高成绩，最后再用一次顺序扫描查询该成绩对应的学生学号。但是排序可能带来更高的时间复杂度，随着数据量增大，其效率可能不如写法 1。

优化：可以考虑在 SC 表的 CNO 字段上加索引，从而加快课程查找速度。

②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。

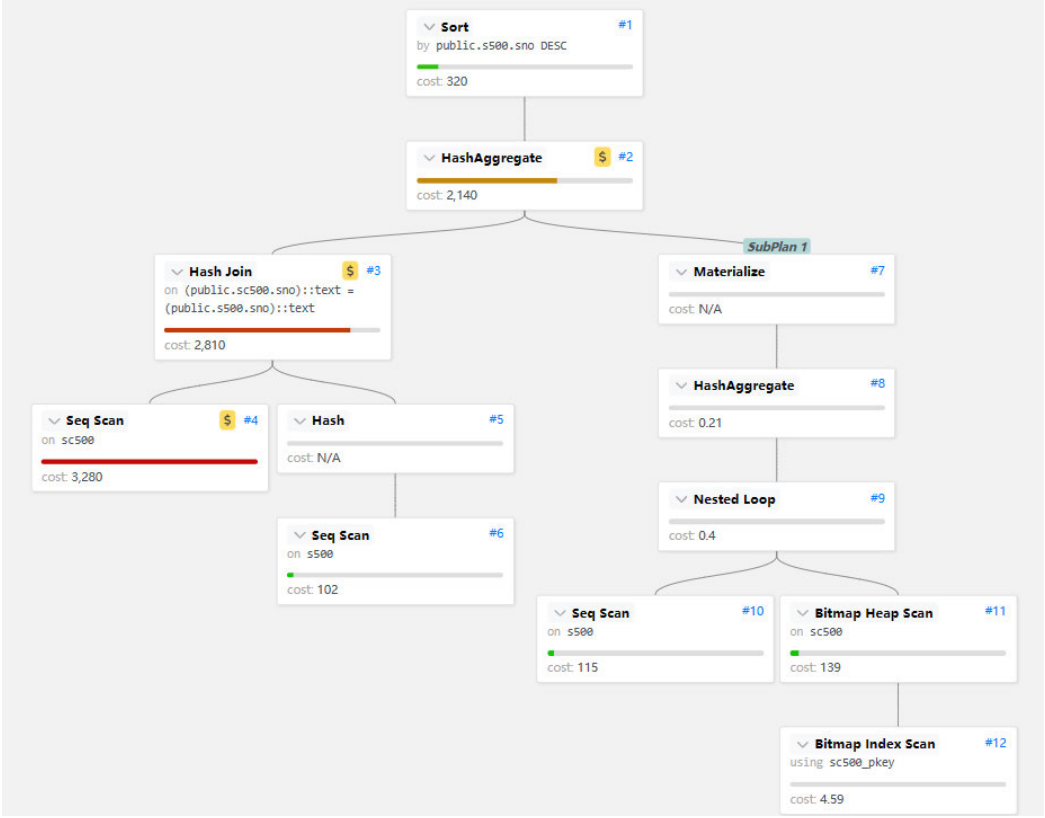
写法 1：

openGauss | my_db | public | 运行 | 停止 | 解释

```
SELECT
    S500.SNO,
    SNAME,
    AVG ( GRADE ) AS AVG_GRADE
FROM
    S500,
    SC500
WHERE
    S500.SNO = SC500.SNO
GROUP BY
    S500.SNO,
    SNAME
HAVING
    AVG ( GRADE ) > ALL (
        SELECT
            AVG ( GRADE )
        FROM
            S500,
            SC500
        WHERE
            S500.SNO = SC500.SNO
            AND SNAME = '王涛'
        GROUP BY
            S500.SNO )
ORDER BY
    S500.SNO DESC;
```

消息 | 摘要 | 解释 1 | 结果 1

sno	sname	avg_grade
03031033	蔡明明	91.0000000000000000
03031011	王倩	88.5000000000000000
01032112	董蔚	88.5000000000000000



分析：写法 1 先在子查询中获得王涛的平均成绩，再进行顺序扫描查询平均成绩超过王涛的同学。

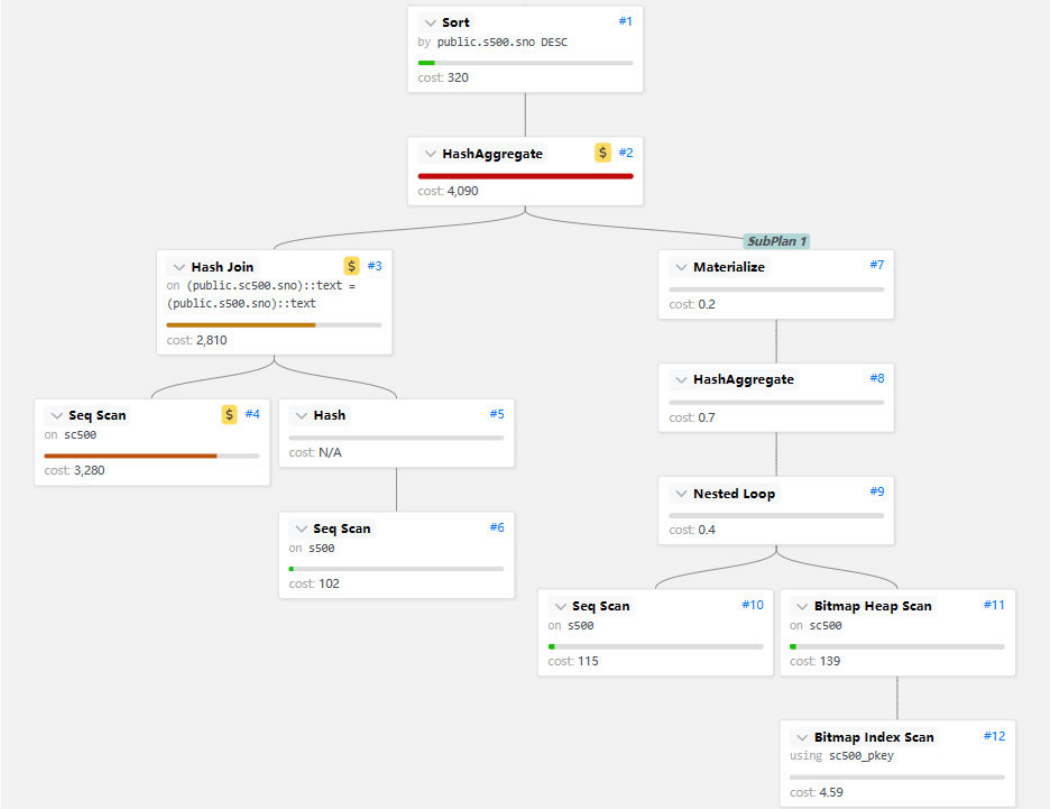
写法 2：

openGauss | my_db | public | 运行 | 停止 | 解释

```
SELECT
    S500.SNO,
    SNAME,
    AVG( GRADE ) AS AVG_GRADE
FROM
    S500 JOIN SC500 ON S500.SNO = SC500.SNO
GROUP BY
    S500.SNO,
    SNAME
HAVING
    AVG( GRADE ) > ALL (
        SELECT
            AVG( GRADE )
        FROM
            SC500
        WHERE
            SNO IN (
                SELECT
                    SNO
                FROM
                    S500
                WHERE
                    SNAME = '王涛' )
        GROUP BY
            SNO )
ORDER BY
    S500.SNO DESC;
```

消息 | 摘要 | 解释 1 | 结果 1

sno	sname	avg_grade
03031033	蔡明明	91.0000000000000000
03031011	王倩	88.5000000000000000
01032112	董蔚	88.5000000000000000



分析：写法 2 与写法 1 的不同在于子查询中，写法 1 先将 S 表与 SC 表连接再进行查询，而写法 2 先从 S 表中获得王涛的学号，再从 SC 表中查询王涛的平均成绩，从而省去了连接表的操作

③查询选修了计算机专业全部课程（课程编号为“CS-××”）的学生姓名及已获得的学分总数。

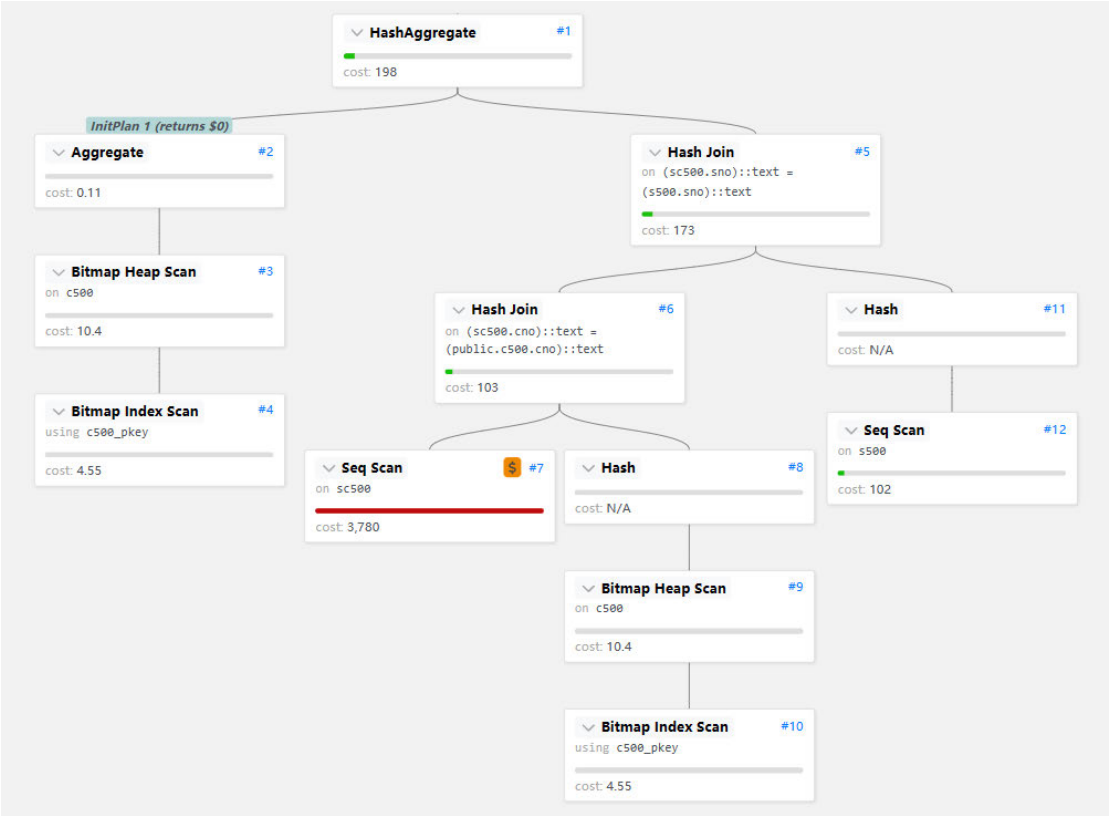
写法 1：

openGauss | my_db | public | 运行 | 停止 | 解释

```
1  SELECT
2      SNAME,
3      SUM ( CASE WHEN COALESCE ( GRADE, 0 ) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END )
4      AS SUM_CREDIT
5  FROM
6      S500,
7      C500,
8      SC500
9  WHERE
10     C500.CNO LIKE 'CS-%'
11     AND C500.CNO = SC500.CNO
12     AND S500.SNO = SC500.SNO
13 GROUP BY
14     S500.SNO,
15     SNAME
16 HAVING
17     COUNT ( * ) = ( SELECT COUNT ( CNO ) FROM C500 WHERE CNO LIKE 'CS-%' );
```

消息 摘要 解释 1 结果 1

sname	sum_credit
(N/A)	(N/A)



分析：写法 1 先将三张表连接，再按照学号分组并查询选修了计算机专业全部课程的学生姓名和获得的学分总数。

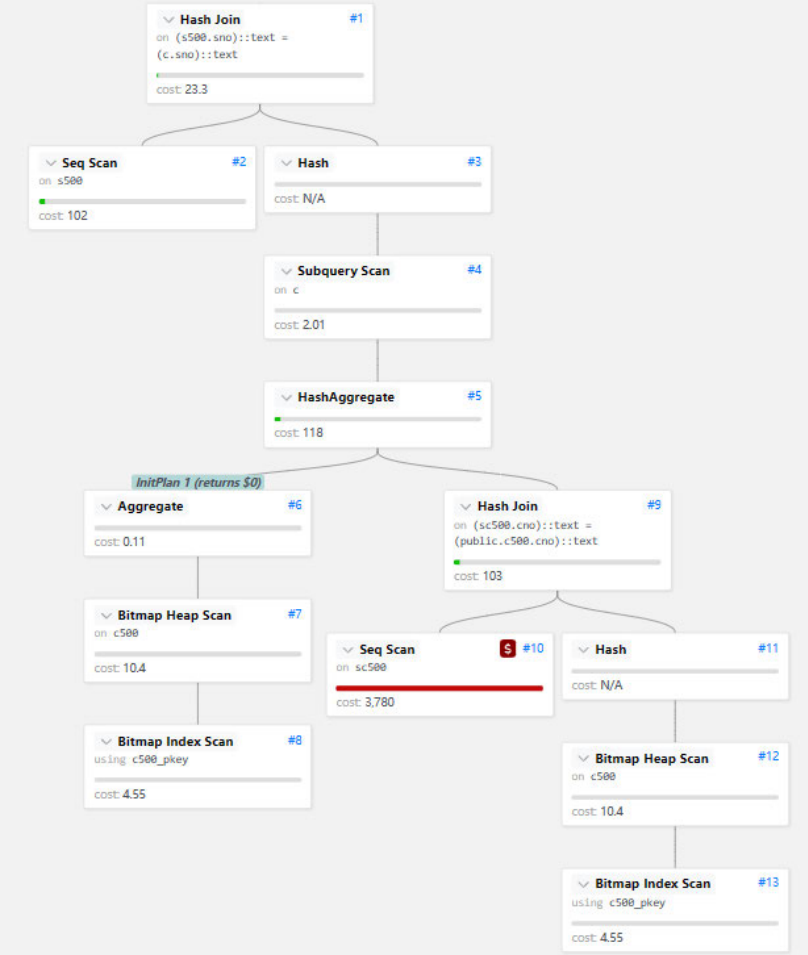
写法 2：

openGauss my_db public 运行 停止 解释

```
1  SELECT
2      SNAME,
3      SUM_CREDIT
4  FROM
5      S500,
6      (SELECT
7          SNO,
8          SUM ( CASE WHEN COALESCE ( GRADE, 0 ) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END )
9          AS SUM_CREDIT
10     FROM
11         SC500,
12         C500
13     WHERE
14         SC500.CNO = C500.CNO
15         AND SC500.CNO LIKE 'CS-%'
16     GROUP BY
17         SNO
18     HAVING
19         COUNT ( * ) = ( SELECT COUNT ( CNO ) FROM C500 WHERE CNO LIKE 'CS-%' )
20     ) AS C
21 WHERE
22     S500.SNO = C.SNO;
```

消息 摘要 解释 1 结果 1

sname	sum_credit
(N/A)	(N/A)



分析：写法 2 先将 SC 表与 C 表连接，再从其中获取选修了计算机专业全部课程的学生学号和获得的学分总数，之后再和 S 表连接以获取学生姓名。与写法 1 相比，写法 2 减少了表连接的数据量。

④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。

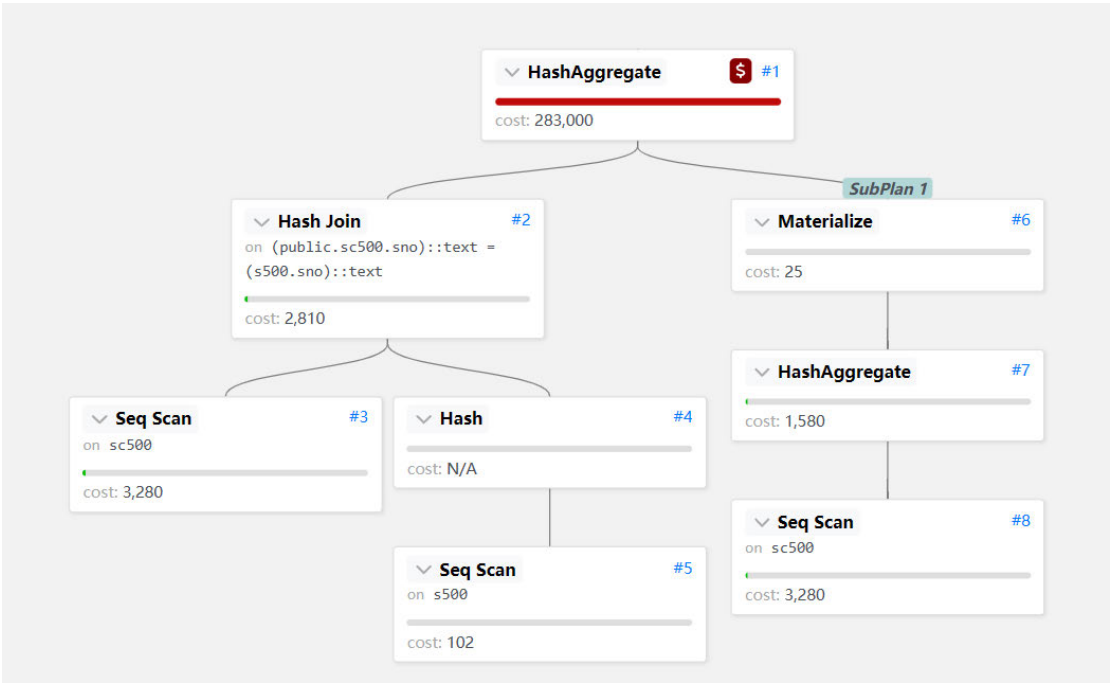
写法 1：

openGauss | my_db | public | 运行 | 停止 | 解释

```
SELECT
    S500.SNO,
    SNAME
FROM
    S500,
    SC500
WHERE
    S500.SNO = SC500.SNO
GROUP BY
    S500.SNO,
    SNAME
HAVING
    COUNT ( * ) >= 3
    AND AVG ( GRADE ) >= ALL (
        SELECT
            AVG ( GRADE )
        FROM
            SC500
        GROUP BY
            SNO
        HAVING
            COUNT ( * ) >= 3 );
```

消息 摘要 解释 1 结果 1

sno	sname
01032112	董蔚



分析：写法 1 先查询选修了 3 门以上课程（包括 3 门）的所有学生的平均成绩，再利用>=ALL 找出取得最高分的学生。

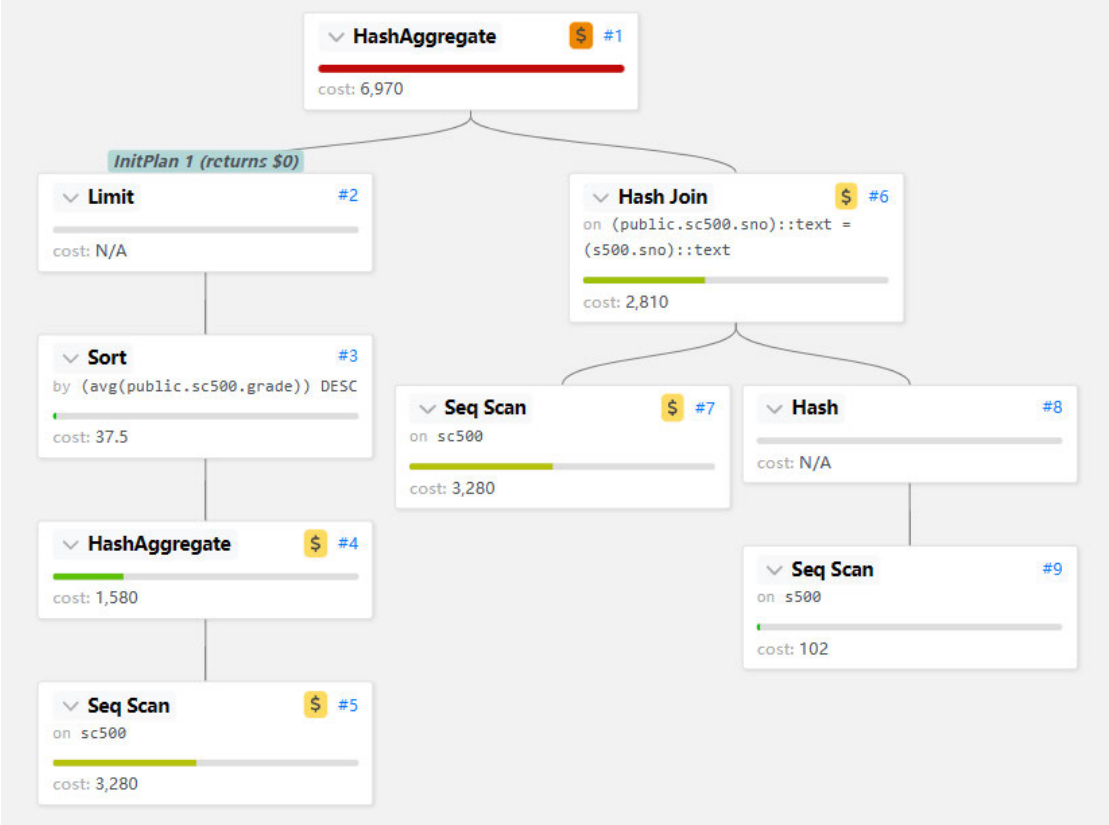
写法 2：

openGauss my_db public 运行 停止 解释

```
SELECT
    S500.SNO,
    SNAME
FROM
    S500,
    SC500
WHERE
    S500.SNO = SC500.SNO
GROUP BY
    S500.SNO,
    SNAME
HAVING
    COUNT ( * ) >= 3
    AND AVG ( GRADE ) = (
        SELECT
            AVG ( GRADE )
        FROM
            SC500
        GROUP BY
            SNO
        HAVING
            COUNT ( * ) >= 3
        ORDER BY
            AVG ( GRADE ) DESC
        LIMIT 0, 1
        );
```

消息 摘要 解释 1 结果 1

sno	sname
01032112	董蔚



分析：写法 2 在查询完选修了 3 门以上课程（包括 3 门）的所有学生的平均成绩后，对其降序排列，从而直接得到了最高分，再利用这个最高分去查询学生信息。与写法 1 对比，写法 2 大大减少了 HashAggregate 的代价。

五、完成上述实验内容后，对数据库进行备份，并交给另一同学进行恢复实验。在成功恢复其他同学交付的数据库备份后，分析其表设计合理性及生成的数据质量，将相应结果截屏保存，并写入实验报告中。

（1）对数据库进行备份，并交给[redacted]同学进行恢复实验

```
[omm@guo ~]$ gs_dump -f guo_db.sql -p 26000 my_db
gs_dump[port='26000'][my_db][2023-06-18 00:10:09]: The total objects number is 408.
gs_dump[port='26000'][my_db][2023-06-18 00:10:09]: [100.00%] 408 objects have been dumped.
gs_dump[port='26000'][my_db][2023-06-18 00:10:10]: dump database my_db successfully
gs_dump[port='26000'][my_db][2023-06-18 00:10:10]: total time: 483  ms
```

（2）恢复[redacted]同学的数据库备份

```
[omm@guo ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:06 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# CREATE DATABASE other_db;
CREATE DATABASE
postgres=# \q
[omm@guo ~]$ gsql -d other_db -p 26000 -f envydb.sql
SET
SET
SET
SET
SET
SET
SET
SET
SET
CREATE TABLE
gsql:envydb.sql:32: ERROR:  role "envy" does not exist
CREATE TABLE
gsql:envydb.sql:49: ERROR:  role "envy" does not exist
```

```
CREATE TABLE
gsql:envydb.sql:64: ERROR:  role "envy" does not exist
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
REVOKE
REVOKE
GRANT
GRANT
total time: 474  ms
```

①S374 表：

```
other_db=# SELECT * FROM S374;
    s#     |  sname   | sex |        bdate        | height |    dorm
-----------+----------+-----+---------------------+--------+-----------
 02710177 | 马彬     | 男  | 2004-03-16 00:00:00 |   1.74 | 东17舍314
 02506014 | 李宇     | 男  | 2004-02-15 00:00:00 |   1.85 | 东17舍119
 01192096 | 蒋桂芳   | 男  | 2003-10-13 00:00:00 |   1.67 | 东17舍312
 02850791 | 何玉珍   | 男  | 2003-07-31 00:00:00 |   1.72 | 东17舍217
 02929957 | 扈林     | 男  | 2003-12-19 00:00:00 |   1.69 | 东18舍230
 02964420 | 李平     | 男  | 2003-07-28 00:00:00 |   1.67 | 东15舍616
 01545685 | 华超     | 女  | 2002-04-26 00:00:00 |   1.46 | 东4舍809
 02358027 | 黄淑华   | 女  | 2002-02-03 00:00:00 |   1.48 | 东1舍130
 02926380 | 白勇     | 女  | 2004-05-03 00:00:00 |   1.41 | 东3舍511
 02797835 | 梁淑珍   | 女  | 2002-02-09 00:00:00 |   1.74 | 东4舍118
 01735190 | 周海燕   | 男  | 2003-06-13 00:00:00 |   1.63 | 东14舍219
 01919938 | 曹玉英   | 男  | 2003-07-09 00:00:00 |   1.86 | 东15舍121
 02596795 | 李桂珍   | 男  | 2002-07-29 00:00:00 |   1.60 | 东17舍425
 01741479 | 陈琴     | 男  | 2002-02-12 00:00:00 |   1.64 | 东16舍124
 02797630 | 杜建华   | 男  | 2002-01-05 00:00:00 |   1.97 | 东16舍825
 01381902 | 徐秀华   | 男  | 2004-01-12 00:00:00 |   1.82 | 东17舍312
 01552433 | 刘丽华   | 女  | 2003-12-27 00:00:00 |   1.68 | 东4舍512
```

```
other_db=# \d+ S374
                                Table "public.s374"
 Column |            Type             |  Modifiers  | Storage  | Stats target | Description
--------+-----------------------------+-------------+----------+--------------+-------------
 s#     | character(8)                | not null    | extended |              |
 sname  | character varying(30)       | not null    | extended |              |
 sex    | character(3)                | not null    | extended |              |
 bdate  | timestamp(0) without time zone | not null    | plain    |              |
 height | numeric(3,2)                | default 0.0 | main     |              |
 dorm   | character varying(20)       |             | extended |              |
Indexes:
    "s374_pkey" PRIMARY KEY, btree ("s#") TABLESPACE pg_default
Referenced by:
    TABLE "sc374" CONSTRAINT "sc374_s#_fkey" FOREIGN KEY ("s#") REFERENCES s374("s#") ON DELETE CASCADE
Has OIDs: no
Options: orientation=row, compression=no
```

S374 表将 S#作为主键，各个字段属性正确，生成的学生信息数据质量高。

②C374 表：

```
other_db=# SELECT * FROM C374;
  c#   |                          cname                          | period | credit |   teacher
-------+---------------------------------------------------------+--------+--------+-------------
 LA-01 | 犯罪学                                                  | 32     |    2.0 | 安婧
 MA-01 | 概率统计与随机过程                                      | 64     |    4.0 | 王宁
 MA-02 | 概率统计与随机过程                                      | 64     |    4.0 | 王宁
 EP-01 | 电工实习 I                                              | 32     |    1.0 | 张春梅
 BI-01 | 医用光学仪器                                            | 32     |    2.0 | 王斯佳
 BI-02 | 生物组学与精准医学                                      | 32     |    2.0 | 王昌河
 CS-06 | 程序设计基础                                            | 56     |    3.0 | 夏秦
 CH-01 | 物质结构与性质-有机篇                                   | 88     |    4.0 | 王婉秦
 CH-02 | 物质结构与性质-有机篇                                   | 88     |    4.0 | 王婉秦
 BI-03 | 生物制药技术                                            | 32     |    2.0 | 孔令洪
 BI-04 | 生物组学与精准医学                                      | 32     |    2.0 | 王昌河
 BI-05 | 医用光学仪器                                            | 32     |    2.0 | 王斯佳
 PH-01 | 体育 2 （民族预科）                                     | 32     |     .5 | 李宝成
 MA-03 | 概率论与数理统计                                        | 48     |    3.0 | 康艳梅
 BE-01 | 建筑设备自动控制                                        | 32     |    2.0 | 王军
 MA-04 | 概率论与数理统计                                        | 48     |    3.0 | 康艳梅
 PH-02 | 体育 2 （民族预科）                                     | 32     |     .5 | 贺智裕
 ML-01 | 中国近现代史纲要                                        | 32     |    2.0 | 宋希斌
```

```
other_db=# \d+ C374
                                Table "public.c374"
 Column  |          Type          | Modifiers | Storage  | Stats target | Description
---------+------------------------+-----------+----------+--------------+-------------
 c#      | character(5)           | not null  | extended |              |
 cname   | character varying(100) | not null  | extended |              |
 period  | tinyint                | not null  | plain    |              |
 credit  | numeric(3,1)           |           | main     |              |
 teacher | character varying(30)  |           | extended |              |
Indexes:
    "c374_pkey" PRIMARY KEY, btree ("c#") TABLESPACE pg_default
Referenced by:
    TABLE "sc374" CONSTRAINT "sc374_c#_fkey" FOREIGN KEY ("c#") REFERENCES c374("c#") ON DELETE CASCADE
Has OIDs: no
Options: orientation=row, compression=no
```

C374 表将 C#作为主键，各个字段属性正确，课程信息来自于学校课表，数据质量高。

③SC374 表：

```
other_db=# SELECT * FROM SC374;
    s#     |  c#   | grade
-----------+-------+-------
 01046883 | MA-06 |  73.7
 01046883 | PH-31 |  98.8
 01046883 | PH-10 |  67.2
 01046883 | CH-03 |  59.9
 01046883 | CH-04 |  97.9
 01046883 | PH-28 |  74.3
 01046883 | PH-40 |  50.0
 01046883 | CH-01 |  98.0
 01046883 | MA-07 |  74.7
 01046883 | ST-03 |  53.4
```

```
other_db=# \d+ SC374
                                  Table "public.sc374"
 Column |     Type      |       Modifiers       | Storage  | Stats target | Description
--------+---------------+-----------------------+----------+--------------+-------------
 s#     | character(8)  | not null              | extended |              |
 c#     | character(5)  | not null              | extended |              |
 grade  | numeric(4,1)  | default NULL::numeric | main     |              |
Indexes:
    "sc374_pkey" PRIMARY KEY, btree ("s#", "c#") TABLESPACE pg_default
Check constraints:
    "sc374_grade_check" CHECK (grade IS NULL OR grade >= 0::numeric AND grade <= 100::numeric)
Foreign-key constraints:
    "sc374_c#_fkey" FOREIGN KEY ("c#") REFERENCES c374("c#") ON DELETE CASCADE
    "sc374_s#_fkey" FOREIGN KEY ("s#") REFERENCES s374("s#") ON DELETE CASCADE
Has OIDs: no
Options: orientation=row, compression=no
```

SC374 表将 S#与 C#作为主键且正确设置了外键，各个字段属性正确，使用 CHECK 对 Grade 进行检查，实现了正确的完整性约束。

六、附录

（1）数据生成

①Python 爬虫获取全校课程信息：

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

# 学号与密码
NetID = None
password = None

# 创建 WebDriver 对象
driver = webdriver.Edge()
```

```
# 打开本科教务平台
url = 'http://ehall.xjtu.edu.cn/new/index.html?browser=no'
driver.get(url)

# 点击登录
login_button = driver.find_element(By.ID, 'ampHasNoLogin')
login_button.click()

# 输入学号和密码
user_box = driver.find_element(By.NAME, 'username')
password_box = driver.find_element(By.NAME, 'pwd')
user_box.send_keys(NetID)
password_box.send_keys(password)

# 统一身份认证网关
login_button = driver.find_element(By.CSS_SELECTOR, '.loginState > #account_login')
driver.execute_script('arguments[0].click();', login_button)

# 全校课表查询
time.sleep(3)
app_list = driver.find_element(By.CSS_SELECTOR, '.amp-aside-box-mini-item > .icon-liebiao1')
driver.execute_script('arguments[0].click();', app_list)
course_table = driver.find_element(By.CSS_SELECTOR, '.appFlag:nth-child(10) > .amp-str-cut')
driver.execute_script('arguments[0].click();', course_table)

# 切换页面
new_window = driver.window_handles[-1]
driver.switch_to.window(new_window)

# 获取课程信息
with open('course_crawler.txt', mode='w') as f:
    for i in range(400):
        time.sleep(3)
        for j in range(10):
            cno1 = f'id("row{j}qxkcb-index-table")/TD[3]/SPAN[1]'
            cno2 = f'id("row{j}qxkcb-index-table")/TD[5]/SPAN[1]'
            cname = f'id("row{j}qxkcb-index-table")/TD[4]/SPAN[1]'
            college = f'id("row{j}qxkcb-index-table")/TD[6]/SPAN[1]'
            period = f'id("row{j}qxkcb-index-table")/TD[8]/SPAN[1]'
            credit = f'id("row{j}qxkcb-index-table")/TD[9]/SPAN[1]'
```

```
            teacher = f'id("row{j}qxkcb-index-table")/TD[10]/SPAN[1]'
            f.write(driver.find_element(By.XPATH,
cno1).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
cno2).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
cname).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
college).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
period).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
credit).get_attribute('title') + '; ')
            f.write(driver.find_element(By.XPATH,
teacher).get_attribute('title') + '\n')
        next_page = driver.find_element(By.CSS_SELECTOR,
'#pagerqxkcb-index-table .icon-keyboardarrowright')
        driver.execute_script('arguments[0].click();', next_page)

# 关闭浏览器
driver.quit()
```

②生成 S500 表：

```
from faker import Faker

data_size = 5000

sno_list = []
name_list = []
sex_list = []
birth_list = []
height_list = []
dorm_list = []

sno_exist = ['01032010', '01032023', '01032001', '01032005',
'01032112',
             '03031011', '03031014', '03031051', '03031009',
'03031033',
             '03031056']
dorm_exist = ['东 14 舍 221', '东 14 舍 221', '东 1 舍 312', '东 1 舍 312',
'东 14 舍 221',
              '东 2 舍 104', '东 18 舍 421', '东 18 舍 422', '东 2 舍 104', '东
18 舍 423',
              '东 2 舍 305']
```

```
# 女生住东 1-东 5 男生住东 14-18
# 假设每栋楼 8 层 每层 25 间宿舍 每间宿舍 4 人
dorm_male = {}
building_male = ['东 14 舍', '东 15 舍', '东 16 舍', '东 17 舍', '东 18 舍']
for building in building_male:
    for i in range(1, 9):
        for j in range(1, 26):
            dorm_male[f'{building}{i*100+j}'] = 4

dorm_female = {}
building_female = ['东 1 舍', '东 2 舍', '东 3 舍', '东 4 舍', '东 5 舍']
for building in building_female:
    for i in range(1, 9):
        for j in range(1, 26):
            dorm_female[f'{building}{i*100+j}'] = 4

for dorm in dorm_exist:
    if dorm in dorm_male:
        dorm_male[dorm] -= 1
    else:
        dorm_female[dorm] -= 1

# 随机生成学生信息
fake = Faker('zh_CN')
for i in range(data_size):
    # 生成学号
    sno = f'0{fake.random.randrange(1032000, 3032000, 1)}'
    while sno in sno_exist:
        sno = f'0{fake.random.randrange(1032000, 3032000, 1)}'
    sno_exist.append(sno)
    # 按照男女比例 7：3 生成数据
    if i % 10 <= 6:
        name = fake.name_male()
        sex = '男'
        height = fake.random.randrange(160, 190) / 100
        # 分配宿舍
        dorm = fake.random.sample(dorm_male.keys(), 1)[0]
        while dorm_male[dorm] == 0:
            dorm = fake.random.sample(dorm_male.keys(), 1)[0]
        dorm_male[dorm] -= 1
    else:
        name = fake.name_female()
        sex = '女'
        height = fake.random.randrange(150, 180) / 100
```

```
        # 分配宿舍
        dorm = fake.random.sample(dorm_female.keys(), 1)[0]
        while dorm_female[dorm] == 0:
            dorm = fake.random.sample(dorm_female.keys(), 1)[0]
        dorm_female[dorm] -= 1
    # 随机生日
    birth = fake.date_of_birth(minimum_age=19,
maximum_age=23).strftime('%Y-%m-%d')

    sno_list.append(sno)
    name_list.append(name)
    sex_list.append(sex)
    birth_list.append(birth)
    height_list.append(height)
    dorm_list.append(dorm)

# 保存数据
with open('student.txt', mode='w', encoding='utf-8') as f:
    for i in range(data_size):
        f.write(sno_list[i] + ' ')
        f.write(name_list[i] + ' ')
        f.write(sex_list[i] + ' ')
        f.write(birth_list[i] + ' ')
        f.write(str(height_list[i]) + ' ')
        f.write(dorm_list[i] + '\n')
```

③生成 C500 表：

```
import random

data_size = 1000
cno_exist = ['CS-01', 'CS-02', 'CS-04', 'CS-05', 'EE-01', 'EE-02',
'EE-03']
cno_list = []
name_list = []
period_list = []
credit_list = []
teacher_list = []

with open('course_crawler.txt', mode='r') as f:
    lines = f.readlines()

total_size = len(lines)
i = 0
while i < data_size:
    line = lines[random.randrange(total_size)][:-1]
```

```
    record = line.split('; ')

    name = record[2]
    teacher = record[6].split(',')[0]
    if len(name) > 16 or len(teacher) > 3:
        continue

    cno = record[0]
    if cno[:4] == 'COMP':
        cno = 'CS'
    else:
        cno = cno[:2]
    num = abs(hash(record[0][4:10])) % 100
    cno += f'-{num}' if num > 9 else f'-0{num}'
    if cno in cno_exist or cno in cno_list:
        continue

    period = record[4]
    credit = record[5]

    cno_list.append(cno)
    name_list.append(name)
    period_list.append(period)
    credit_list.append(credit)
    teacher_list.append(teacher)
    i += 1

with open('course.txt', mode='w', encoding='utf-8') as f:
    for i in range(data_size):
        f.write(cno_list[i] + '; ')
        f.write(name_list[i] + '; ')
        f.write(period_list[i] + '; ')
        f.write(credit_list[i] + '; ')
        f.write(teacher_list[i] + '\n')
```

④生成 SC500 表：

```
import random

data_size = 200000
limit = 100     # 每名学生选课上限 100 门
student_limit = {}
primary_key = {}
sno_list = []
cno_list = []
grade_list = []
```

```
with open('student.txt', mode='r', encoding='utf-8') as f:
    student = f.readlines()

with open('course.txt', mode='r', encoding='utf-8') as f:
    course = f.readlines()

student_num = len(student)
course_num = len(course)
i = 0
while i < data_size:
    sno = student[random.randrange(student_num)].split(' ')[0]
    if sno not in student_limit:
        student_limit[sno] = 0
    else:
        if student_limit[sno] < limit:
            student_limit[sno] += 1
        else:
            continue

    cno = course[random.randrange(course_num)].split('; ')[0]

    if (sno, cno) in primary_key:
        continue

    primary_key[(sno, cno)] = True
    sno_list.append(sno)
    cno_list.append(cno)

    grade = round(random.uniform(0, 100) / 0.5) * 0.5
    grade_list.append(grade)

    i += 1

with open('sc.txt', mode='w', encoding='utf-8') as f:
    for i in range(data_size):
        f.write(sno_list[i] + ' ')
        f.write(cno_list[i] + ' ')
        f.write(str(grade_list[i]) + '\n')
```

（2）数据插入

①插入 S500 表：

```
import java.sql.*;
import java.io.*;
```

```
public class UpdateS500 {
    public static void main(String[] args) {
        // 数据库用户与密码
        String username = "guo";
        String password = "Bigdata@123";
        // 保存待插入数据的文件名
        String filename = "./data/student.txt";
        // 待插入的数据量
        int data_num = 5000;
        // 字段分隔符
        String delimiter = " ";
        // 驱动程序
        String driver = "org.postgresql.Driver";
        // 数据库 URL
        String url = "jdbc:postgresql://124.70.51.155:26000/my_db";

        try {
            // 加载驱动程序
            Class.forName(driver);
            // 连接数据库
            Connection conn = DriverManager.getConnection(url,
username, password);
            // 创建预编译 SQL 语句
            String sql = "INSERT INTO S500 (SNO, SNAME, SEX, BDATE,
HEIGHT, DORM) VALUES (?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
            // 读取数据文件
            BufferedReader reader = new BufferedReader(new
FileReader(filename));
            String line = reader.readLine();
            for (int i = 0; i < data_num && line != null; i++) {
                // 获取每条记录中各字段值
                String[] fields = line.split(delimiter);
                // 向 SQL 语句中填入各字段值
                ps.setString(1, fields[0]);
                ps.setString(2, fields[1]);
                ps.setString(3, fields[2]);
                ps.setDate(4, Date.valueOf(fields[3]));
                ps.setFloat(5, Float.parseFloat(fields[4]));
                ps.setString(6, fields[5]);
                // 执行插入语句
                ps.executeUpdate();
                // 读取下一行记录
                line = reader.readLine();
```

```
            }
            // 关闭数据文件
            reader.close();
            // 关闭预编译 SQL 语句
            ps.close();
            // 关闭数据库连接
            conn.close();
            System.out.println("Update successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

②插入 C500 表：

```
import java.sql.*;
import java.io.*;

public class UpdateC500 {
    public static void main(String[] args) {
        // 数据库用户与密码
        String username = "guo";
        String password = "Bigdata@123";
        // 保存待插入数据的文件名
        String filename = "./data/course.txt";
        // 待插入的数据量
        int data_num = 1000;
        // 字段分隔符
        String delimiter = "; ";
        // 驱动程序
        String driver = "org.postgresql.Driver";
        // 数据库 URL
        String url = "jdbc:postgresql://124.70.51.155:26000/my_db";

        try {
            // 加载驱动程序
            Class.forName(driver);
            // 连接数据库
            Connection conn = DriverManager.getConnection(url,
username, password);
            // 创建预编译 SQL 语句
            String sql = "INSERT INTO C500 (CNO, CNAME, PERIOD,
CREDIT, TEACHER) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
            // 读取数据文件
```

```
            BufferedReader reader = new BufferedReader(new
FileReader(filename));
            String line = reader.readLine();
            for (int i = 0; i < data_num && line != null; i++) {
                // 获取每条记录中各字段值
                String[] fields = line.split(delimiter);
                // 向 SQL 语句中填入各字段值
                ps.setString(1, fields[0]);
                ps.setString(2, fields[1]);
                ps.setInt(3, Integer.parseInt(fields[2]));
                ps.setFloat(4, Float.parseFloat(fields[3]));
                ps.setString(5, fields[4]);
                // 执行插入语句
                ps.executeUpdate();
                // 读取下一行记录
                line = reader.readLine();
            }
            // 关闭数据文件
            reader.close();
            // 关闭预编译 SQL 语句
            ps.close();
            // 关闭数据库连接
            conn.close();
            System.out.println("Update successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

③插入 SC500 表：

```
import java.sql.*;
import pack.InsertThread;
import pack.DeleteThread;

public class UpdateSC500 {
    public static void main(String[] args) {
        // 数据库用户与密码
        String username = "guo";
        String password = "Bigdata@123";
        // 保存待插入数据的文件名
        String filename = "./data/sc.txt";
        // 待插入的数据量
        int insert_data_num = 200000;
        // 待删除的数据量
```

```
        int delete_data_num = 200;
        // 字段分隔符
        String delimiter = " ";
        // 是否删除数据
        boolean flag = false;
        // 驱动程序
        String driver = "org.postgresql.Driver";
        // 数据库 URL
        String url = "jdbc:postgresql://124.70.51.155:26000/my_db";

        try {
            // 加载驱动程序
            Class.forName(driver);
            // 连接数据库
            Connection conn = DriverManager.getConnection(url,
username, password);
            // 创建两个线程
            InsertThread thread1 = new InsertThread(conn, filename,
delimiter, insert_data_num);
            DeleteThread thread2 = new DeleteThread(conn, flag,
delete_data_num);
            thread1.start();
            thread2.start();
            thread1.join();
            thread2.join();
            // 关闭数据库连接
            conn.close();
            System.out.println("Update successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
package pack;

import java.sql.*;
import java.io.*;

public class InsertThread extends Thread {
    private final Connection conn;
    private final String filename;
    private final String delimiter;
    private final int data_num;
```

```
    private final Object lock = new Object();

    public InsertThread(Connection conn, String filename, String
delimiter, int data_num) {
        this.conn = conn;
        this.filename = filename;
        this.delimiter = delimiter;
        this.data_num = data_num;
    }

    @Override
    public void run() {
        try {
            // 创建预编译 SQL 语句
            String sql = "INSERT INTO SC500 (SNO, CNO, GRADE) VALUES
(?, ?, ?)";
            PreparedStatement ps = this.conn.prepareStatement(sql);
            // 读取数据文件
            BufferedReader reader = new BufferedReader(new
FileReader(this.filename));
            String line = reader.readLine();
            for (int i = 0; i < this.data_num && line != null; i++) {
                synchronized (lock) {
                    // 获取每条记录中各字段值
                    String[] fields = line.split(this.delimiter);
                    // 向 SQL 语句中填入各字段值
                    ps.setString(1, fields[0]);
                    ps.setString(2, fields[1]);
                    ps.setFloat(3, Float.parseFloat(fields[2]));
                    // 执行插入语句
                    ps.executeUpdate();
                    // 读取下一行记录
                    line = reader.readLine();
                }
            }
            // 关闭数据文件
            reader.close();
            // 关闭预编译 SQL 语句
            ps.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
package pack;

import java.sql.*;

public class DeleteThread extends Thread {
    private final Connection conn;
    private final boolean flag;
    private final int data_num;
    private final Object lock = new Object();

    public DeleteThread(Connection conn, boolean flag, int
data_num) {
        this.conn = conn;
        this.flag = flag;
        this.data_num = data_num;
    }

    @Override
    public void run() {
        try {
            if (this.flag) {
                Statement stmt = conn.createStatement();
                String sql = "DELETE FROM SC500 WHERE (SNO, CNO) IN "
+
                        "(SELECT SNO, CNO FROM SC500 WHERE GRADE < 60
ORDER BY RANDOM() LIMIT 1)";
                for (int i = 0; i < this.data_num; i++) {
                    synchronized (lock) {
                        if (stmt.executeUpdate(sql) == 0){
                            i--;
                        }
                    }
                }
                stmt.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```