MySQL练习题

1查询"01"课程比"02"课程成绩高的学生的信息及课程分数

select st.\*,sc.s\_score as '语文' ,sc2.s\_score '数学'

from student st

left join score sc on sc.s\_id=st.s\_id and sc.c\_id='01'

left join score sc2 on sc2.s\_id=st.s\_id and sc2.c\_id='02'

where sc.s\_score>sc2.s\_score

2、查询"01"课程比"02"课程成绩低的学生的信息及课程分数

select st.\*,sc.s\_score '语文',sc2.s\_score '数学' from student st

left join score sc on sc.s\_id=st.s\_id and sc.c\_id='01'

left join score sc2 on sc2.s\_id=st.s\_id and sc2.c\_id='02'

where sc.s\_score<sc2.s\_score

-- 3、查询平均成绩大于等于60分的同学的学生编号和学生姓名和平均成绩

select st.s\_id,st.s\_name,ROUND(AVG(sc.s\_score),**2**) cjScore from student st

left join score sc on sc.s\_id=st.s\_id

group by st.s\_id having AVG(sc.s\_score)>=**60**

-- 5、查询所有同学的学生编号、学生姓名、选课总数、所有课程的总成绩

select st.s\_id,st.s\_name,count(c.c\_id),( case when SUM(sc.s\_score) is null or sum(sc.s\_score)="" then **0** else SUM(sc.s\_score) end) from student st

left join score sc on sc.s\_id =st.s\_id

left join course c on c.c\_id=sc.c\_id

group by st.s\_id

case语句：

CASE case\_value

WHEN when\_value THEN

statement\_list

ELSE

statement\_list

END CASE;

-- 6、查询"李"姓老师的数量

select t.t\_name,count(t.t\_id) from teacher t

group by t.t\_id having t.t\_name like "李%";

7查询学过"张三"老师授课的同学的信息

SELECT st.\*

FROM student st

LEFT JOIN score sc ON sc.s\_id = st.s\_id

LEFT JOIN course c ON sc.c\_id = c.c\_id

LEFT JOIN teacher t ON c.t\_id = t.t\_id

WHERE t.t\_name='张三'

-- 8、查询没学过"张三"老师授课的同学的信息

-- 张三老师教的课

select c.\* from course c left join teacher t on t.t\_id=c.t\_id where t.t\_name="张三"

-- 有张三老师课成绩的st.s\_id

select sc.s\_id from score sc where sc.c\_id in (select c.c\_id from course c left join teacher t on t.t\_id=c.t\_id where t.t\_name="张三")

-- 不在上面查到的st.s\_id的学生信息,即没学过张三老师授课的同学信息

select st.\* from student st where st.s\_id not in(

select sc.s\_id from score sc where sc.c\_id in (select c.c\_id from course c left join teacher t on t.t\_id=c.t\_id where t.t\_name="张三")

)

-- 10、查询学过编号为"01"但是没有学过编号为"02"的课程的同学的信息

select st.\* from student st

inner join score sc on sc.s\_id = st.s\_id

inner join course c on c.c\_id=sc.c\_id and c.c\_id="**01**"

where st.s\_id not in (

select st2.s\_id from student st2

inner join score sc2 on sc2.s\_id = st2.s\_id

inner join course c2 on c2.c\_id=sc2.c\_id and c2.c\_id="**02**"

)

-- 11、查询没有学全所有课程的同学的信息

-- 太复杂,下次换一种思路,看有没有简单点方法

-- 此处思路为查学全所有课程的学生id,再内联取反面

select \* from student where s\_id not in (

select st.s\_id from student st

inner join score sc on sc.s\_id = st.s\_id and sc.c\_id="**01**"

where st.s\_id in (

select st2.s\_id from student st2

inner join score sc2 on sc2.s\_id = st2.s\_id and sc2.c\_id="**02**"

) and st.s\_id in (

select st2.s\_id from student st2

inner join score sc2 on sc2.s\_id = st2.s\_id and sc2.c\_id="**03**"

))

-- 12、查询至少有一门课与学号为"01"的同学所学相同的同学的信息

select distinct st.\* from student st

left join score sc on sc.s\_id=st.s\_id

where sc.c\_id in (

select sc2.c\_id from student st2

left join score sc2 on sc2.s\_id=st2.s\_id

where st2.s\_id ='01'

)

-- 13、查询和"01"号的同学学习的课程完全相同的其他同学的信息

select st.\* from student st

left join score sc on sc.s\_id=st.s\_id

group by st.s\_id

having group\_concat(sc.c\_id) =

(

select **group\_concat**(sc2.c\_id) from student st2

left join score sc2 on sc2.s\_id=st2.s\_id

where st2.s\_id ='01'

)

【自己写】

SELECT st.\*

FROM student st

INNER JOIN score sc ON st.s\_id=sc.s\_id GROUP BY st.s\_id **HAVING GROUP\_CONCAT**(sc.c\_id) = (

SELECT GROUP\_CONCAT(sc.c\_id)

FROM student st

INNER JOIN score sc ON st.s\_id=sc.s\_id AND st.s\_id='01'

)

**注意HAVING GROUP\_CONCAT(sc.c\_id) 要用having关键词**

-- 14、查询没学过"张三"老师讲授的任一门课程的学生姓名

select st.s\_name from student st

where st.s\_id not in (

select sc.s\_id from score sc

inner join course c on c.c\_id=sc.c\_id

inner join teacher t on t.t\_id=c.t\_id and t.t\_name="张三"

)

【自己写的】

SELECT st1.s\_name

FROM student st1 WHERE st1.s\_id NOT IN (

SELECT st.s\_id FROM student st

INNER JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON sc.c\_id=c.c\_id

INNER JOIN teacher te ON c.t\_id=te.t\_id AND te.t\_name="张三"

)

-- 15、查询两门及其以上不及格课程的同学的学号，姓名及其平均成绩

【答案】

select st.s\_id,st.s\_name,avg(sc.s\_score) from student st

left join score sc on sc.s\_id=st.s\_id

where sc.s\_id in (

select sc.s\_id from score sc

where sc.s\_score<**60** or sc.s\_score is NULL

group by sc.s\_id having COUNT(sc.s\_id)>=**2**

)

group by st.s\_id

【自己写的】

SELECT st.s\_id,st.s\_name,AVG(sc.s\_score),COUNT(1)

FROM student st

INNER JOIN score sc ON sc.s\_id=st.s\_id AND sc.s\_score<60

GROUP BY st.s\_id

HAVING COUNT(1) >=2

-- 16、检索"01"课程分数小于60，按分数降序排列的学生信息

select st.\*,sc.s\_score from student st

inner join score sc on sc.s\_id=st.s\_id and sc.c\_id="**01**" and sc.s\_score<**60**

order by sc.s\_score desc

SELECT st.\*,sc.s\_score

FROM student st

INNER JOIN score sc ON sc.s\_id=st.s\_id

AND sc.c\_id='01' AND sc.s\_score<60

ORDER BY sc.s\_score DESC

**这里用inner join是对的 : )**

**Inner join 取交集 ，需要选01课程的所有学生编号就可以 √**

-- 17、按平均成绩从高到低显示所有学生的所有课程的成绩以及平均成绩

-- 可加round,case when then else end 使显示更完美

【错误】SELECT st.\*,sc.s\_score,@avgscore:=((SELECT scc.s\_id, AVG(scc.s\_score)

FROM score scc

GROUP BY scc.s\_id) WHERE scc.s\_id=sc.s\_id)

FROM student st

LEFT JOIN score sc ON st.s\_id=sc.s\_id

【正确】

SELECT st.s\_name,st.s\_id,AVG(sc4.s\_score) "平均分",sc.s\_score "语文",sc2.s\_score "数学",sc3.s\_score "英语"

FROM student st

LEFT JOIN score sc ON st.s\_id=sc.s\_id AND sc.c\_id='01'

LEFT JOIN score sc2 ON st.s\_id=sc2.s\_id AND sc2.c\_id='02'

LEFT JOIN score sc3 ON st.s\_id=sc3.s\_id AND sc3.c\_id='03'

LEFT JOIN score sc4 ON st.s\_id=sc4.s\_id

GROUP BY st.s\_id

ORDER BY SUM(sc4.s\_score) DESC

注意：先想好字段名称，可关联**多张表**，leftjoin 多次

-- 18.查询各科成绩最高分、最低分和平均分：以如下形式显示：课程ID，课程name，最高分，最低分，平均分，及格率，中等率，优良率，优秀率

-- 及格为>=60，中等为：70-80，优良为：80-90，优秀为：>=90

SELECT c.c\_id AS "课程编号" , c.c\_name AS "课程name" , AVG(sc1.s\_score) AS "平均分",MAX(sc1.s\_score) AS "最高分", MIN(sc1.s\_score) AS "最低分",(

(SELECT COUNT(1)

FROM score sc

WHERE sc.s\_score>=60 AND sc.c\_id=c.c\_id)

/

(

SELECT COUNT(1)

FROM score sc

WHERE sc.c\_id=c.c\_id)

) AS "及格率"

FROM course c

LEFT JOIN score sc1 ON c.c\_id=sc1.c\_id

GROUP BY c.c\_id

**注意，groupby之后的参数是可以使用的！ 算**及格率= 及格人数/所有考试的人

单独算各科及格人数 SELECT COUNT(1) FROM score sc WHERE sc.s\_score>=60

GROUP BY sc.c\_id

**但此处，已经有groupby，所以可省略，单独计算列值的时候运用此参数，**

SELECT COUNT(1) FROM score sc WHERE sc.s\_score>=60 AND **sc.c\_id=c.c\_id**

**Group By 参数已经单独算了 c.c\_id**

-- 19、按各科成绩进行排序，并显示排名(实现不完全)

-- mysql没有rank函数

-- 加@score是为了防止用union all 后打乱了顺序

-- 20、查询学生的总成绩并进行排名

SELECT

st.s\_id,

st.s\_name,

CASE

WHEN SUM( sc.s\_score ) IS NULL

OR SUM( sc.s\_score ) = "" THEN

0 ELSE SUM( sc.s\_score )

END allscore

FROM

student st

LEFT JOIN score sc ON st.s\_id = sc.s\_id

GROUP BY

st.s\_id

ORDER BY

allscore DESC

-- 21、查询不同老师所教不同课程平均分从高到低显示

SELECT

te.t\_id,

te.t\_name,

c.c\_name,

AVG( sc.s\_score ) avgscore

FROM

teacher te

LEFT JOIN course c ON c.t\_id = te.t\_id

LEFT JOIN score sc ON sc.c\_id = c.c\_id

GROUP BY

te.t\_id

ORDER BY

avgscore DESC

-- 22、查询所有课程的成绩第2名到第3名的学生信息及该课程成绩

SELECT a.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='01'

ORDER BY sc.s\_score

DESC

LIMIT 1,2

) a

UNION ALL

SELECT b.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='02'

ORDER BY sc.s\_score

DESC

LIMIT 1,2

)b

UNION ALL

SELECT c.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='03'

ORDER BY sc.s\_score

DESC

LIMIT 1,2

) c

**重点：union all 是对数据进行并集操作，包括重复行，但是不排序**

**Union 是对数据进行并集操作，不包括重复行，默认排序**

**Union合并两个或多个select语句的结果集。**

3个课程分开来搜索，然后并集，如果用group by c.c\_id只会有一条数据

-- 23、统计各科成绩各分数段人数：课程编号,课程名称,[100-85],[85-70],[70-60],[0-60]及所占百分比

SELECT c.c\_id,c.c\_name,

(

(SELECT COUNT(1) FROM score sc WHERE sc.c\_id=c.c\_id AND sc.s\_score<=100 AND sc.s\_score>85)/

(SELECT COUNT(1) FROM score sc WHERE sc.c\_id=c.c\_id)

)\*100

AS "100-85"

FROM course c

**重点：这里没有用left join 思考一下**

select c.c\_id,c.c\_name

,((select count(**1**) from score sc where sc.c\_id=c.c\_id and sc.s\_score<=**100** and sc.s\_score>**80**)/(select count(**1**) from score sc where sc.c\_id=c.c\_id )) "**100**-**85**"

,((select count(**1**) from score sc where sc.c\_id=c.c\_id and sc.s\_score<=**85** and sc.s\_score>**70**)/(select count(**1**) from score sc where sc.c\_id=c.c\_id )) "**85**-**70**"

,((select count(**1**) from score sc where sc.c\_id=c.c\_id and sc.s\_score<=**70** and sc.s\_score>**60**)/(select count(**1**) from score sc where sc.c\_id=c.c\_id )) "**70**-**60**"

,((select count(**1**) from score sc where sc.c\_id=c.c\_id and sc.s\_score<=**60** and sc.s\_score>=**0**)/(select count(**1**) from score sc where sc.c\_id=c.c\_id )) "**60**-**0**"

from course c order by c.c\_id

-- 24、查询学生平均成绩及其名次

SELECT

a.\*,

@curRank := @curRank + 1 AS rank

FROM

(

SELECT

st.s\_id,

st.s\_name,

AVG( sc.s\_score )

FROM

student st

LEFT JOIN score sc ON st.s\_id = sc.s\_id

GROUP BY

st.s\_id

ORDER BY

AVG( sc.s\_score ) DESC

) a,

( SELECT @curRank := 0 ) q

**重点：**排名问题怎么解决，mysql里没有rank函数

基本知识：

sql语句中，使用@来定义一个变量。如：@abc

sql语句中，使用:=来给变量赋值,：@abc:=123,则变量abc的值为123

sql语句中，if(A,B,C)表示，如果A条件成立，那么执行B，否则执行C，如:

@abc := if(2>1,100,200)的结果是，abc的值为100。

◎case...when...then语句

case...when...then语句有两种情况：

case情况一（CASE 后面不带表达式）:

CASE WHEN expression THEN 操作1

           WHEN expression THEN 操作2

            .......

           ELSE 操作n

END

注：自上而下，凡是走了其中一个when或者是走了else了，其他的都不再走了。

        case情况二（CASE 后面带表达式，此时WHEN 后面的则是该表达式可能的值）:

CASE expression

WHEN  expression的值1 THEN  操作1

WHEN  expression的值2 THEN  操作2

     .......

    ELSE 操作n

END

答案：

set @i=**0**;

select a.\*,@i:=@i+**1** from (

select st.s\_id,st.s\_name,round((case when avg(sc.s\_score) is null then **0** else avg(sc.s\_score) end),**2**) "平均分" from student st

left join score sc on sc.s\_id=st.s\_id

group by st.s\_id order by sc.s\_score desc) a

**注意重点：**

先查找出来按照平均分的排名，再给一个新的字段rank，初始化rank值为0，再分别进行rank操作

\*\*from tablename aliasname\*\*

当age相同时，排名相同，当下一个age不同时，排名跳级+n

SELECT sc.s\_score, sc.s\_id,

@curRank:= IF(@preRank=sc.s\_score,@curRank,@incRank) as rank,

@incRank:=@incRank+1,

@preRank:=sc.s\_score

FROM score sc,

(SELECT @curRank :=0, @preRank:=NULL,@incRank:=1) q

ORDER BY sc.s\_score

-- 25、查询各科成绩前三名的记录

SELECT a.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='01'

ORDER BY sc.s\_score

DESC

LIMIT 0,3

) a

UNION ALL

SELECT b.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='02'

ORDER BY sc.s\_score

DESC

LIMIT 0,3

)b

UNION ALL

SELECT c.\* FROM (

SELECT st.\*,c.c\_id,c.c\_name,sc.s\_score

FROM student st

LEFT JOIN score sc ON sc.s\_id=st.s\_id

INNER JOIN course c ON c.c\_id=sc.c\_id AND c.c\_id='03'

ORDER BY sc.s\_score

DESC

LIMIT 0,3

) c

和第22题相同

-- 26、查询每门课程被选修的学生数

SELECT

c.c\_id,

c\_name,

COUNT( st.s\_id )

FROM

course c

LEFT JOIN score sc ON c.c\_id = sc.c\_id

LEFT JOIN student st ON sc.s\_id = st.s\_id

GROUP BY

c.c\_id

给出的答案：

SELECT

c.c\_id,

c.c\_name,

count( 1 )

FROM

course c

LEFT JOIN score sc ON sc.c\_id = c.c\_id

**INNER JOIN** student st ON st.s\_id = c.c\_id

GROUP BY

st.s\_id

-- 27、查询出只有两门课程的全部学生的学号和姓名

SELECT

st.s\_id,

st.s\_name,

COUNT( 1 )

FROM

student st

LEFT JOIN score sc ON st.s\_id = sc.s\_id

GROUP BY

st.s\_id

HAVING

COUNT( 1 ) = 2

-- 28、查询男生、女生人数

SELECT

st.s\_sex,

**COUNT( 1 )**

FROM

student st

GROUP BY

st.s\_sex

重点：**COUNT( 1 )**

Count(1) 返回查询到的行中第一列 not null 的个数

Count(\*) 返回查询到的结果集中的行个数，不论是否含有null

Count（col）返回查询到的行中col列中not null的个数

-- 29、查询名字中含有"风"字的学生信息

SELECT

st.\*

FROM

student st

WHERE

st.s\_name LIKE '%风%'

-- 30、查询同名同性学生名单，并统计同名人数

SELECT

st.\*,

COUNT( 1 )

FROM

student st

GROUP BY

st.s\_name,

st.s\_sex

HAVING

**COUNT( 1 ) >1**

重点：**COUNT( 1 )**

每个人都会有单独的记录，所以找到同名同性，只要筛选条件大于1就行

