Contents

实验三 单表查询

实验目的

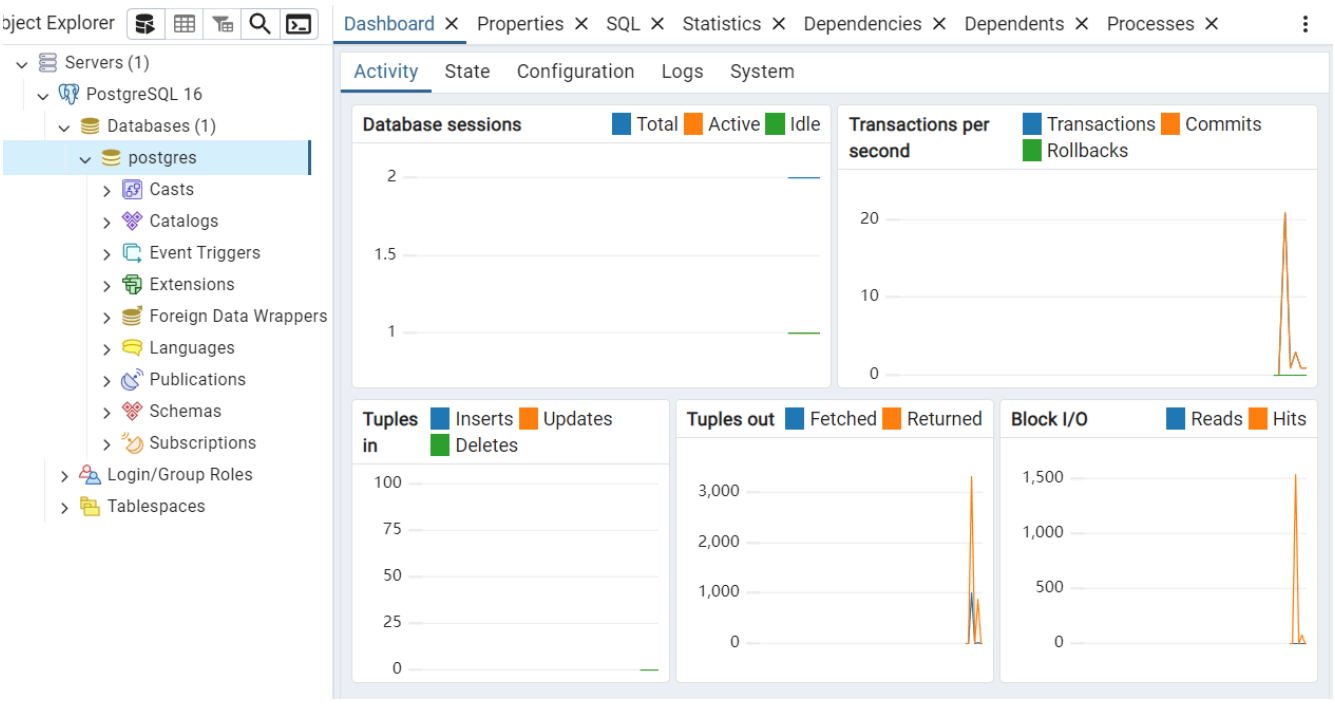
熟悉 SQL 语句的数据查询语言，能够使用 SQL 语句对数据库进行单表查询。

实验环境

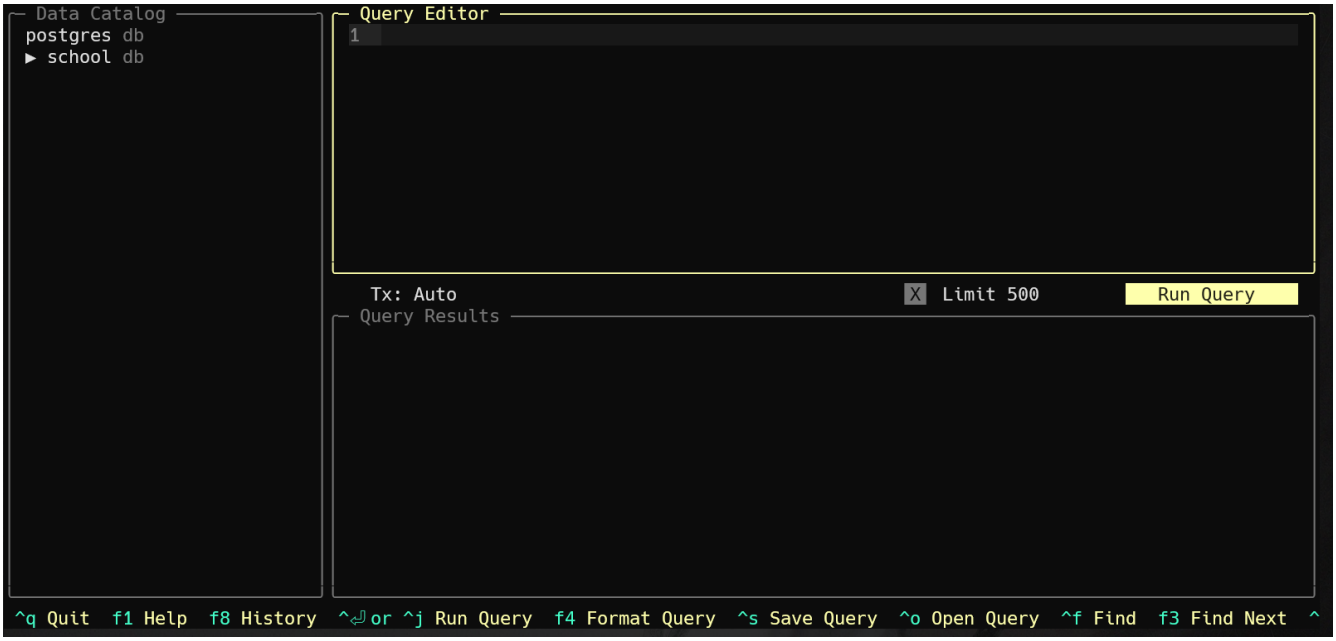
- OS: Windows 11

```
OsName                                    : Microsoft Windows 11 企业版
OsType                                    : WINNT
OsOperatingSystemSKU                      : EnterpriseEdition
OsVersion                                 : 10.0.22631
```

- Database: PostgreSQL 16



- UI: harlequin-postgres



实验内容

1. 查询的目标表达式为所有列、指定列或指定列的运算。
2. 使用 DISTINCT 保留字消除重复行。
3. 对查询结果排序和分组。
4. 集合分组使用集函数进行各项统计。

实验步骤

课内实验

以 school 数据库为例在该数据库中存在 4 张表格，分别为：

STUDENTS(sid,sname,email,grade)

TEACHERS(tid,tname,email,salary)

COURSES(cid,cname,hour)

CHOICES(no,sid,tid,cid,score)

在数据库中，存在这样的关系：学生可以选择课程。一个课程对应一个老师。在表 CHOICES 中保存学生的选课记录。

请按照以下要求编写 SQL 语句：

查询学生的选课成绩合格的课程成绩，并把成绩换算为绩点(60分对应绩点为1,每增加1分，绩点增加0.1)；

查询课时是48或64的课程的名称；

查询所有课程名称中含有data的课程编号；

查询所有选课记录的课程号(不重复显示)；

统计所有老师的平均工资；

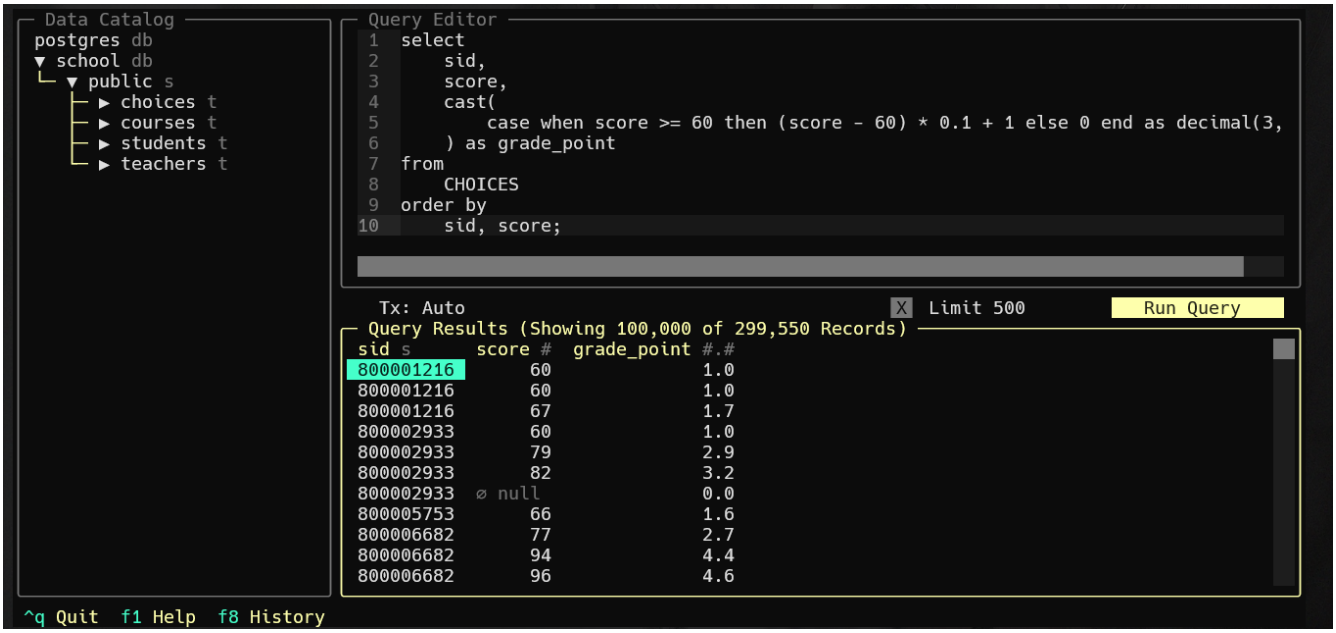
查询所有学生的编号，姓名和平均成绩，按总平均成绩降序排列；

统计各个课程的选课人数和平均成绩；

查询至少选修了三门课程的学生编号。

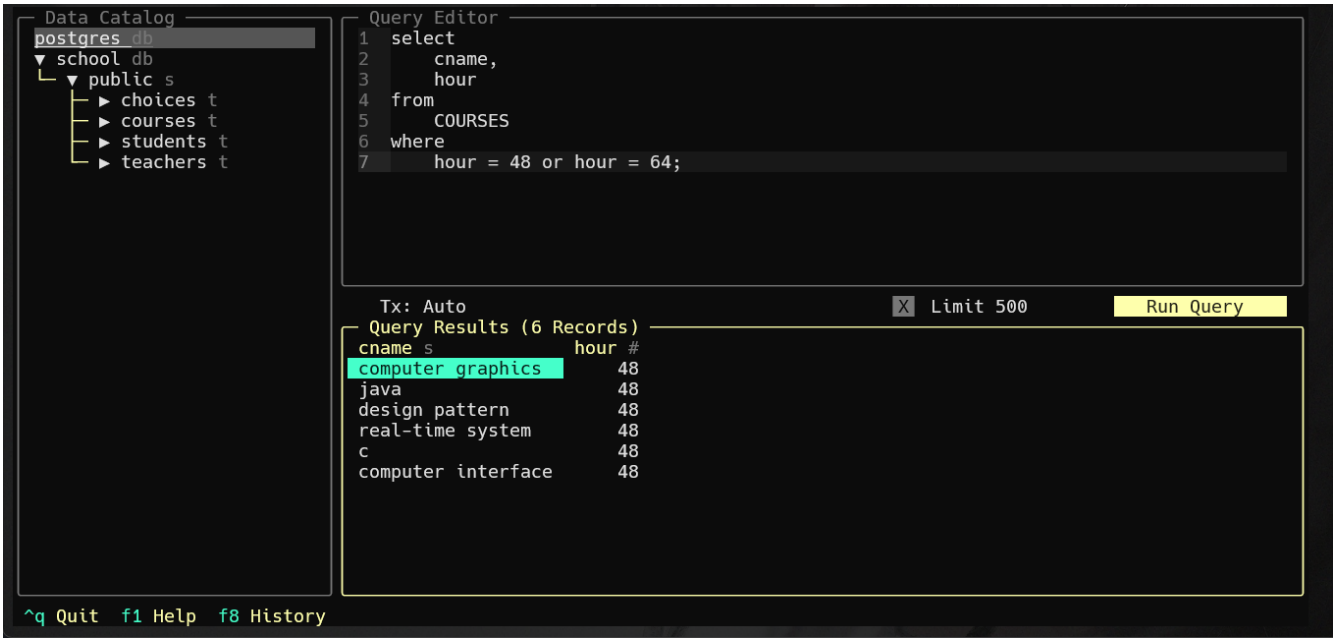
1. 查询学生的选课成绩合格的课程成绩，并把成绩换算为绩点 (60 分对应绩点为 1, 每增加 1 分，绩点增加 0.1)：

```
select
    sid,
    score,
    cast(
        case when score >= 60 then (score - 60) * 0.1 + 1 else 0 end as decimal(3, 1)
    ) as grade_point
from
    CHOICES
order by
    sid, score;
```



2. 查询课时是 48 或 64 的课程的名称：

```
select
    cname,
    hour
from
    COURSES
where
    hour = 48 or hour = 64;
```



3. 查询所有课程名称中含有 data 的课程编号：

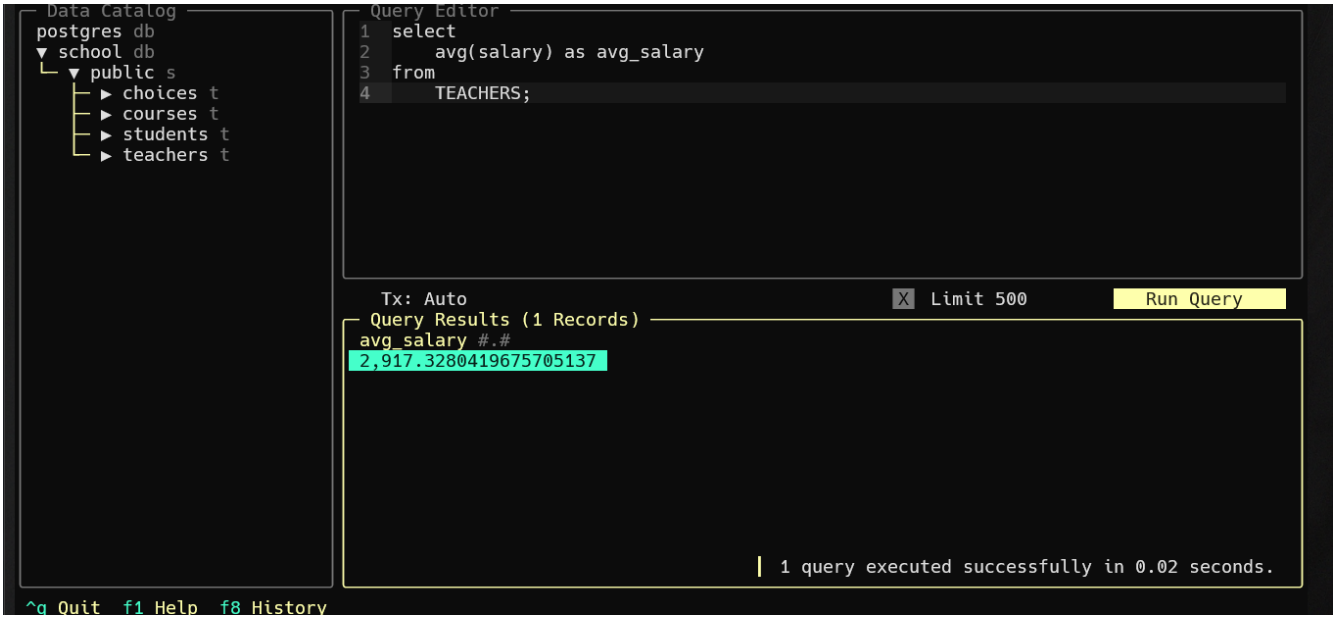
```
select
    cid,
    cname
from
    COURSES
where
    cname like '%data%';
```

4. 查询所有选课记录的课程号 (不重复显示)：

```
select
    distinct cid
from
    CHOICES;
```

5. 统计所有老师的平均工资：

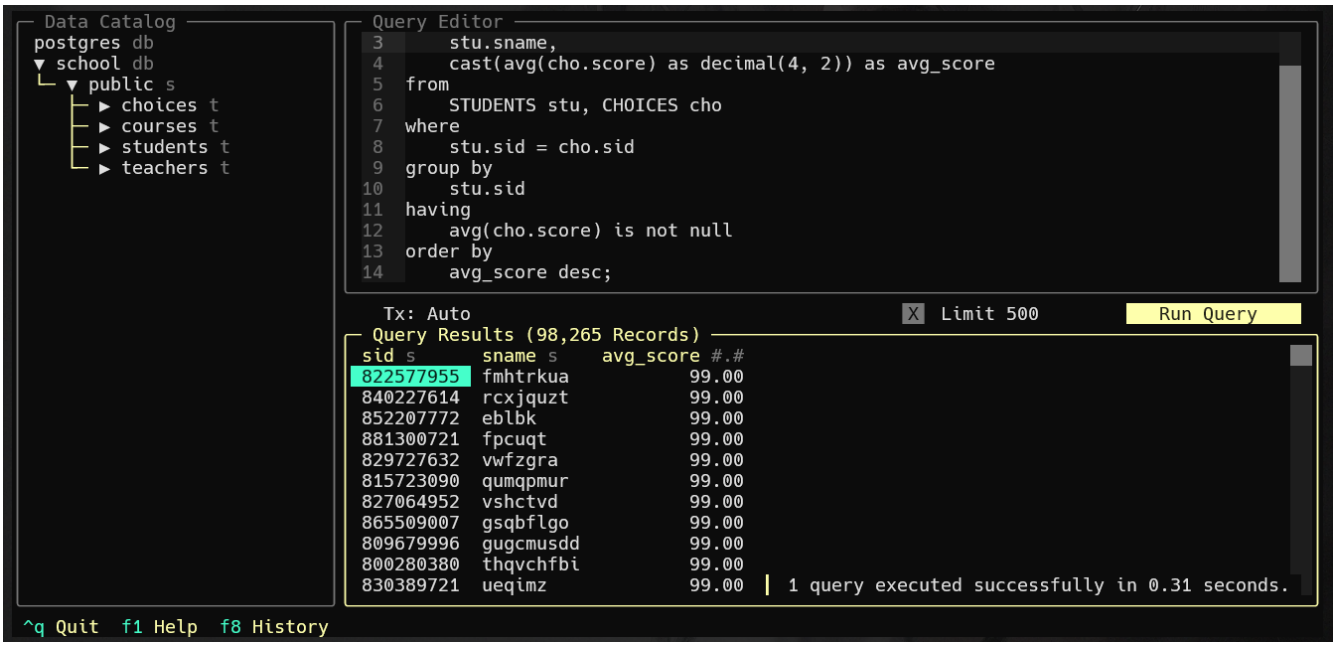
```
select
    avg(salary) as avg_salary
from
    TEACHERS;
```



6. 查询所有学生的编号，姓名和平均成绩，按总平均成绩降序排列：

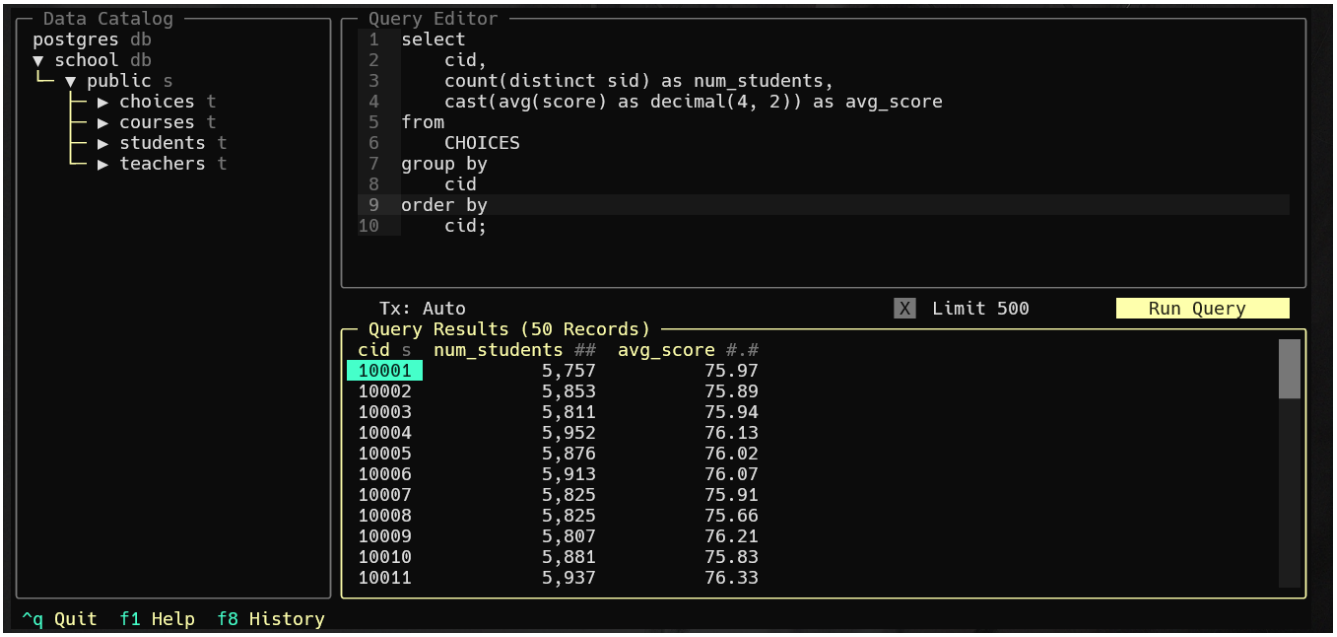
```
select
    stu.sid,
    stu.sname,
    -- avg function will ignore null values,
    -- return null if all values of a column
    -- are null.
    cast(avg(cho.score) as decimal(4, 2)) as avg_score
from
    STUDENTS stu, CHOICES cho
where
```

```
    stu.sid = cho.sid
group by
    stu.sid
having
    -- There are some students who do not have any records,
    -- if we use avg function, the result will be null.
    -- If you don't want to ignore null values,
    -- just remove the "having" clause.
    avg(cho.score) is not null
order by
    avg_score desc;
```



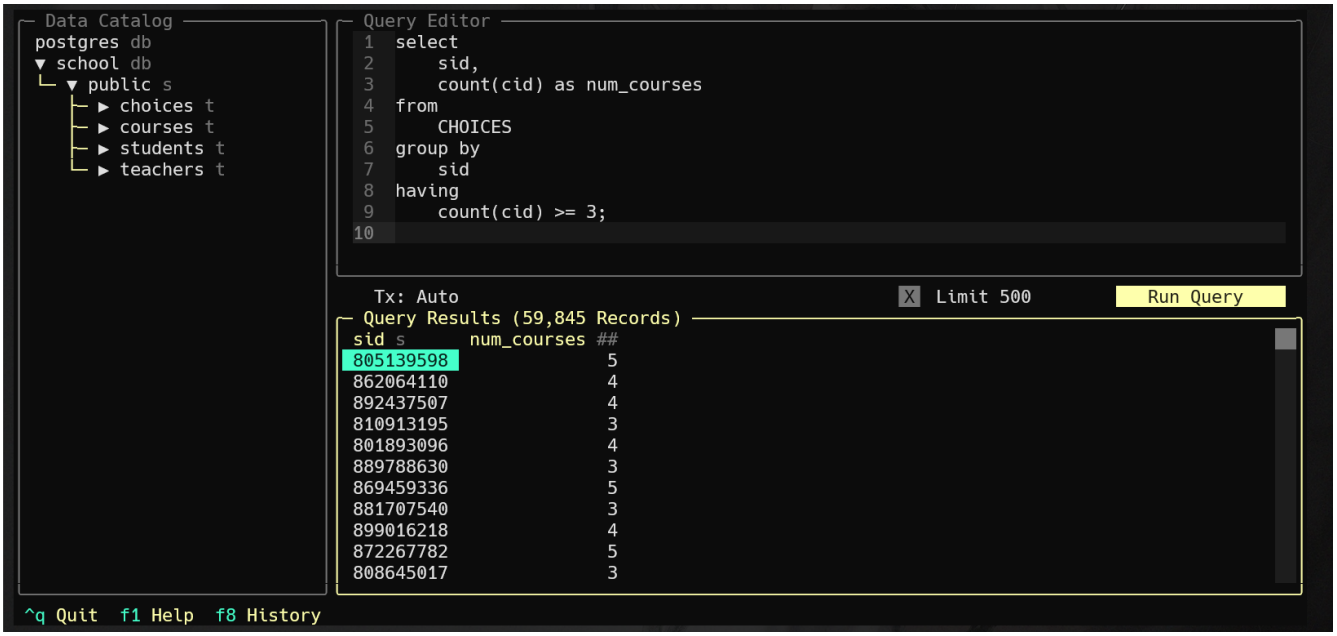
7. 统计各个课程的选课人数和平均成绩：

```
select
    cid,
    -- I don't know why some students can choose the same
    -- course multiple times, so I use "distinct" to eliminate
    -- duplicates. If you don't want to eliminate duplicates,
    -- just remove the "distinct" keyword.
    count(distinct sid) as num_students,
    cast(avg(score) as decimal(4, 2)) as avg_score
from
    CHOICES
group by
    cid
order by
    cid;
```



8. 查询至少选修了三门课程的学生编号：

```
select
    sid,
    count(cid) as num_courses
from
    CHOICES
group by
    sid
having
    count(cid) >= 3;
```



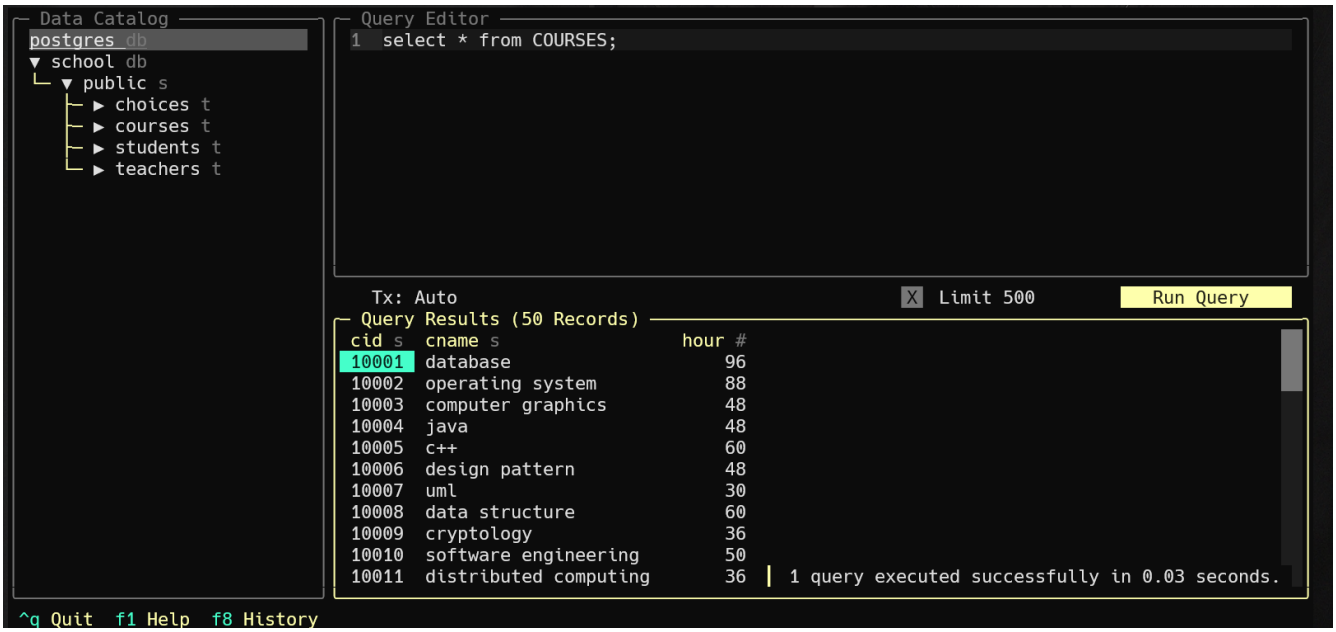
自我实践

查询全部课程的详细记录；
查询所有有选修课的学生的编号；
查询课时<88(hour)的课程的编号；
请找出总分超过400分的学生；

查询课程的总数；
查询所有课程和选修该课程的学生总数；
查询选修成绩合格的课程超过两门的学生编号；
统计各个学生的选修课程数目和平均成绩；

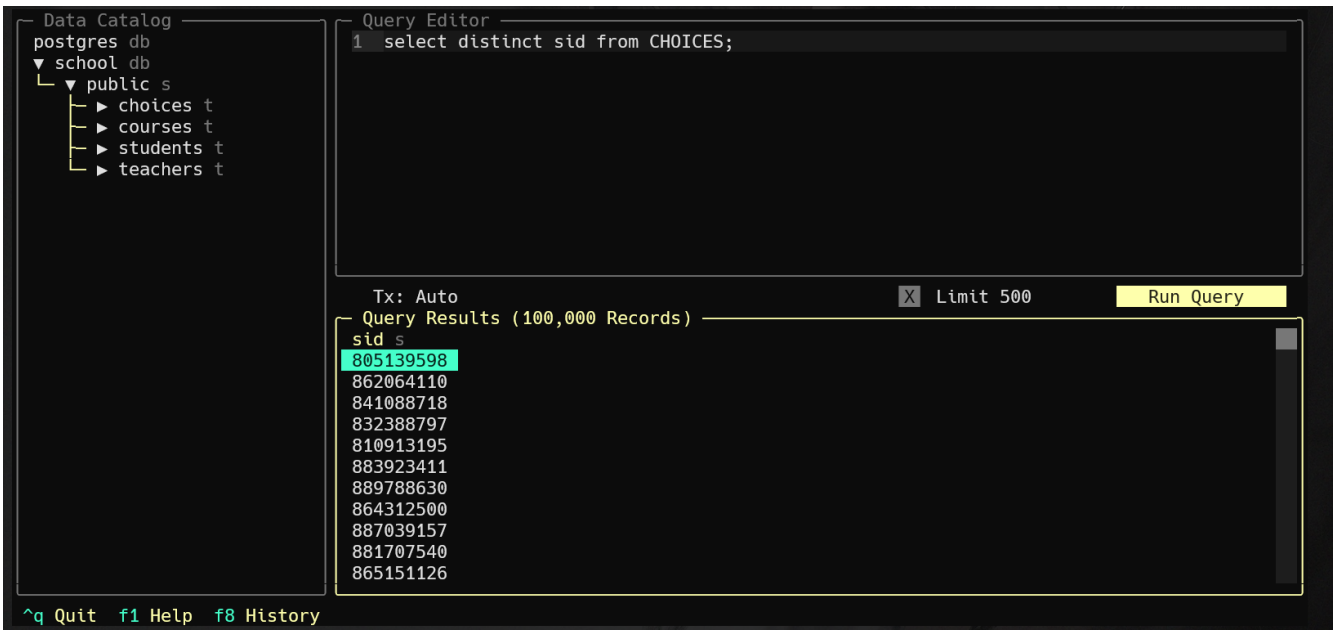
1. 查询全部课程的详细记录：

```
select * from COURSES;
```



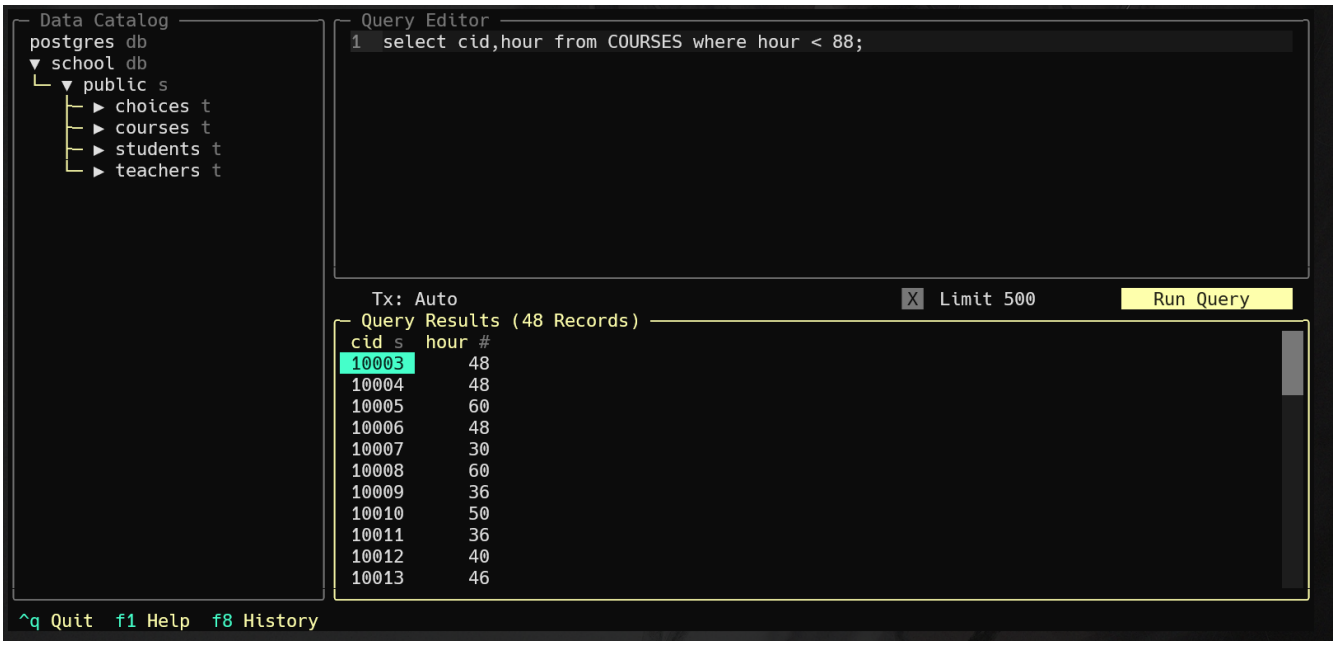
2. 查询所有有选修课的学生的编号：

```
select distinct sid from CHOICES;
```



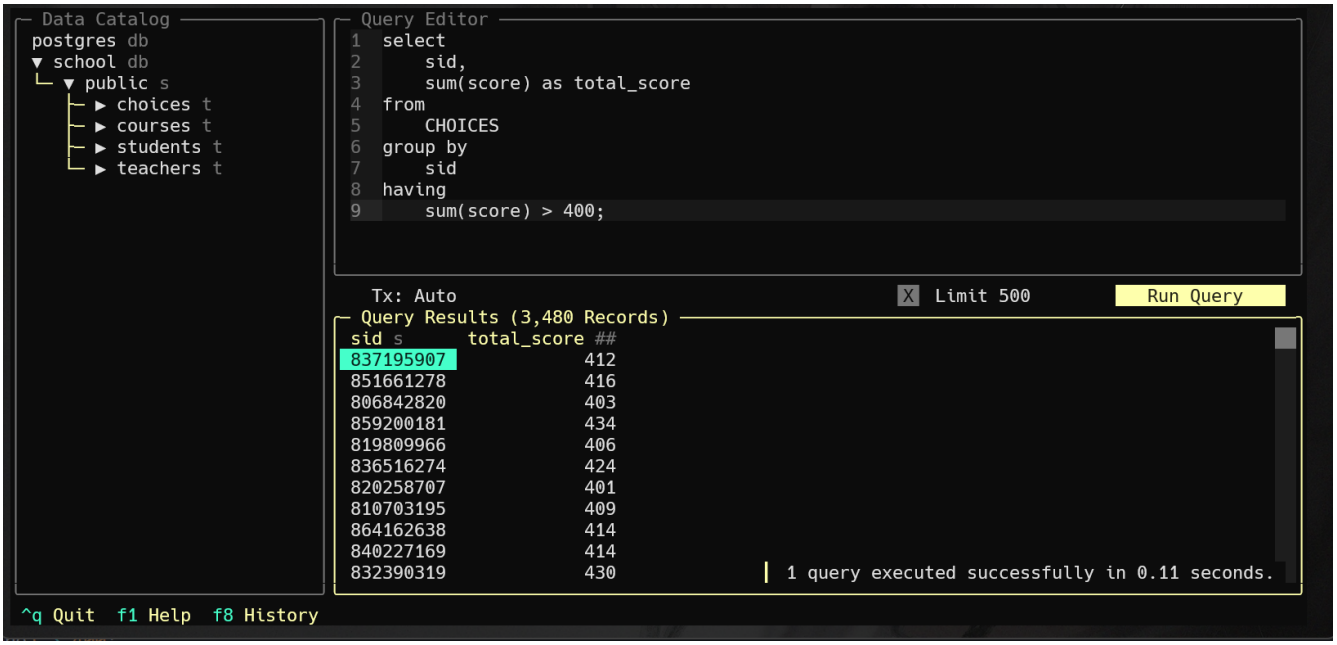
3. 查询课时 <88(hour) 的课程的编号：

```
select cid from COURSES where hour < 88;
```



4. 请找出总分超过 400 分的学生：

```
select
    sid,
    sum(score) as total_score
from
    CHOICES
group by
    sid
having
    sum(score) > 400;
```



5. 查询课程的总数：

```
select count(*) as total_courses from COURSES;
```

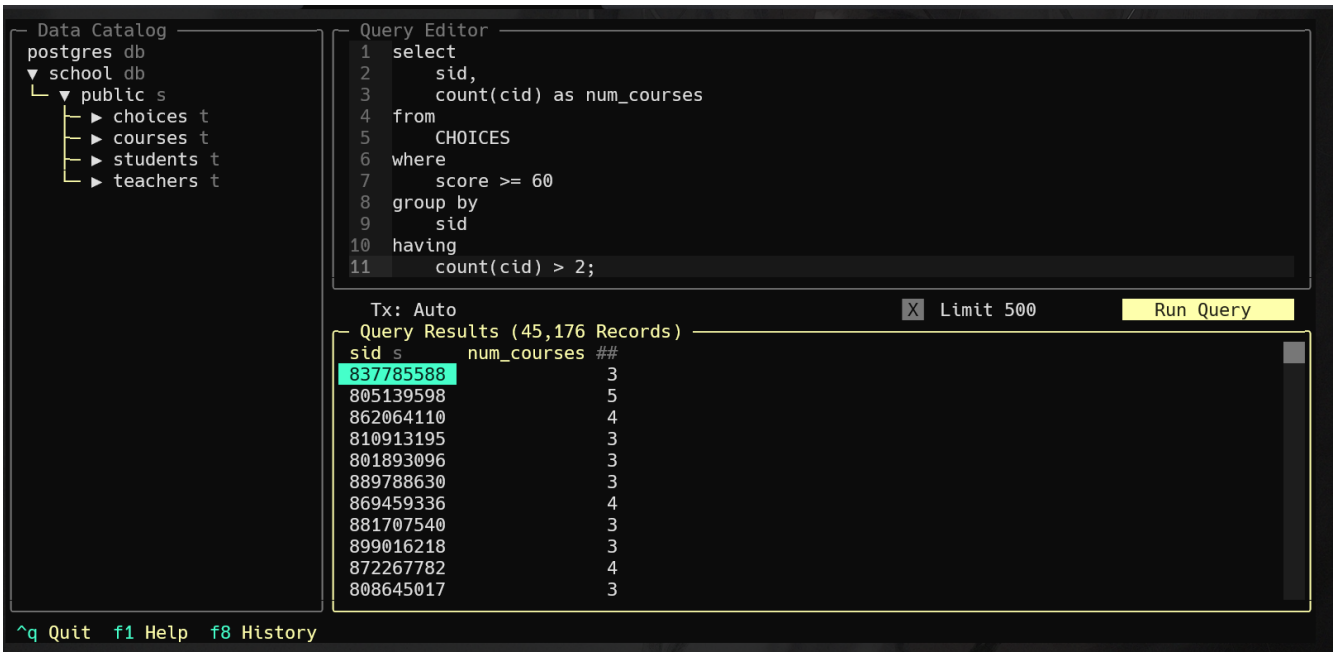
6. 查询所有课程和选修该课程的学生总数：

```
select
    cid, count(distinct sid) as num_students
from
    CHOICES
group by
    cid;
```

7. 查询选修成绩合格的课程超过两门的学生编号：

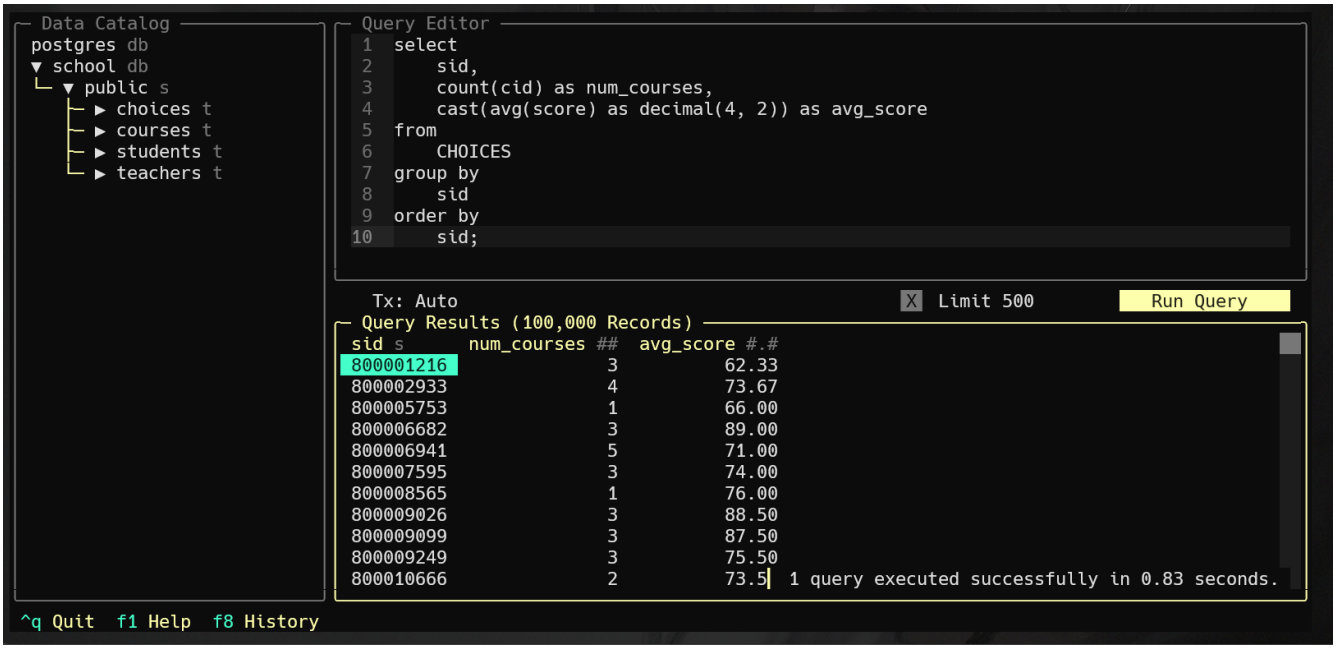
```
select
    sid,
    count(cid) as num_courses
from
    CHOICES
where
    score >= 60
```

```
group by
    sid
having
    count(cid) > 2;
```



8. 统计各个学生的选修课程数目和平均成绩：

```
select
    sid,
    count(cid) as num_courses,
    cast(avg(score) as decimal(4, 2)) as avg_score
from
    CHOICES
group by
    sid
order by
    sid;
```


Data Catalog
postgres db
▼ school db
▼ public s
▶ choices t
▶ courses t
▶ students t
▶ teachers t
Query Editor
1 select
2 sid,
3 count(cid) as num_courses,
4 cast(avg(score) as decimal(4, 2)) as avg_score
5 from
6 CHOICES
7 group by
8 sid
9 order by
10 sid;
Tx: Auto
X Limit 500
Run Query
Query Results (100,000 Records)
sid s num_courses ## avg_score #.#
800001216 3 62.33
800002933 4 73.67
800005753 1 66.00
800006682 3 89.00
800006941 5 71.00
800007595 3 74.00
800008565 1 76.00
800009026 3 88.50
800009099 3 87.50
800009249 3 75.50
800010666 2 73.5
1 query executed successfully in 0.83 seconds.
^q Quit f1 Help f8 History