**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：**

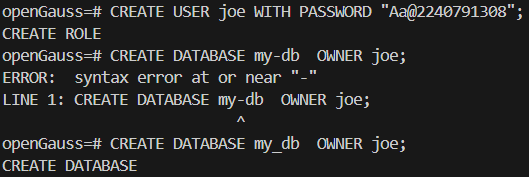
**S549（S#，SNAME，SEX，BDATE，HEIGHT，DORM）**

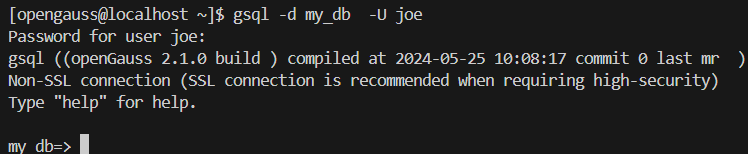
**C549（C#，CNAME，PERIOD，CREDIT，TEACHER）**

**SC549（S#，C#，GRADE）其中 S#、C#均为外键**

本次实验在本地主机上安装openEuler和opengauss，通过vscode实现ssh远程连接

设置密码





CREATE TABLE IF NOT EXISTS S549

(Sno Integer PRIMARY KEY, Sname VARCHAR(32), Sex Char(4), BDATE Date, Height

Number, Dorm VARCHAR(32));

CREATE TABLE IF NOT EXISTS C549

(Cno VARChar(16) PRIMARY KEY, Cname VARCHAR(32), Period Integer, Credit

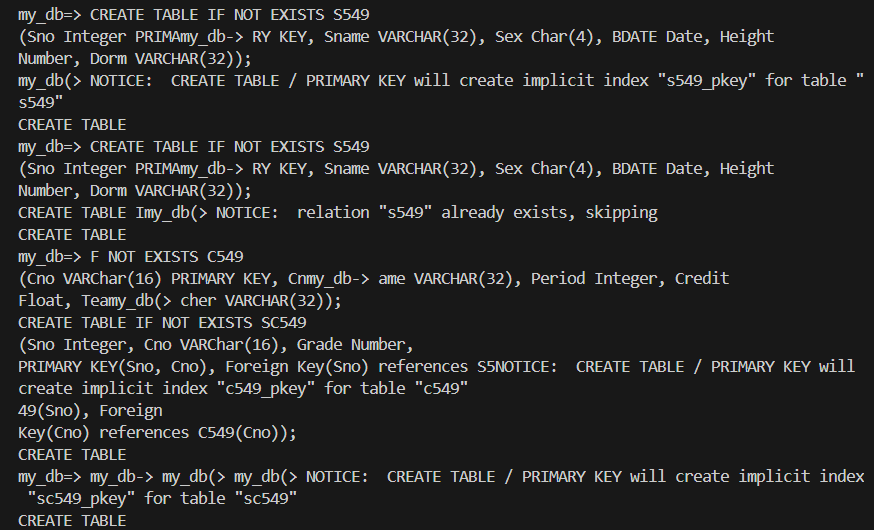
Float, Teacher VARCHAR(32));

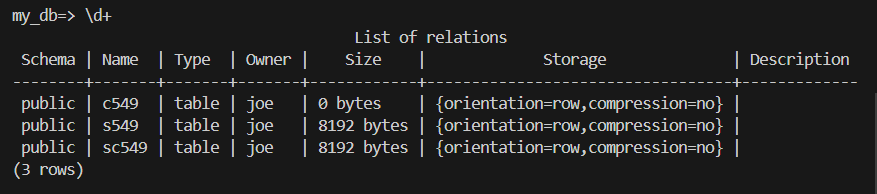
CREATE TABLE IF NOT EXISTS SC549

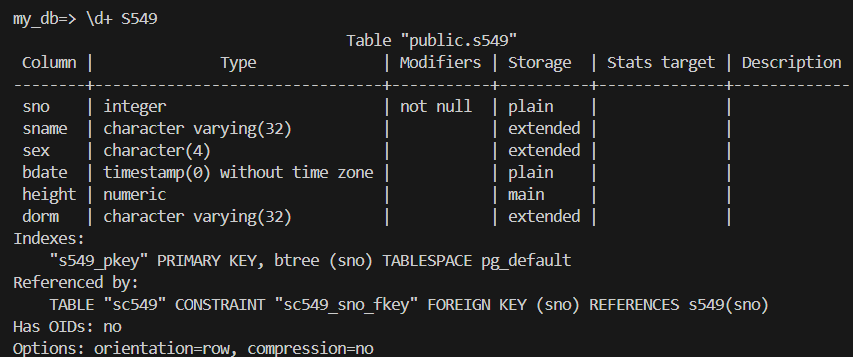
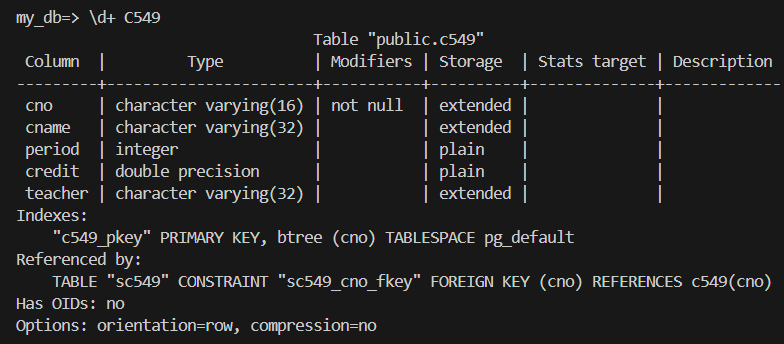
(Sno Integer, Cno VARChar(16), Grade Number,

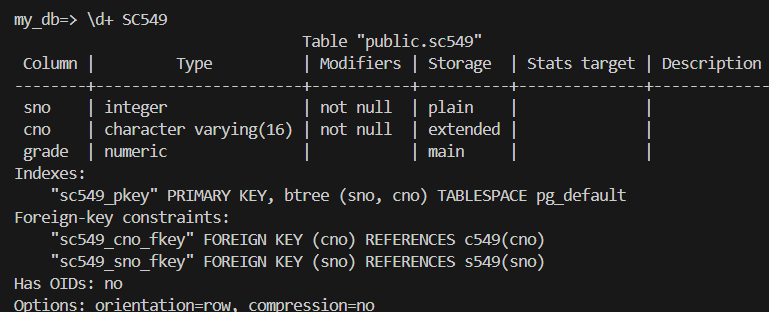
PRIMARY KEY(Sno, Cno), Foreign Key(Sno) references S549(Sno), Foreign

Key(Cno) references C549(Cno))







**二、将数据加入相应的表中。**

INSERT INTO S549 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'),

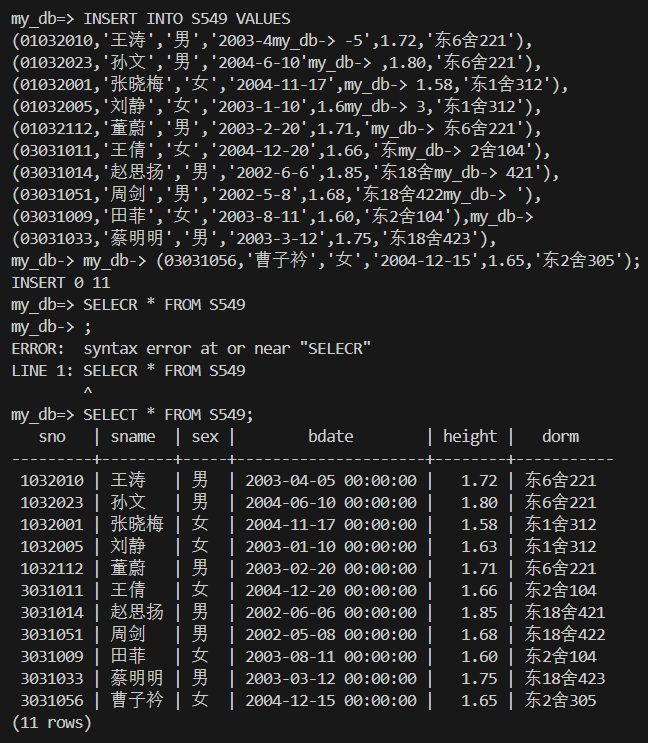
(03031014,'赵思扬','男','2002-6-6',1.85,'东18舍421'),

(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 INTO C549 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'),

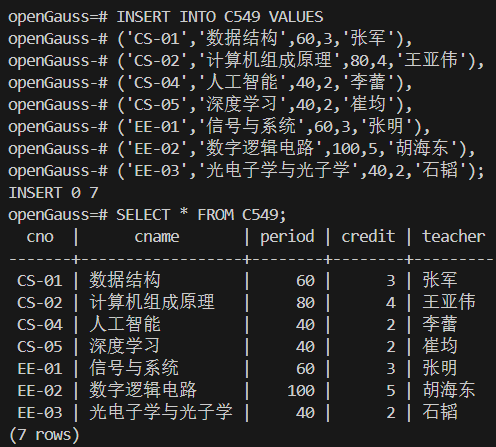
('CS-04','人工智能',40,2,'李蕾'),

('CS-05','深度学习',40,2,'崔均'),

('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 INTO SC549 VALUES

(01032010,'CS-01',82),

(01032010,'CS-02',91),

(01032010,'CS-04',83.5),

(01032001,'CS-01',77.5),

(01032001,'CS-02',85),

(01032001,'CS-04',83),

(01032005,'CS-01',62),

(01032005,'CS-02',77),

(01032005,'CS-04',82),

(01032023,'CS-01',55),

(01032023,'CS-02',81),

(01032023,'CS-04',76),

(01032112,'CS-01',88),

(01032112,'CS-02',91.5),

(01032112,'CS-04',86),

(01032112,'CS-05',NULL),

(03031033,'EE-01',93),

(03031033,'EE-02',89),

(03031009,'EE-01',88),

(03031009,'EE-02',78.5),

(03031011,'EE-01',91),

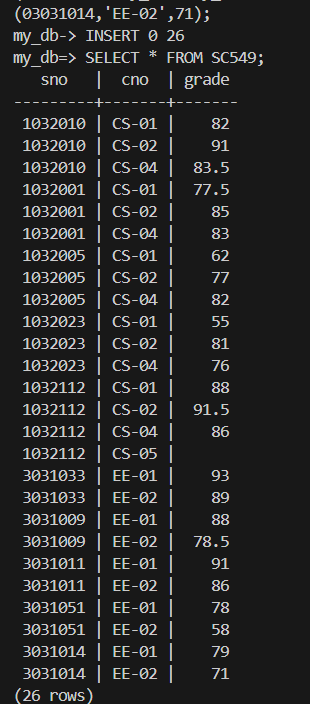
(03031011,'EE-02',86),

(03031051,'EE-01',78),

(03031051,'EE-02',58),

(03031014,'EE-01',79),

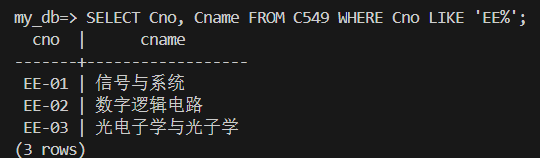
(03031014,'EE-02',71);



**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。

SELECT Cno, Cname FROM C549 WHERE Cno LIKE 'EE%';

(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。

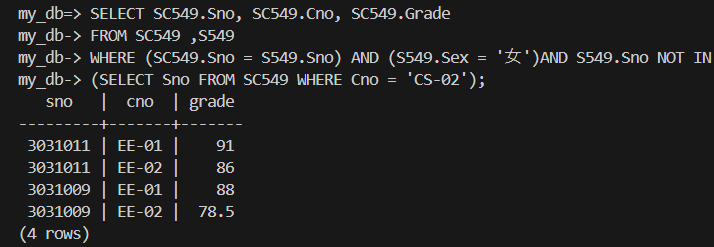
SQL 语句解释：先通过 SELECT 子查询获得选修课程“CS-02”的学生学号，

再使用 NOT IN 语句得到最终结果。

SELECT SC549.Sno, SC549.Cno, SC549.Grade

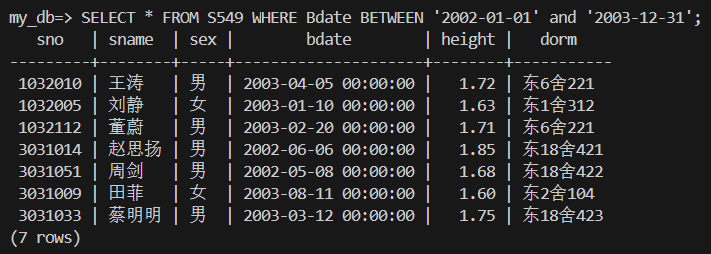
FROM SC549 ,S549

WHERE (SC549.Sno = S549.Sno) AND (S549.Sex = '⼥')AND S549.Sno NOT IN (SELECT Sno FROM SC549 WHERE Cno = 'CS-02');

 (3) 查询 2002 年～2003 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。

SELECT \* FROM S549 WHERE Bdate BETWEEN '2002-01-01' and '2003-12-31';

 (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，S549 表中的部分学生未选修任何课程，在 SC549 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。

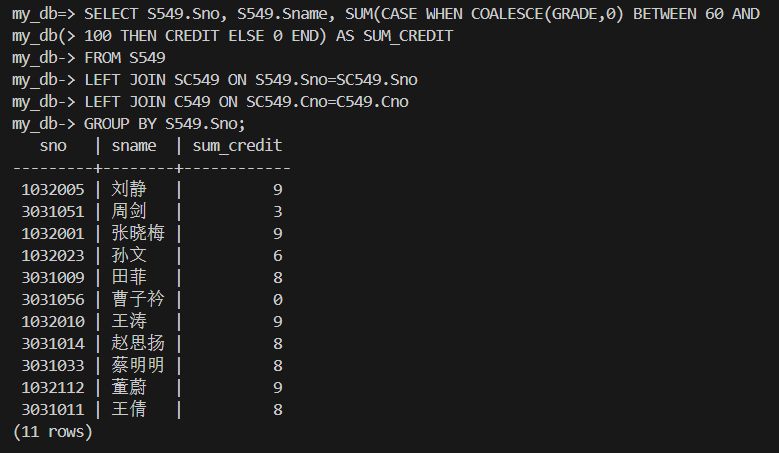
SELECT S549.Sno, S549.Sname, SUM(CASE WHEN COALESCE(GRADE,0) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END) AS SUM\_CREDIT

FROM S549

LEFT JOIN SC549 ON S549.Sno=SC549.Sno

LEFT JOIN C549 ON SC549.Cno=C549.Cno

GROUP BY S549.Sno;

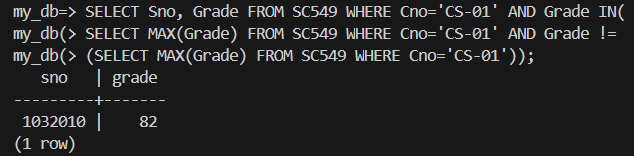


(5) 查询选修课程“CS-01”的学生中成绩第二高的学生学号。

SELECT Sno, Grade FROM SC549 WHERE Cno='CS-01' AND Grade IN(

SELECT MAX(Grade) FROM SC549 WHERE Cno='CS-01' AND Grade !=

(SELECT MAX(Grade) FROM SC549 WHERE Cno='CS-01'));



(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 Sno, Sname , AvgGrade FROM

(SELECT SC.Sno Sno, S549.Sname Sname, AVG(SC.Grade) AvgGrade

FROM S549 JOIN (SELECT \* FROM SC549 WHERE Grade IS NOT NULL) AS SC

ON S549.Sno=SC.Sno GROUP BY SC.Sno, S549.Sname) AS Tmp

WHERE AvgGrade > (SELECT AvgGrade FROM

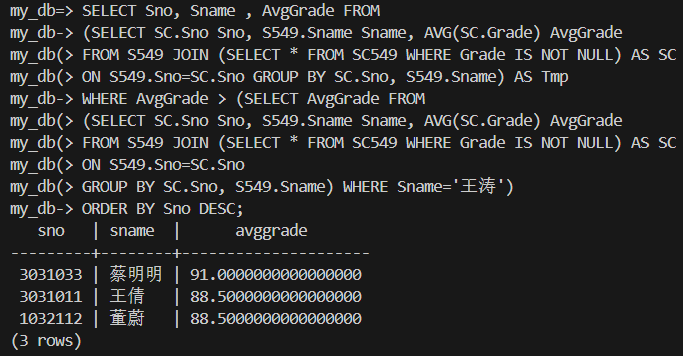
(SELECT SC.Sno Sno, S549.Sname Sname, AVG(SC.Grade) AvgGrade

FROM S549 JOIN (SELECT \* FROM SC549 WHERE Grade IS NOT NULL) AS SC

ON S549.Sno=SC.Sno

GROUP BY SC.Sno, S549.Sname) WHERE Sname='王涛')

ORDER BY Sno DESC;



(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。

SELECT Sname,SUM(CASE WHEN COALESCE(GRADE,0) BETWEEN 60 AND 100 THEN CREDIT ELSE 0 END) AS SUM\_CREDIT

FROM S549,SC549,C549

WHERE S549.Sno=SC549.Sno AND SC549.Cno=C549.Cno AND NOT EXISTS

(SELECT \*

FROM (SELECT Cno FROM C549 WHERE Cno LIKE CONCAT('CS', '%')) AS Cor

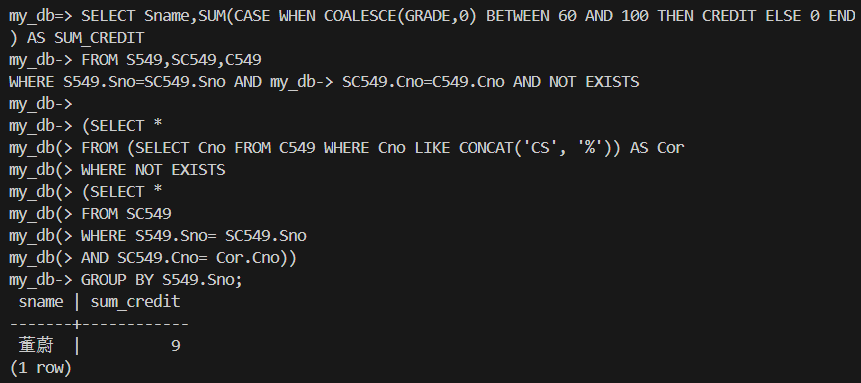
WHERE NOT EXISTS

(SELECT \*

FROM SC549

WHERE S549.Sno= SC549.Sno AND SC549.Cno= Cor.Cno))

GROUP BY S549.Sno;



(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姓名。

SELECT ST.Sno, Sname

FROM

( (SELECT S549.Sno Sno

FROM S549 JOIN SC549 SC

ON S549.Sno=SC.Sno

GROUP BY S549.Sno HAVING COUNT(SC.Cno)>=3) AS ST

JOIN

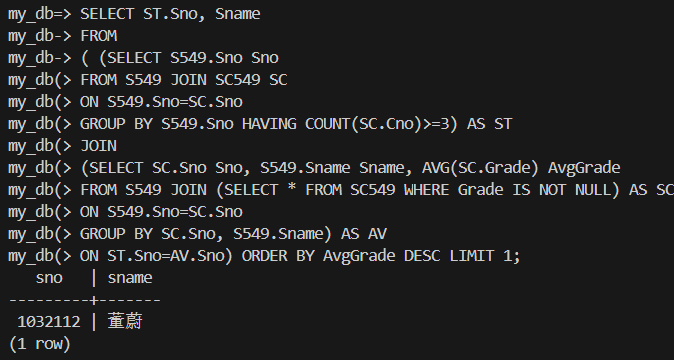
(SELECT SC.Sno Sno, S549.Sname Sname, AVG(SC.Grade) AvgGrade

FROM S549 JOIN (SELECT \* FROM SC549 WHERE Grade IS NOT NULL) AS SC

ON S549.Sno=SC.Sno

GROUP BY SC.Sno, S549.Sname) AS AV

ON ST.Sno=AV.Sno) ORDER BY AvgGrade DESC LIMIT 1;



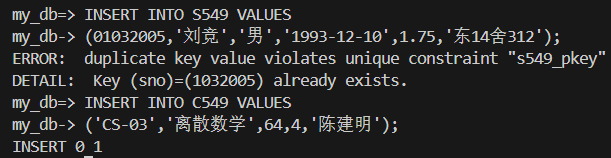
2．分别在 S549 和 C549 表中加入记录(‘01032005’，‘刘竞’，‘男’，

‘2003-12-10’，1.75，‘东14舍 312’)及(‘CS-03’，“离散数学”，64，4， ‘陈建明’)。

INSERT INTO S549 VALUES

(01032005,'刘竞','男','1993-12-10',1.75,'东14舍312');

INSERT INTO C549 VALUES

('CS-03','离散数学',64,4,'陈建明');

由于主键已经存在，故插入失败。成功在 C549 表中插入一条“CS-03”的记录。

3．将 S549 表中已修学分数大于 60 的学生记录删除。

DELETE FROM SC549

WHERE Sno IN

(SELECT S549.Sno Sno

FROM S549 JOIN

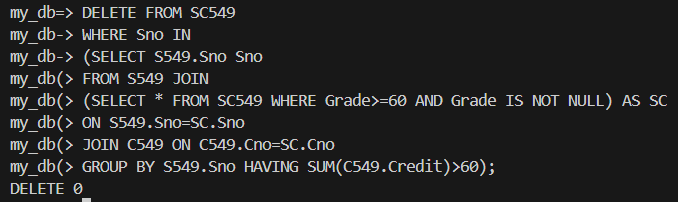
(SELECT \* FROM SC549 WHERE Grade>=60 AND Grade IS NOT NULL) AS SC

ON S549.Sno=SC.Sno

JOIN C549 ON C549.Cno=SC.Cno

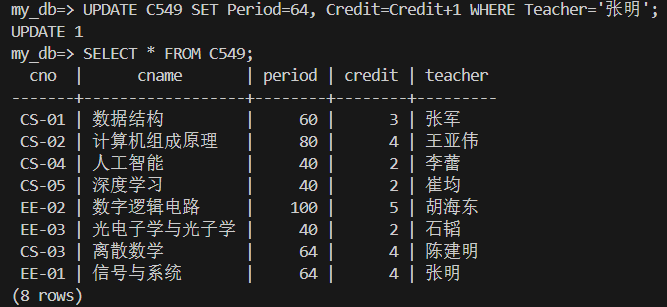
GROUP BY S549.Sno HAVING SUM(C549.Credit)>60

) ;



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。

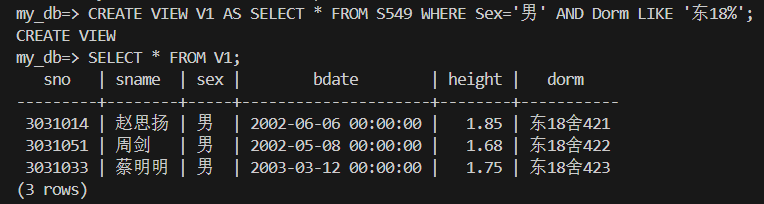
UPDATE C549 SET Period=64, Credit=Credit+1 WHERE Teacher='张明';



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

CREATE VIEW V1 AS SELECT \* FROM S549 WHERE Sex='男' AND Dorm LIKE '东18%';



(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

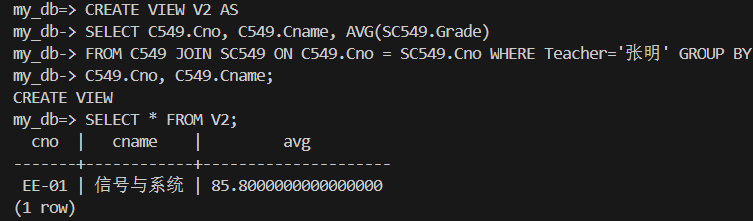
成绩等属性。

CREATE VIEW V2 AS

SELECT C549.Cno, C549.Cname, AVG(SC549.Grade)

FROM C549 JOIN SC549 ON C549.Cno = SC549.Cno WHERE Teacher='张明' GROUP BY

C549.Cno, C549.Cname;



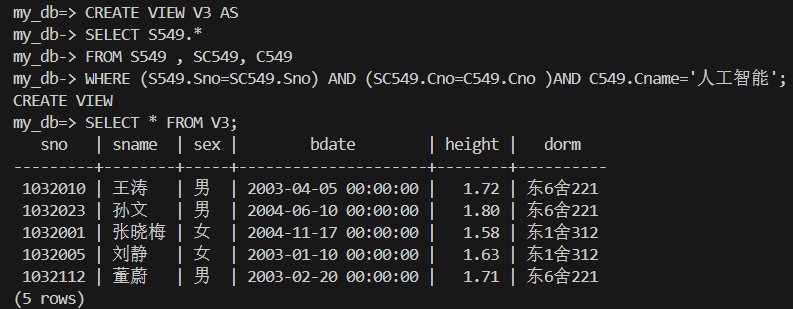
(3)所有选修了“人工智能”课程的学生视图，包括学号、姓名、成绩等属性

CREATE VIEW V3 AS

SELECT S549.\*

FROM S549 , SC549, C549

WHERE (S549.Sno=SC549.Sno) AND (SC549.Cno=C549.Cno )AND C549.Cname='人工智能';



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 S549 表中补充数据至约 1000 行，在 C549 表中补充数据至约 100 行，在 SC549 表中补充数据至约 20000 行。在向 SC549 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。

**Python 随机⽣成数据**

在 script/ ⽬录下创建 expand ⽂件，在其中写⼊随机⽣成的命令。

为保证 SC 表中外键依赖，将⽣成的 sno 和 cno 储存，在⽣成 SC 表随机数据时将其随机组合作为主键。

为保证 SC 表中主键唯⼀，考虑到 python dict 底层为 HASH ，使⽤ dict 数据结构储存主键

若 dict.get(主键) == True ，说明该主键已⽣成过，则重新随机⽣成。

使⽤迭代器，优化代码结构。

import random

import time

import os

S\_LEN = 1000

C\_LEN = 549

SC\_LEN = 2000

SNO\_START = int(1033e3)

# yyyy, mm, dd, h, m ,s

date1 = (2002, 1, 1, 0, 0, 0, -1, -1, -1)

time1 = time.mktime(date1)

date2 = (2005, 1, 1, 0, 0, 0, -1, -1, -1)

time2 = time.mktime(date2)

first\_name = ["赵", "钱", "孙", "李", "刘","周", "吴", "郑", "王", "冯", "陈", "褚", "卫", "蒋", "沈", "韩", "杨", "朱", "秦", "尤", "许", "何",

              "吕", "施", "张", "孔", "曹", "严", "华","石", "金", "魏", "陶", "姜", "戚", "谢", "邹", "喻", "柏", "水", "窦", "章", "云", "苏",

              "潘", "葛", "奚", "范", "彭", "郎", "鲁", "韦", "昌", "马", "苗", "凤", "花", "方", "俞", "任", "袁", "柳", "酆", "鲍", "史",

              "唐", "费", "廉", "岑", "薛", "雷", "贺", "倪", "汤", "滕", "殷", "罗", "毕", "郝", "邬", "安", "常", "乐", "于", "时", "傅",

              "皮", "卞","徐", "齐", "康", "伍", "余", "元", "卜", "顾", "孟", "平", "黄", "和", "穆", "萧", "尹", "姚", "邵", "堪", "汪"]

last\_name = ['玉', '明', '龙', '芳', '军', '玲', '', '立', '玲', '', '国', "地", "为", "子", "中", "", "", "", "国", "年", "着", "就",

             "那", "和", "要", "刚","她", "出", "也", "", "", "", "自", "以", "会", "家", "可", "下", "事", "把", "还", "用", "第", "样", "道",

             "想", "作", "种", "开", "美", "总", "从", "无", "情", "己", "面", "最", "女", "但", "现", "前", "些", "所", "同", "日", "手",

             "又", "行", "丽","意", "动", "方", "期", "它", "头", "经", "长", "儿", "回", "位", "分", "爱", "老", "因", "很", "给", "名", "法",

             "间", "斯", "知","雪", "世", "什", "两", "次", "使", "身", "者", "被", "高", "已", "亲", "其", "进", "此", "话", "常", "与", "活",

             "正", "感", "见", "明","建", "问", "力", "理", "尔", "点", "文", "几", "定", "本", "公", "特", "做", "外", "孩", "相", "西", "果",

             "走", "将", "月", "十", "实", "向", "声", "车", "全", "信", "重", "三", "机", "工", "物", "气", "每", "并", "别", "真", "打",

             "太", "新", "比", "才", "便", "夫", "再", "书", "部", "水", "像", "眼", "等", "体", "却", "加", "电", "主", "界", "门", "利",

             "海", "受", "听", "表", "德", "少", "克", "代", "员", "许", "稜", "先", "口", "由", "死", "安", "写", "性", "马", "光", "白",

             "或", "住", "难", "望", "教", "命", "花", "结", "乐", "色", "更", "拉", "东", "神", "记", "处", "让", "母", "父", "应", "直",

             "字", "场", "平", "报", "友", "关", "放", "至", "张", "认", "接", "告", "入", "笑", "内", "英", "军", "候", "民", "岁", "往",

             "何", "度", "山"]

genders = ['女', '男']

dorms = ['东', '西']

sno = SNO\_START

first\_class = ['深度', '爱情', '经济', '电机', '电路', '数据结构','物理','数学分析','医学','睡眠', '操作', '数据库', '网络', '计算机组成']

last\_class = ['学习', '理论','课程','导论', '教学', '实践','项目', '基础']

deps = ['CS', 'EE','HT','MI','ML', 'SC', 'FI','PH','ST','HH', 'LLM','CV','BA','AI', 'HW']

snos = []

cnos = []

log = open(os.path.join('.', 'expand'), 'w')

def gen\_name():

    while True:

        full\_name = random.choice(first\_name) + random.choice(last\_name) + random.choice(last\_name)

        if len(full\_name) > 1:

            return full\_name

def record(msg):

    print(msg, end='')

    log.write('%s' % msg)

    log.flush()

record('INSERT INTO S549 VALUES \n')

for i in range(S\_LEN):

    count = random.randint(1, 3)

    sno = sno + count

    full\_name = gen\_name()

    random\_time = random.uniform(time1, time2)  # uniform返回随机实数 time1 <= time < time2

    birthday = time.strftime("%Y-%m-%d", (time.localtime(random\_time)))

    gender = random.choice(genders)

    height = random.uniform(1.4, 2.0)

    height = round(height, 2)

    dorm = '%s%d舍%d%d' % (random.choice(dorms), random.randint(1, 20),

                          random.randint(1, 20), random.randint(1, 22),)

    snos.append(sno)

    if i != S\_LEN - 1:

        record("({}, '{}', '{}', '{}', {}, '{}'),\n".format(sno, full\_name, gender, birthday, height, dorm))

    else:

        record("({}, '{}', '{}', '{}', {}, '{}');\n".format(sno, full\_name, gender, birthday, height, dorm))

record('\n\n')

def cache(func):

    ca = {}

    while True:

        args = func()

        if not ca.get(args):

            ca[args] = True

            yield args

record('INSERT INTO C549 VALUES \n')

cno\_gen = cache(lambda: ('%s-%d' % (random.choice(deps), random.randint(1, 100))))

for i in range(C\_LEN):

    cno = next(cno\_gen)

    class\_name = random.choice(first\_class)  + random.choice(last\_class)

    full\_name = gen\_name()

    ctime = random.randrange(20, 60, 4)

    gender = random.choice(genders)

    credit = random.randrange(1, 13) / 2

    credit = round(credit, 1)

    cnos.append(cno)

    if i != C\_LEN - 1:

        record("('{}', '{}', {}, {}, '{}'),\n".format(cno, class\_name, ctime, credit, full\_name))

    else:

        record("('{}', '{}', {}, {}, '{}');\n".format(cno, class\_name, ctime, credit, full\_name))

record('\n\n')

record('INSERT INTO SC549 VALUES \n')

key\_gen = cache(lambda: (random.choice(snos), random.choice(cnos)))

for i in range(SC\_LEN):

    key = next(key\_gen)

    grade = random.randrange(80, 200) / 2

    grade = round(grade, 1)

    if i != SC\_LEN - 1:

        record("({}, '{}', {}),\n".format(key[0], key[1], grade))

    else:

        record("({}, '{}', {});\n".format(key[0], key[1], grade))

record('\n\n')

record('-- Finish')

log.flush()

log.close()

**JDBC**

编写 ExecCommand 函数，可以执⾏普通操作，如有错则则会提⽰

public static void ExecCommand(Connection conn, String command) throws InterruptedException {

        Statement stmt = null;

        try {

            stmt = conn.createStatement();

            stmt.execute(command);

            stmt.close();

            TimeUnit.MICROSECONDS.sleep(1000);

        } catch (SQLException e) {

            System.out.println("Error occurs when executing " + command);

            if (stmt != null) {

                try {

                    stmt.close();

                } catch (SQLException e1) {

                    e1.printStackTrace();

                }

            }

            e.printStackTrace();

        }

    }

编写 ExecSelect 函数，可以执⾏ SELECT 操作并输出查询结果

public static void ExecSelect(Connection conn, String sql) {

        Statement stmt =null;

        try {

            stmt = conn.createStatement();

            System.out.println("=======================================");

            System.out.printf("Executing %s:%n",sql);

            ResultSet rs = stmt.executeQuery(sql);

            String str=null;

            while(rs.next()){

                str = "";

                for(int i=1;i<=rs.getMetaData().getColumnCount();i++){

                    str += rs.getString(i)+",";

                }

                System.out.println(str);

            }

            if (str == null){

                System.out.println("Found empty!");

            }

            System.out.println("=======================================");

            rs.close();

            stmt.close();

        } catch (SQLException e) {

            if (stmt != null) {

                try {

                    stmt.close();

                }

                catch (SQLException e1) {

                    e1.printStackTrace();

                }

            }

            System.out.println("Error!");

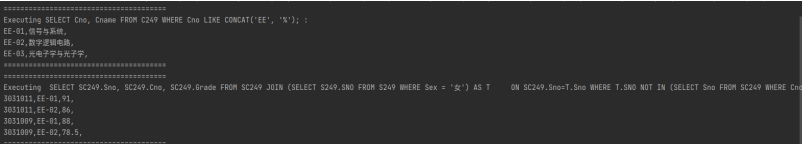
            e.printStackTrace();

            System.out.println("=======================================");

        }

    }

输出示例：



编写 ExecFile 函数，可以将⽂件内的⾮注释⾏读⼊并执⾏sql语句

public static void ExecFile(Connection conn, String filename) throws IOException, InterruptedException {

        FileReader fr=new FileReader(filename);

        BufferedReader br=new BufferedReader(fr);

        String line;

        String buf="";

        while ((line=br.readLine())!=null) {

            if (!line.contains("--")){

                buf = buf + line + " ";

            }

            if (line.contains(";")){

                if (buf.toLowerCase().contains("select") & !buf.toLowerCase().contains("create")){

                    ExecSelect(conn, buf);

                }

                else {

                    System.out.println(buf);

                    ExecCommand(conn, buf);

                }

                buf = "";

            }

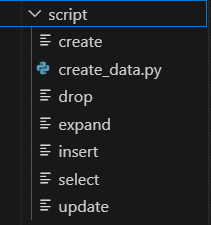
        }

        br.close();

        fr.close();

    }

可以在 script/ ⽬录下创建若⼲sql命令⽂件，通过 ExecFile 函数读⼊并执⾏



 public static void main(String[] args){

        //创建数据库连接。

        String USERNAME = "joe";

        String PASSWORD = "ba@2265932745";

        String DB = "my\_db";

        Integer PORT = 5432;

        try {

            Connection conn = GetConnection(USERNAME, PASSWORD, DB, PORT);

            assert conn != null;

            ExecFile(conn, "./script/drop");

            ExecFile(conn, "./script/create");

            ExecFile(conn, "./script/insert");

            ExecFile(conn, "./script/expand");

            ExecFile(conn, "./script/select");

//

            ExecFile(conn, "./script/update");

            ExecFile(conn, "./script/drop");

            conn.close();

        } catch (SQLException e) {

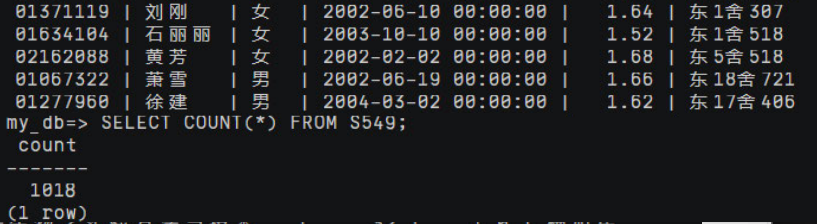
            e.printStackTrace();

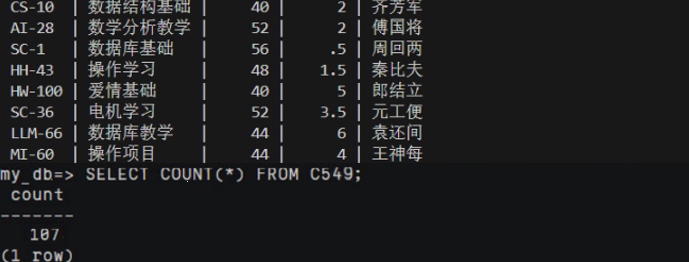
        } catch (IOException | InterruptedException e) {

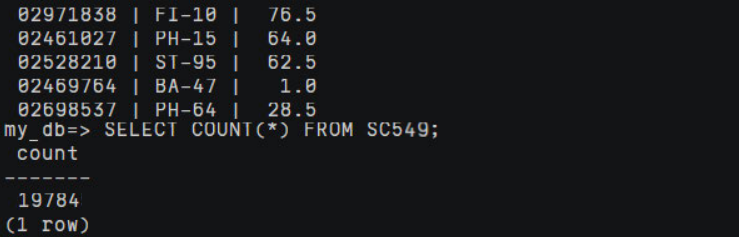
            throw new RuntimeException(e);

        }

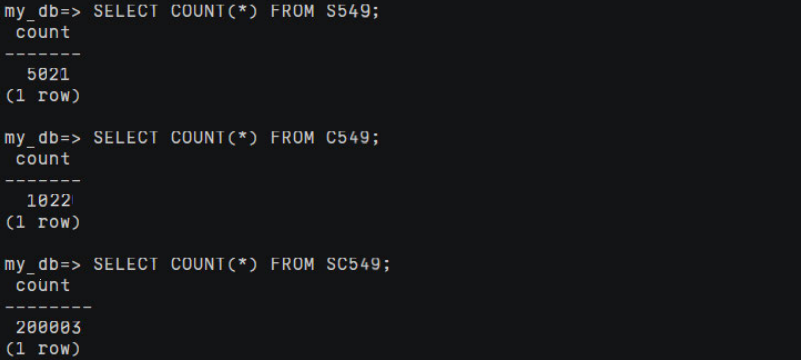
    }







2．在 S549 表中补充数据至约 5490 行，在 C549 表中补充数据至约 1000 行，在 SC549 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。



1. 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。

优化前：

SELECT SC549.Sno, SC549.Cno, SC549.Grade

FROM SC549 JOIN (SELECT S549.SNO FROM S549 WHERE Sex = '⼥') AS T

ON SC549.Sno=T.Sno

WHERE T.SNO NOT IN (SELECT Sno FROM SC549 WHERE Cno = 'CS-02');

优化后：

SELECT SC549.Sno, SC549.Cno, SC549.Grade

FROM SC549 JOIN (SELECT S549.SNO FROM S549 WHERE Sex = '⼥') AS T

ON SC549.Sno=T.Sno

WHERE NOT EXISTS (SELECT Sno FROM SC549 WHERE Cno = 'CS-02' AND T.Sno = sc549.Sno

分析：

在本例上性能基本相同，但最好使⽤NOT EXISTS⽽不是NOT IN, 原因是如果查询语句使⽤了not in，那么对内外表都进⾏全表扫描，没有⽤到索引；⽽not exists的⼦查询依然能⽤到表上的索引。所以⽆论哪个表⼤，⽤not exists都⽐not in 要快。

➁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 S549.Sno, S549.Sname, SUM(C549.Credit)

FROM S549, C549, (SELECT \* FROM SC549 WHERE Grade>=60 AND Grade IS NOT NULL) AS SC

WHERE S549.Sno=SC.Sno AND C549.Cno=SC.Cno

GROUP BY S549.Sno;

考虑将SC549删除掉成绩不合格或者没有成绩的记录之后再与S549和C549进行连接，然后按照学号进行分组输出结果

➂ 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 Sno, Sname , AvgGrade

FROM

(SELECT SC.Sno Sno, S549.Sname Sname, AVG(SC.Grade) AvgGrade

FROM S549 JOIN (SELECT \* FROM SC549 WHERE Grade IS NOT NULL) AS SC ON

S549.Sno=SC.Sno

GROUP BY SC.Sno, S549.Sname

HAVING Avggrade >any(

SELECT AVG(SC.Grade) AvgGrade FROM (SELECT \* FROM SC549 WHERE

Grade IS NOT NULL) AS SC WHERE Sno=(SELECT Sno FROM S549 WHERE Sname='王涛')

GROUP BY SC.Sno

)

)

ORDER BY Sno DESC;

这条SQL语句首先通过连接和聚合计算得到每个学生的平均成绩，然后筛选出平均成绩超过“王涛”同学的学生，最后按学号降序排列结果。通过一个内层子查询 (SELECT \* FROM SC549 WHERE Grade IS NOT NULL) AS SC，获取所有成绩不为空的记录，并将其命名为 SC。

接着，通过 S549 JOIN SC ON S549.Sno = SC.Sno，将学生表 S549 与刚才获取的成绩表 SC 按学号 Sno 进行连接，获取每个学生的详细信息和他们的成绩。

然后，使用 GROUP BY SC.Sno, S549.Sname 对这些连接结果按学号和姓名进行分组，计算每个学生的平均成绩 AVG(SC.Grade)，并命名为 AvgGrade。

最后，使用 HAVING AvgGrade > ANY (...) 进行过滤，筛选出平均成绩超过“王涛”同学的学生。这里的 ANY 子查询：首先获取所有成绩不为空的记录，接着根据 Sno = (SELECT Sno FROM S549 WHERE Sname = '王涛') 筛选出“王涛”的成绩记录，并计算“王涛”的平均成绩。因此，HAVING AvgGrade > ANY (...) 的作用是只选择那些平均成绩大于“王涛”同学的学生记录。外部查询从子查询的结果中选择学生的学号 Sno、姓名 Sname 和平均成绩 AvgGrade，并使用 ORDER BY Sno DESC 按学号降序排列结果。

➃ 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。

SELECT S549.Sname

FROM S549

WHERE NOT EXISTS

(SELECT \*

FROM C549 COR

WHERE NOT EXISTS

(SELECT \*

FROM SC549

WHERE S549.Sno= SC549.Sno

外层查询选择所有满足条件的学生姓名 S549.Sname。条件是 NOT EXISTS 后面的子查询返回结果为假。NOT EXISTS 用于检查子查询是否返回任何行，如果没有返回行则为真。

第一个子查询从课程表 C549 中选择所有课程（使用别名 COR）。条件是对于这些课程，存在一个嵌套的子查询。

嵌套子查询从选课表 SC549 中选择记录，条件是：学生编号 S549.Sno 与 SC549.Sno 匹配。课程编号 SC549.Cno 与 COR.Cno 匹配。课程编号 Cno 以 "CS" 开头。

➄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。

SELECT Sno, Sname FROM S549 WHERE Sno = (

SELECT Sno FROM

(SELECT Sno, Grade FROM SC549 WHERE Sno IN

(SELECT S549.Sno Sno

FROM S549 JOIN SC549 SC

ON S549.Sno=SC.Sno

GROUP BY S549.Sno HAVING COUNT(SC.Cno)>=3))

GROUP BY Sno ORDER BY AVG(Grade) DESC LIMIT 1)

分析：

选择学生表 S549 中的学号 (Sno) 和姓名 (Sname)。条件是学生学号必须匹配子查询返回的学号。外层查询的作用是最终返回符合条件的学生的学号和姓名。

从嵌套子查询中选择学号 (Sno)，这个子查询返回平均成绩最高的学生的学号。这个子查询的作用是找到符合条件的学生中成绩最高的学生学号。

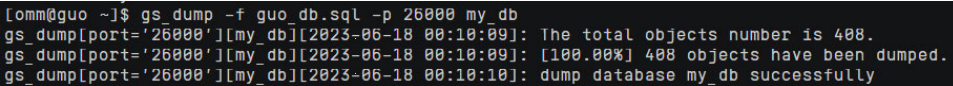
这个嵌套子查询从选课表 SC549 中选择学生学号 (Sno) 和成绩 (Grade)，但仅限于那些选修了 3 门及以上课程的学生。结果按平均成绩降序排列，并限制结果只返回一行。嵌套子查询的作用是筛选出符合条件的学生并计算他们的平均成绩，最终找出平均成绩最高的学生。

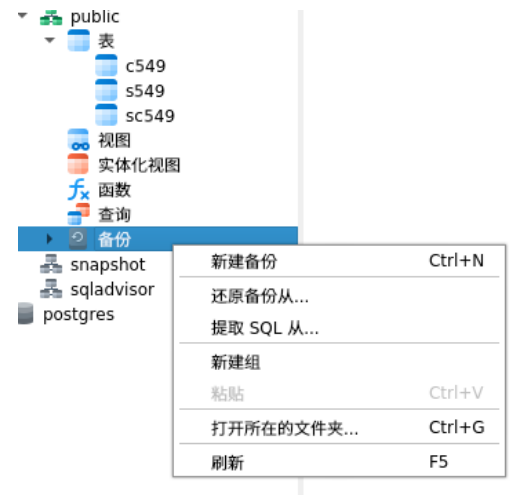
这个子查询从学生表 S549 和选课表 SC549 中选择学号 (S549.Sno)，通过连接操作 (JOIN) 将这两个表按学生学号进行连接。然后使用 GROUP BY 对学号进行分组，并使用 HAVING COUNT(SC.Cno) >= 3 筛选出选修了 3 门及以上课程的学生。嵌套的 IN 子查询的作用是找出所有选修了 3 门及以上课程的学生。

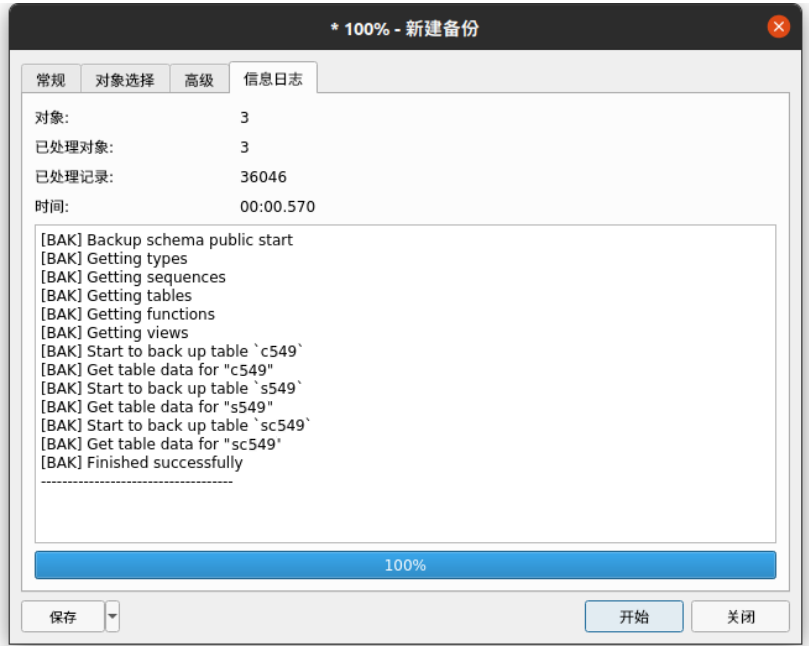
这条SQL语句通过多层嵌套查询，首先筛选出选修了 3 门及以上课程的学生，然后在这些学生中计算每个学生的平均成绩，最后选择平均成绩最高的学生，并返回该学生的学号和姓名。

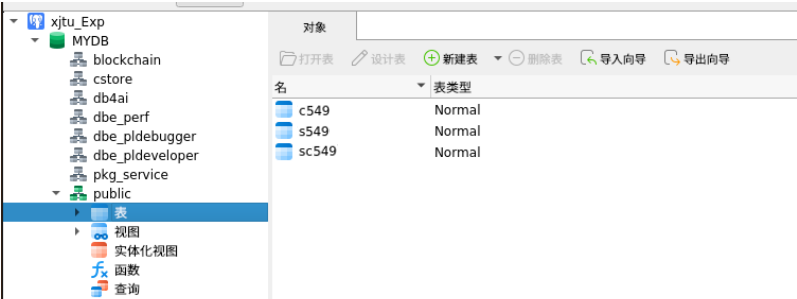
**五、完成上述实验内容后，对数据库进行备份，并交给另一位同学进行恢复 实验。在成功恢复其他同学交付的数据库备份后，分析其表设计合理性及生成 的数据质量，将相应结果截屏图保存，并写入实验报告中**。

对数据库进行备份，并交给余小康同学进行恢复实验









（2）恢复余小康同学的数据库备份

