实验3 SQL高级查询的应用

* **实验环境：**

操作系统：Windows 10 专业版

数据库管理系统：8.0.18 MySQL Community Server - GPL

客户端：MySQL Workbench 8.0 Community+VScode 1.71.2

数据库模式：Test2

学生表：Student(Sno,Sname,Ssex,Sage,Sdept)

课程表：Course(Cno,Cname,Cpno,Ccredit)

选课表：SC(Sno,Cno,Grade)

数据库的值：如下图所示



1. 在学生选课表中，试用SUM、AVG、COUNT函数统计所有学生的总分、平均分、课程门数。

**实验代码：**

-- 1、在学生选课表中，统计所有学生的总分、平均分、课程门数。

select sum(Grade) as sum, -- 查询值重命名用as

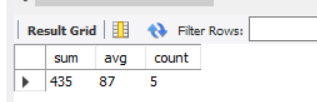
       floor(avg(Grade)) as avg, -- 保留整数用floor(属性)

       count(\*) as count

from Sc;

**运行结果：**

正常而言使用平均函数返回值为保留四位小数的浮点数，实验中加以floor限制实现了下面输出整数的结果。





2、在学生表和学生选课表中，查询学生的学号、姓名、成绩、课程编号信息。

**实验代码：**

-- 2、在学生表和学生选课表中，查询学生的学号、姓名、成绩、课程编号信息。

select Student.Sno,Sname,Grade,Cno

from Student left outer join Sc on(Student.Sno=Sc.Sno) -- 左外连接

order by Student.Sno -- 注意参考结果里是升序排列

**运行结果：**

注意参考的运行结果里王敏和张立的输出顺序，显然是增添了升序排列的限制。





3、在学生表、学生选课表、课程表中，查询学生的学号、姓名、课程编号、课程名称和课程成绩信息。

**实验代码：**

-- 3、三表查询学生的学号、姓名、课程编号、课程名称和课程成绩信息。

select Student.Sno,Sname,Course.Cno,Cname,Grade

from Student

left outer join Sc on(Student.Sno=Sc.Sno) -- 外连接

left outer join Course on(Sc.Cno=Course.Cno) -- 外连接

order by Student.Sno -- 注意参考结果里是升序排列

**运行结果：**



4、在学生选课表和学生表中，查询性别为女的所有学生成绩记录。

**实验代码：**

方法一（内连接）：

-- 在学生选课表和学生表中，查询性别为女的所有学生成绩记录。

select Sc.\*,Student.\*

from Sc,Student

where Sc.Sno=Student.Sno and Student.Ssex='女'; -- 内连接

方法二（外连接）：

select Sc.\*,Student.\*

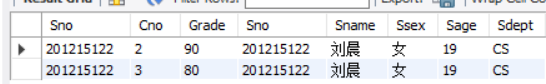
from Sc

left outer join Student on(Student.Sno=Sc.Sno) -- 外连接

where Student.Ssex='女';

**运行结果：**

两种方法运行结果相同，主要原因在于没有空值出现。

5、在学生选课表中，统计学号为201215121的学生的成绩总分、成绩平均分、所修课程门数。

**实验代码：**

-- 在学生选课表中，统计学号为201215121的学生的成绩总分、成绩平均分、所修课程门数。

select sum(Grade) as Sum,

       floor(avg(Grade)) as avg, -- 取整

       count(Cno) as count

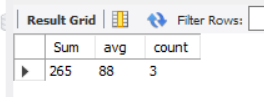
from Sc

where Sno='201215121'

**运行结果：**

注意输出的平均分为83，如果不加round保留整数则结果为83.3333

这里应添加round限制均值





6、查询平均分最高的10个学生成绩信息，包括学生平均分、课程门数、最高分、最低分字段。

**实验代码：**

-- 6、查询平均分最高的10个学生成绩信息，包括学生平均分、课程门数、最高分、最低分字段。

select floor(avg(grade)) as Avg, -- 记得取整

       count(\*) as Count,

       max(grade) as Max,

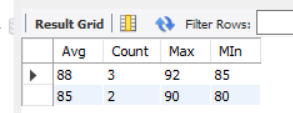
       min(grade) as MIn

from sc

group by sc.Sno -- 同学号为同一组

order by grade desc limit 10 -- 降序排列取前10

**运行结果：**





7、在学生选课表和学生表中，统计各学生所有课程平均分、总分、最高分、最低分、所修课程门数，包括学生学号、学生姓名、平均分、总分、最高分、最低分、所修课程门数字段，并按平均分高低排序。

**实验代码：**

-- 7、在学生选课表和学生表中

-- 统计各学生所有课程平均分、总分、最高分、最低分、所修课程门数，

-- 包括学生学号、学生姓名、平均分、总分、最高分、最低分、所修课程门数字段

-- 并按平均分高低排序。

select student.Sno,

       Sname,

       floor(avg(grade)) as Avg,

       sum(grade) as sum,

       max(grade) as Max,

       min(grade) as Min,

       count(\*) as Count

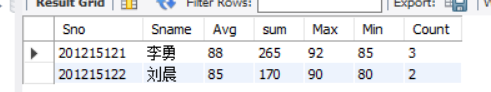
from student,sc

where student.Sno = sc.Sno -- 内连接

group by student.Sno

order by Avg(grade) desc; -- 按平均分降序排列

**运行结果：**



8、在学生选课表和课程表中，统计各门课程平均分、所修人数、课程最高分、课程最低分，包括课程编号、课程名称、课程平均分、课程最高分、课程最低分、所修人数字段，并按平均分高低排序。

**实验代码：**

-- 8、在学生选课表和课程表中，统计各门课程平均分、所修人数、课程最高分、课程最低分

-- 包括课程编号、课程名称、课程平均分、课程最高分、课程最低分、所修人数字段

-- 并按平均分高低排序。

select course.Cno,Cname,

       floor(avg(grade)) as Avg,

       count(sc.Cno) as Count,

       max(grade) as max,

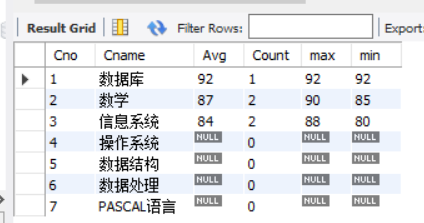
       min(grade) as min

from course left join sc on(sc.Cno=course.Cno) -- 外连接

group by course.Cno

order by avg(grade) desc; -- 按照平均分降序排列

**运行结果：**





* 实验总结：

1.学会使用SQL自带的聚集函数例如avg、sum、count等增强检索功能。

2.当我们计算平均值的时候，其返回值变成了浮点数，如果我们想要保留整数就要考虑floor或者round，前者不考虑四舍五入，后者需要四舍五入。

3.我们需要将某一个人/课程的所有信息作为一组的时候，需要加以group by的限制。

4.当某个学生没有选课，我们仍需把Student的悬浮元组保存在结果关系中，SC的属性填空值NULL时，就需要用外连接；不需要的话则内连接即可。

5.选择多表查询的时候一定要连接，否则返回的就为笛卡尔积了。

6.仔细审题，好好观察示例结果的内容，有的属性名字是需要重命名的，而有的数据是需要取整的。