中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

学生姓名：　丁晓琪

学生学号：　22336057

专业名称：　计科

一：实验要求

(1)查询全部课程的详细记录;
(2)查询所有有选修课的学生的编号;
(3)查询课时<88(小时)的课程的编号;
(4)请找出总分超过400分的学生;
(5)查询课程的总数;
(6)查询所有课程和选修该课程的学生总数;
(7)查询选修成绩超过60的课程超过两门的学生编号;
(8)统计各个学生的选修课程数目和平均成绩;
(9)查询选修Java的所有学生的编号及姓名;
(10)查询姓名为sssht的学生所选的课程的编号和成绩;
(11)查询其他课时比课程C++多的课程的名称

二：实验过程

背景：

- students (sid, sname , email , grade)
- teachers (tid, tname , email , salary)
- courses (cid, cname , hour)
- choices (no, sid , tid , cid , score)

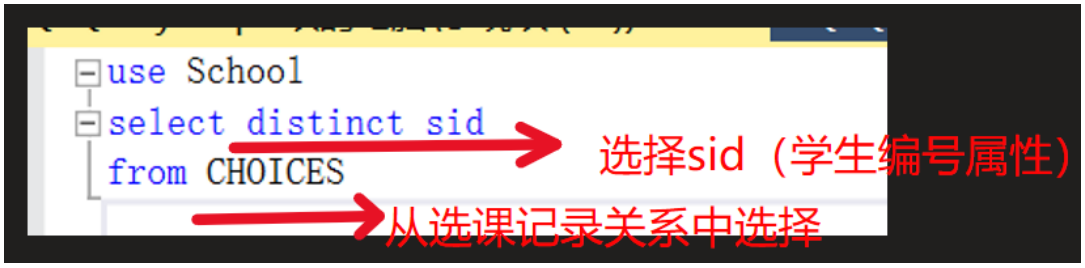
1. 查询全部课程的详细记录：

   - 运行结果：得到所有课程的相关信息（包括`cid`,`cname`,`hour`）

-

	cid	cname	hour
1	10001	database	96
2	10002	operating system	88
3	10003	computer graphics	48
4	10004	java	48
5	10005	c++	60
6	10006	design pattern	48
7	10007	uml	30
8	10008	data structure	60
9	10009	cryptology	36
10	10010	software engineering	50
11	10011	distributed computing	36
12	10012	erp	40
13	10013	artifical intelligence	46
14	10014	computer network	60
15	10015	tcp/ip protocol	68
16	10016	data mining	40
17	10017	algorithm	72

2. 查询所有有选修课的学生的编号：
   - 由于可能有一个学生选多门课的记录，所以需要对学生编号去重

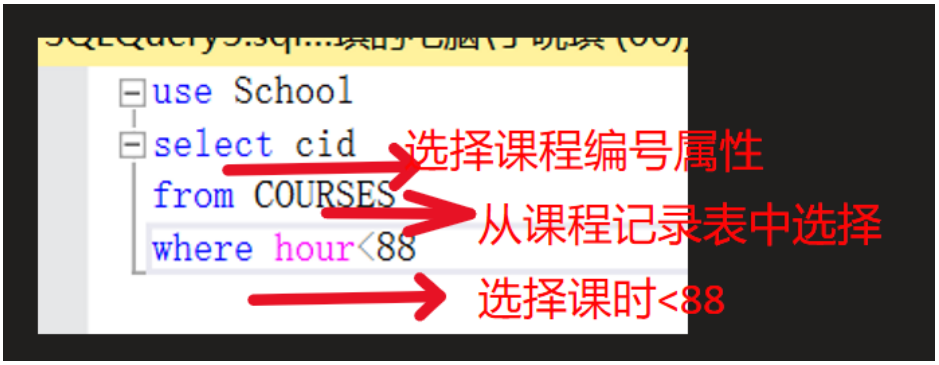


- 运行结果：得到一列多行的关系

结果 消息

	sid
1	801855166
2	827469796
3	884069270
4	828045342
5	883347944
6	891664888
7	837955959
8	835009071
9	873055545
10	860947076
11	843642926
12	857329894
13	883737481
14	872847191
15	881648723
16	884805325
17	861532990
18	888669809
19	880438701

3. 查询课时<88(小时)的课程的编号：

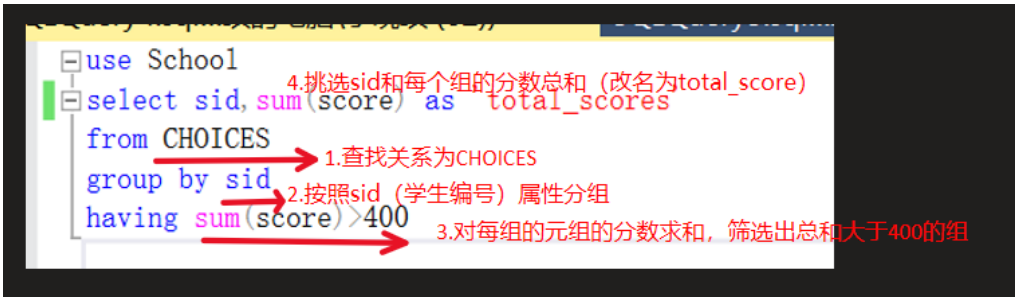


- 运行结果：获得课时小于88的所有课程的编号(`cid`)

结果 消息

	cid
1	10003
2	10004
3	10005
4	10006
5	10007
6	10008
7	10009
8	10010
9	10011
10	10012
11	10013
12	10014
13	10015
14	10016
15	10017
16	10018
17	10019
18	10020
19	10021

4. 请找出总分超过400分的学生：



- 运行结果：得到两列的表，`total_scores`为学生再所有选课上的总分，`sid`为学生编号

结果 消息

	sid	total_scores
1	832910376	411
2	810362785	417
3	889057524	424
4	860617178	441
5	859164164	428
6	882537262	401
7	860489114	413
8	877301330	416
9	829079479	430
10	885651239	411
11	824332034	422
12	818921465	427
13	830772269	412
14	859834462	404
15	874270136	410
16	812821020	401
17	823087852	415
18	807795688	424
19	831096911	429

5. 查询课程的总数：

- `from`子句表示基于`COURSES`关系表，使用聚合函数`count`计算属性`cid`的数量，也即计算课程总数

```
use School
select count(cid) as 'count'
from COURSES
```

- 运行结果：得到一行一列的关系表（聚合函数的输出就为一个值），课程总数为50

结果 消息

	count
1	50

6. 查询所有课程和选修该课程的学生总数 ：

```
use School
select cid,count(sid) as 'count_student'
from CHOICES
group by cid
```

3.挑选出属性cid和count_student(使用函数count计算的每组sid的数量)

1.查找关系为CHOICES

2.按照sid（学生编号）属性分组

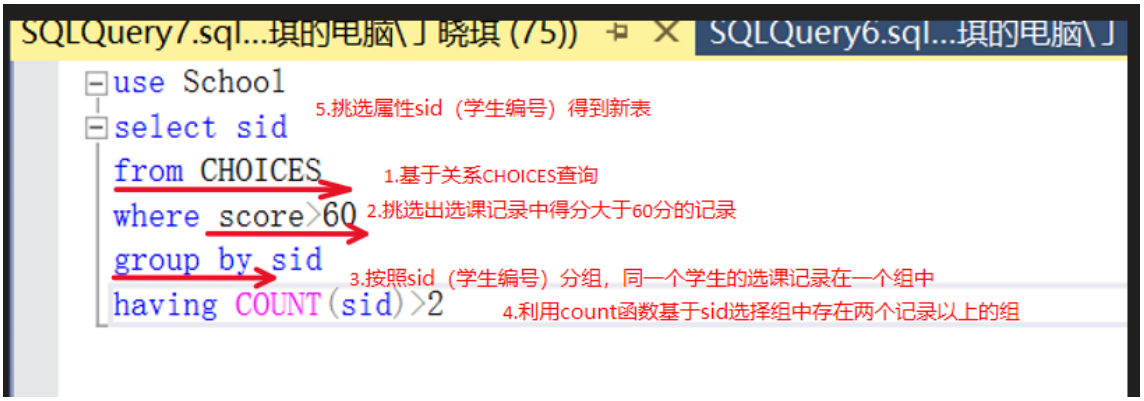
- 运行结果：得到50行2列的关系表，对应着50个课程的编号和选修的学生总数

结果 消息

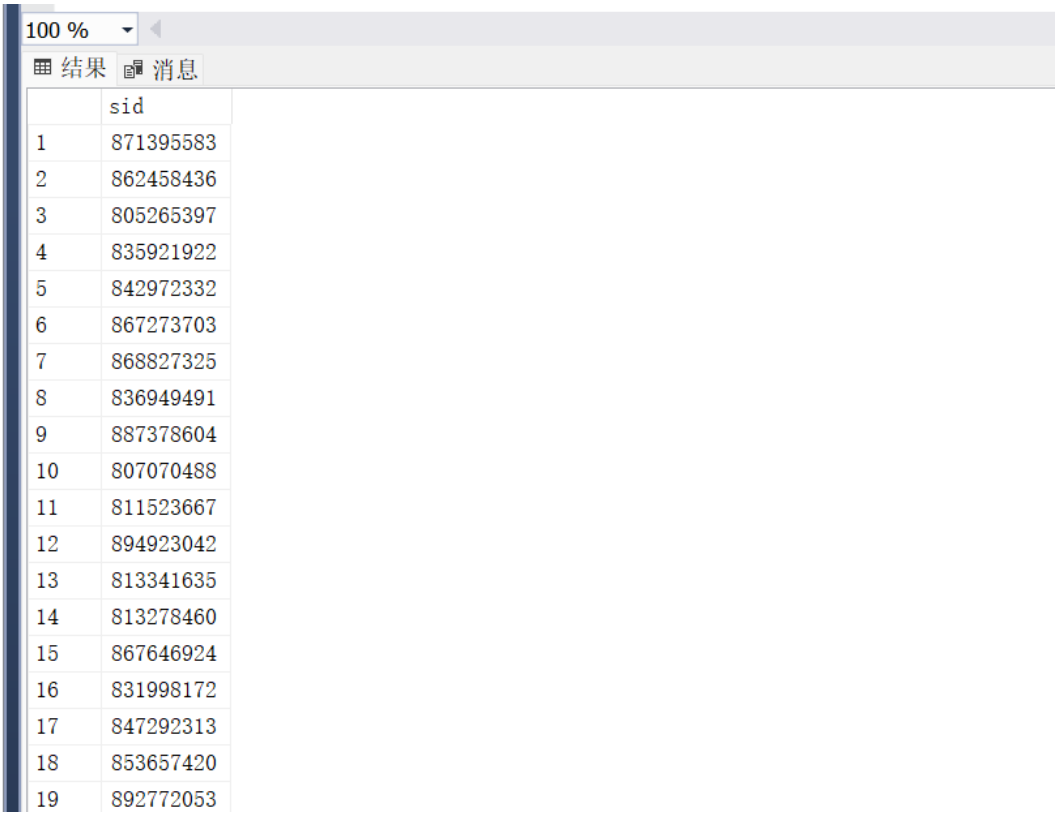
	cid	count_student
1	10008	5985
2	10019	6074
3	10018	5969
4	10040	6102
5	10011	6086
6	10028	6042
7	10035	6104
8	10021	5916
9	10046	6069
10	10032	6073
11	10005	6031
12	10050	6077
13	10034	6000
14	10015	5990
15	10036	6043
16	10045	6041
17	10025	5955
18	10030	5985
19	10049	5945

7. 查询选修成绩超过60的课程超过两门的学生编号:

- 先挑选出得分大于60的记录，再按学生把记录分组，然后再挑选选课记录大于2的组



- 运行结果：

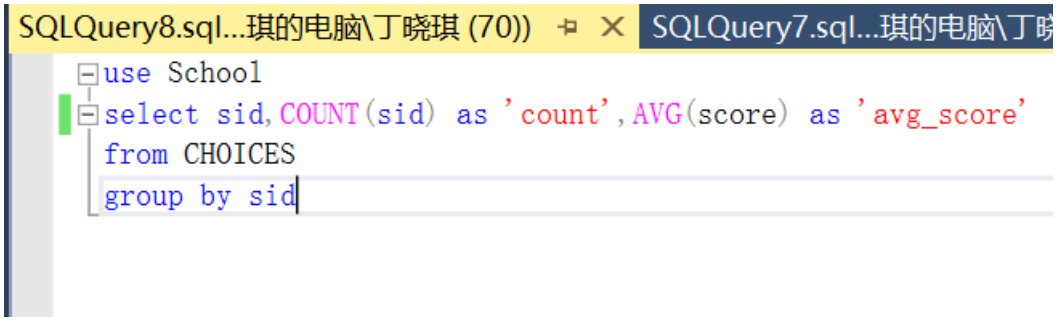

100 %

结果 消息

	sid
1	871395583
2	862458436
3	805265397
4	835921922
5	842972332
6	867273703
7	868827325
8	836949491
9	887378604
10	807070488
11	811523667
12	894923042
13	813341635
14	813278460
15	867646924
16	831998172
17	847292313
18	853657420
19	892772053

8. 统计各个学生的选修课程数目和平均成绩：

- `from`子句表明查询关系为`CHOICES`，`group by`子句表示按照学生编号`sid`分组（同一学生的选课记录在一组），`select`子句表示选择`sid`，`COUNT(sid)`（每个学生的选课记录数也即选课总数），`AVG(score)`（每个学生选课的所有课程的平均分）
- 注意：当分数为`null`时，`AVG`函数自动跳过。但是`count`函数不会跳过


SQLQuery8.sql...琪的电脑\丁晓琪 (70)) SQLQuery7.sql...琪的电脑\丁晓

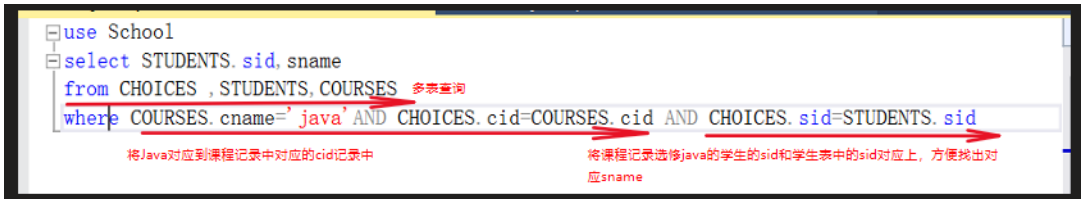
```
use School
select sid,COUNT(sid) as 'count',AVG(score) as 'avg_score'
from CHOICES
group by sid
```

- 运行结果：

结果 消息

	sid	count	avg_score
1	840741597	3	75
2	852548704	3	84
3	812917218	5	77
4	838235261	5	75
5	833074041	5	88
6	831362259	3	78
7	819452029	1	85
8	884999156	1	93
9	847030674	5	70
10	891770368	3	74
11	815830773	4	80
12	825769835	5	79
13	818473409	5	71
14	802890952	5	75
15	889351393	4	85
16	810581631	4	81
17	808999533	3	85
18	835480095	4	62
19	843500370	4	78

9. 查询选修Java的所有学生的编号及姓名：
   - 选修课程的记录由关系`CHOICES`提供，但是它只能查询编号`cid,sid`，无法得到课程名称和学生姓名，所以需要和`STUDENTS,COURSES`多个表查询



   - 运行结果：


100 %

结果 消息

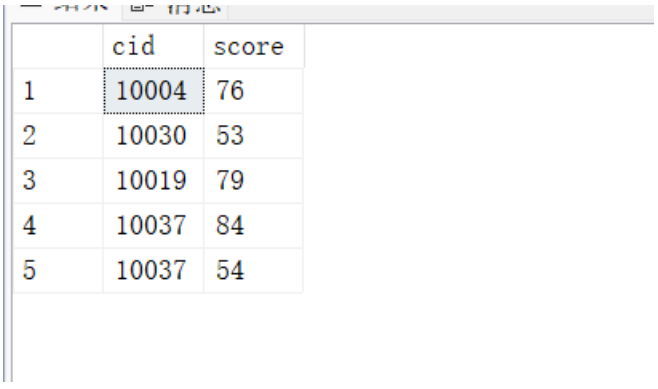
	sid	sname
1	821211846	aghoxgtrn
2	821281722	kdodaof
3	821657680	kndjwxs
4	821905381	fqswtfqtx
5	822034708	fwbhyks
6	822415676	qyspi
7	822419647	gokcwd
8	822472664	npaqquxnc
9	822519306	oopil
10	822633076	ihsnsu
11	822722979	kbpqahkv
12	822846524	thkqimgo
13	822848737	wqdquhwsk
14	823041239	jmwdw
15	823319107	qcmvc
16	823801128	tjuoklew
17	823810473	yryocex
18	823938594	lbzfkys
19	824156429	fskqa

10. 查询姓名为sssht的学生所选的课程的编号和成绩：

   - 多课表查询：通过`where`子句中的`STUDENTS.sname='sssht'`挑选出`STUDENTS`中的sssht的学生记录，然后`STUDENTS.sid=CHOICES.sid`拿sssht的学生编号筛选选课记录中ssht的选课记录

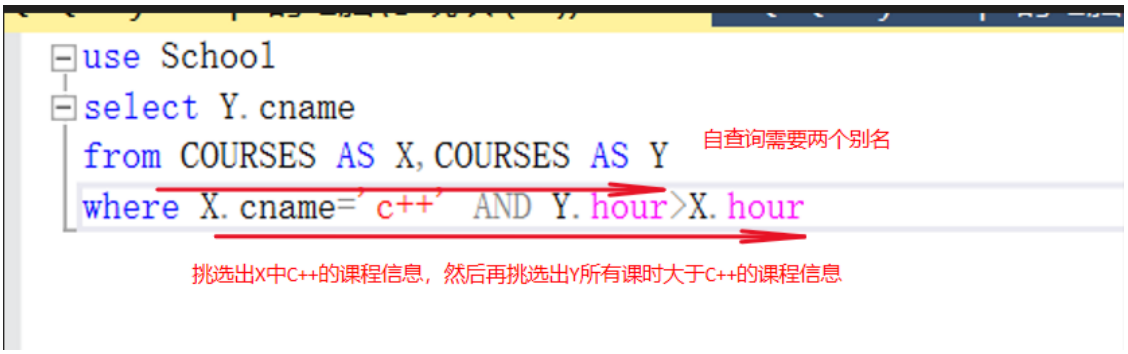
```
use School
select cid,score
from CHOICES,STUDENTS
where STUDENTS.sname= 'sssht' AND STUDENTS.sid=CHOICES.sid
```

   - 运行结果：

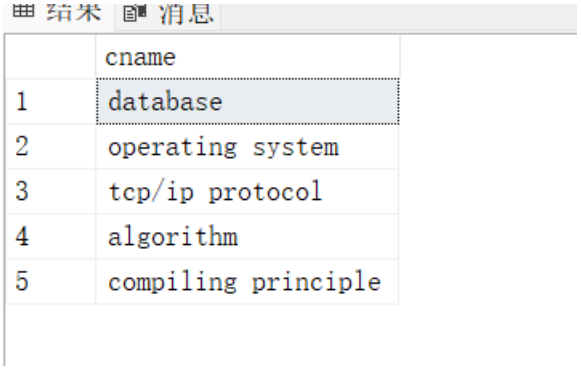


	cid	score
1	10004	76
2	10030	53
3	10019	79
4	10037	84
5	10037	54

11. 查询其他课时比课程C++多的课程的名称:



- 运行结果：



	cname
1	database
2	operating system
3	tcp/ip protocol
4	algorithm
5	compiling principle

三：总结

1. 基本查询结构：

```
select #查询得到的元组得到的新关系组成属性
from #查询的关系名称
where #查询元组需要满足的谓词关系
```

2. 分组查询，分组选择谓词：

```
group by #分组查询，参数为属性
having #分组后对组的选择谓词条件（where是对元组）
#（注意select和having要不出现在group by中，要不被聚集）
```

3. 聚集函数：（参数为属性）

```
count #计算属性的元组个数 count(*)计算总行数
sum   #计算关系在对应属性上的元组和
avg   #计算关系在对应属性上的元组平均值
min   ##计算关系在对应属性上的最小值
max   #计算关系在对应属性上的元组最大值
```

4. null值：除了`count(*)`外所有聚集函数都忽略输入中的`null`。
5. `as`： 别名可用于`select`子句对属性改名和`from`子句中对自查询的关系赋予多个别名
6. `distinct`：去重

7. 字符串查询：

- 特殊字符：% 匹配任意子串，_ 匹配任意一个字符，\ 转义符（\%表示真正的字符串）
- 模式匹配：（大小写敏感）
  - ‘Intro%’：匹配以Intro开头的字符串
  - '_ _ _' 匹配任意三个字符的字符串
  - ‘_ _%’匹配至少有两个字符的字符串
  - ‘%Comp%’:匹配含有Comp的字符串

8. 外连接`join..on`