《数据库系统》2023-2024第二学期SQL考试

**样题（供练习）**

学号： 姓名： 专业：

机房号： 座位号： 分数：

注意事项：

1. 答案以Word文件的方式提交，文件名为“学号\_姓名.docx”，文件放在D盘根目录下，未按照规定方式命名会影响考试成绩。
2. 每个题目的SQL语句都必须在查询分析器中调试，运行无误后提交查询的SQL语句（文字）和查询结果截图（需将SQL语句和运行结果保存在同一张截图中提交）。
3. 查询结果集中的列名必须采用查询需求中给出的列名。

**数据库模式**

**学院**（学院编号\*，学院名，负责人）

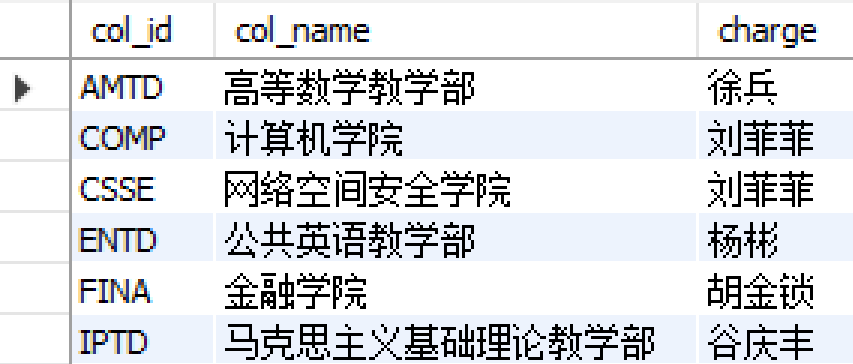
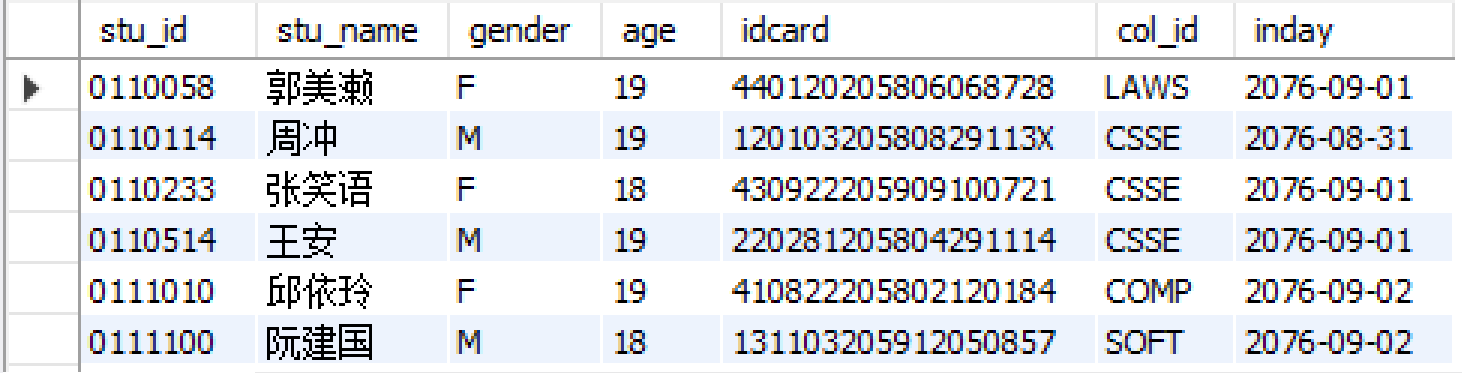
**学生**（学号\*，姓名，性别，年龄，身份证，学院编号+，入学日期）

**课程**（课号\*，课程名，学院编号+，学分，星期，开始时间，结束时间）

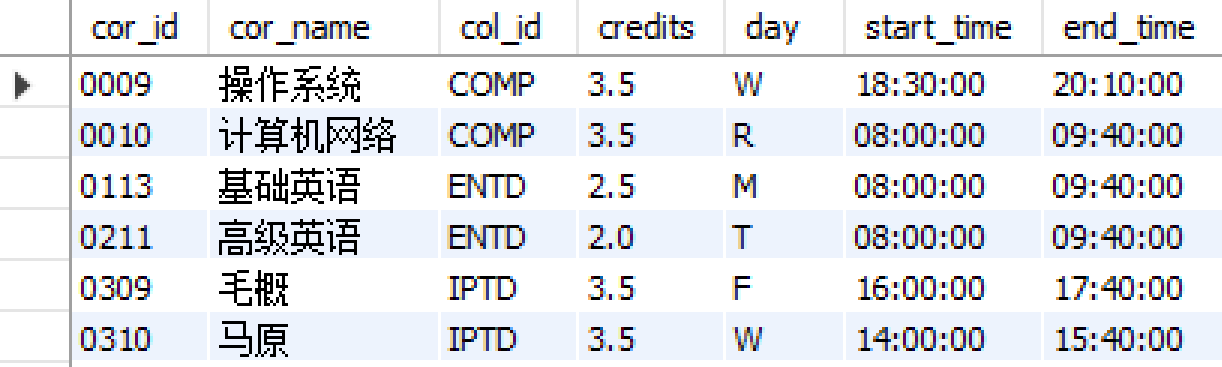
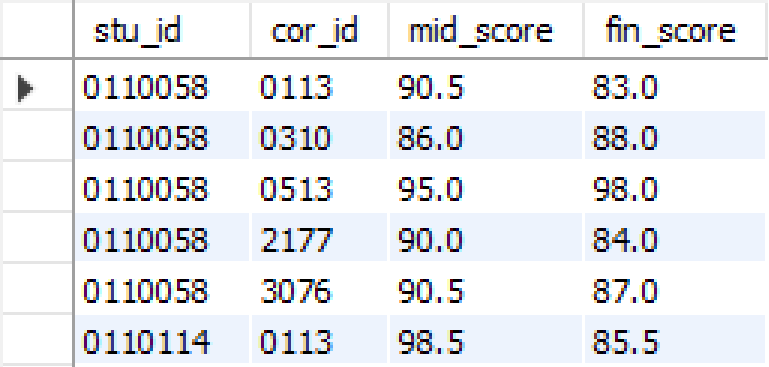
**选课**（学号\*+，课号\*+，平时成绩，期末成绩）

注：\*为主键，+为外键

**部分示例数据如下（含表名、属性名）：**

**学院（college） 学生（student）**

 ****

**课程（course）** **选课（takes）**

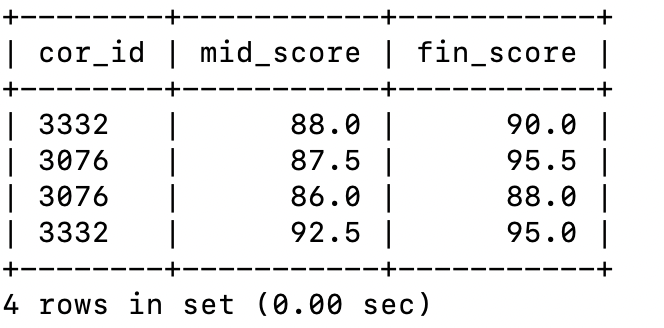
**注：以上部分示例数据仅供参考，具体的SQL语句不应该和具体的数据有关，请在答题过程中合理考虑*空值、重复值*等。题目中给出了结果模式的属性**

**1. 查询 ‘选课’ 表中所有不同的课号、平时成绩、期末成绩；要求只包括期末成绩高于平时成绩且课号大于3000的记录。（cor\_id, mid\_score, fin\_score）（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT DISTINCT cor\_id, mid\_score, fin\_score  FROM takes  WHERE fin\_score > mid\_score and cor\_id>3000; |

**预期结果：**

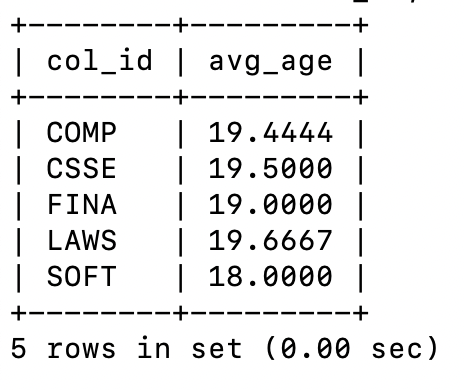


**2. 查询每个学院的学生平均年龄 (注：过滤学院编号为空的数据)。(col\_id, avg\_age)（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT col\_id, AVG(age) AS avg\_age  FROM student where col\_id is not null  GROUP BY col\_id; |

**预期结果：**

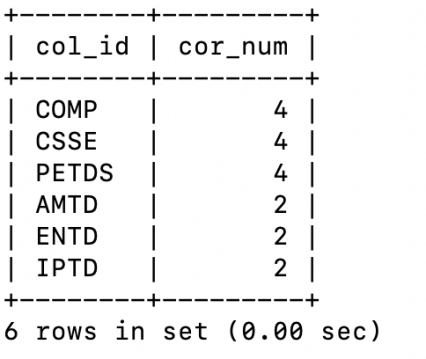


**3. 查询开设课程数量大于1门的学院编号、开设课程数量，并按课程数量从多到少排序。(col\_id, cor\_num)（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT col\_id, COUNT(\*) AS cor\_num  FROM course  GROUP BY col\_id  HAVING cor\_num>1  ORDER BY cor\_num DESC; |

**预期结果：**

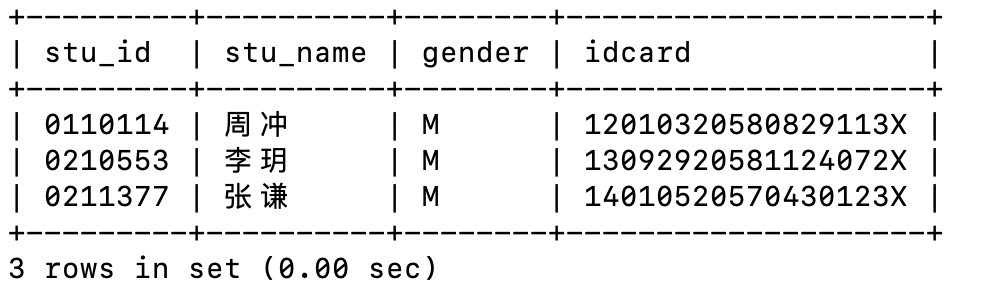


**4. 查询身份证号码中包含‘X’的学生的学号、姓名、性别、身份证号。(stu\_id, stu\_name, gender, idcard)（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT stu\_id, stu\_name, gender,idcard  FROM student  WHERE idcard LIKE '%X%'; |

**预期结果：**



**5. 查询每门课程的平均成绩和选课人数，其中课程最终成绩=0.3\*平时成绩+0.7\*期末成绩，按选课人数降序。（cor\_id，cor\_name，avg\_score，student\_count）（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT  c.cor\_id,  c.cor\_name,  AVG(0.3\*t.mid\_score + 0.7\*t.fin\_score) AS avg\_score,  COUNT(t.stu\_id) AS student\_count  FROM  course c  LEFT JOIN  takes t ON c.cor\_id = t.cor\_id  GROUP BY  c.cor\_id, c.cor\_name  ORDER BY student\_count DESC; |

**预期结果：**



**6. 查询属于同一学院开设且在同一天上课的课程信息对儿。（cor\_id, cor\_name, dup\_id, dup\_name）（10分）**

**参考sql语句：**

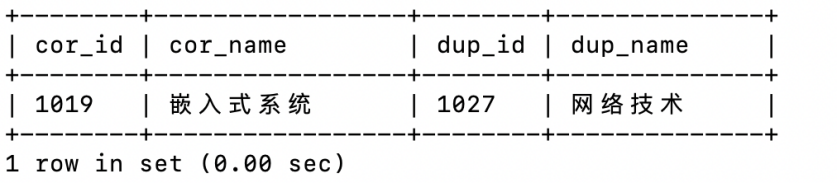
select c1.cor\_id,c1.cor\_name,c2.cor\_id,c2.cor\_name

from course c1, course c2

where c1.col\_id=c2.col\_id and c1.day=c2.day and c1.cor\_id<c2.cor\_id

|  |
| --- |
| Select c1.cor\_id, c1.cor\_name, c2.cor\_id as dup\_id, c2.cor\_name as dup\_name  From course c1  Join course c2 on c1.cor\_id <>c2.cor\_id  Where c1.day=c2.day and c1.col\_id=c2.col\_id and c1.cor\_id <c2.cor\_id; |

**预期结果：**

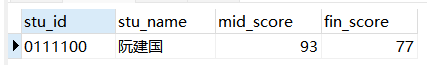


**7. 找出“高等数学A”课程成绩排名第十的学生信息，其中课程最终成绩=0.3\*平时成绩+0.7\*期末成绩。（stu\_id，stu\_name，mid\_score，fin\_score）（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT t1.stu\_id, s.stu\_name, t1.mid\_score, t1.fin\_score  FROM takes t1  INNER JOIN student s ON t1.stu\_id = s.stu\_id  INNER JOIN course c ON t1.cor\_id = c.cor\_id  WHERE c.cor\_name = '高等数学A'  AND (  SELECT COUNT(\*)  FROM takes t2  INNER JOIN course c2 ON t2.cor\_id = c2.cor\_id  WHERE c2.cor\_name = '高等数学A' AND ((t2.mid\_score \* 0.3) + (t2.fin\_score \* 0.7)) > ((t1.mid\_score \* 0.3) + (t1.fin\_score \* 0.7))  ) = 9; |

**预期结果：**

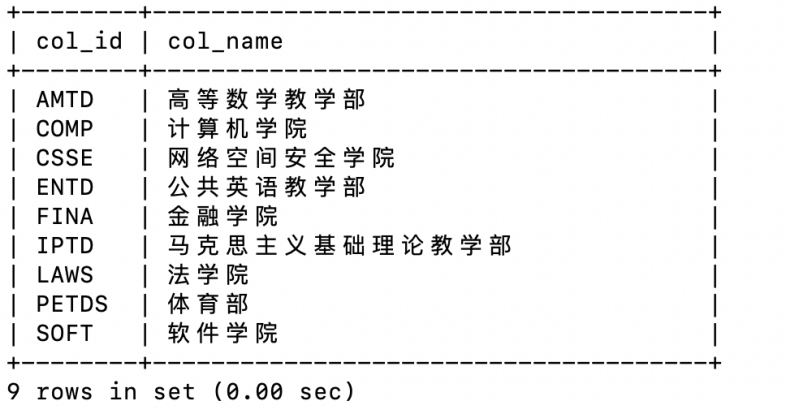


**8. 查询所有没在周末（周末对应星期属性值为S）开设课程的学院的学院编号、学院名。（col\_id, col\_name）(10分)**

**参考sql语句：**

|  |
| --- |
| SELECT c.col\_id,col\_name  FROM college c  WHERE NOT EXISTS (  SELECT \*  FROM course cou  WHERE cou.col\_id = c.col\_id and day= 'S');  或者  Select col\_id,col\_name  From college  Where not exists (  Select 1  From course  Where college.col\_id=course.col\_id  And day= 'S'); |

**预期结果：**

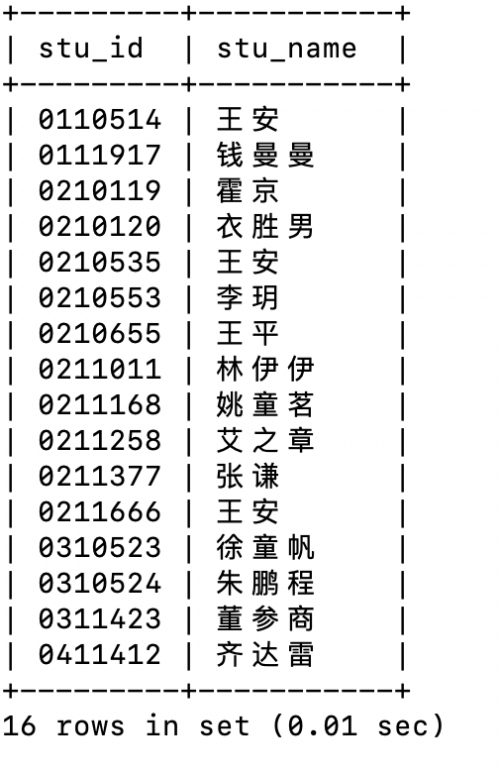
****

**9. 查询至少选修了6门课程的学生的学号、姓名。（stu\_id, stu\_name）(10分)**

**参考sql语句：**

|  |
| --- |
| SELECT  s.stu\_id,  s.stu\_name  FROM  student s  JOIN (  SELECT  sc.stu\_id  FROM  takes sc  GROUP BY  sc.stu\_id  HAVING  COUNT(sc.cor\_id) >= 6  ) AS subquery ON s.stu\_id = subquery.stu\_id; |

**预期结果：**



**10. 查询至少选修了一门课程的学生的学号、姓名、学院编号以及他们的加权平均分，并且按照加权平均分从高到低排序。其中，加权平均分计算方式：[(课程A期末成绩\*课程A学分)+ (课程B期末成绩\*课程B学分)] / 总课程学分。（stu\_id, stu\_name, col\_id, w\_score）（10分）**

**参考sql语句：**

|  |
| --- |
| SELECT  s.stu\_id,  s.stu\_name,  s.col\_id,  SUM(sc.fin\_score \* c.credits) / SUM(c.credits) AS w\_score  FROM  student s  JOIN  takes sc ON s.stu\_id = sc.stu\_id  JOIN  course c ON sc.cor\_id = c.cor\_id  GROUP BY  s.stu\_id, s.stu\_name, s.col\_id  HAVING  SUM(c.credits) > 0  ORDER BY  w\_score DESC; |

**预期结果：**

