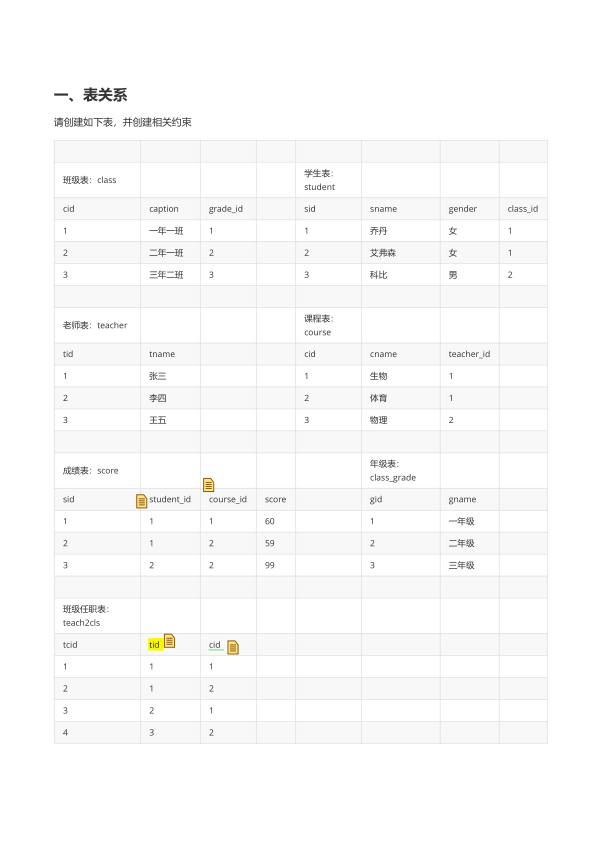
[**\_王晓东**](https://www.cnblogs.com/Mr-wangxd/)

[Django的orm练习---多表查询](https://www.cnblogs.com/Mr-wangxd/p/10015657.html)

[Django的orm练习---多表查询](https://www.cnblogs.com/zmc940317/p/9873965.html)

 表关系如下



表结构 :

[复制代码](javascript:void(0);)

复制代码

from django.db import models

# Create your models here.

# 多对多----->>>老师和班级

# 一对多----->>>学生和班级 : 一个班级可以有多个学生----学生设置主键

# 老师和课程 : 一个老师可以教多门课程----课程设置主键

#

# 一对一---->>>>班级和年级 : 一个班级对应一个年级

#

# 成绩表----学生 : 一对多. 一个学生多个成绩 /

# 成绩表----课程 : 一对多

# 学生

class Student(models.Model):

sid=models.AutoField(primary\_key=True)

sname=models.CharField(max\_length=32)

gender=models.CharField(max\_length=32)

class\_id=models.ForeignKey(to="Class",on\_delete=models.CASCADE)

# 班级

class Class(models.Model):

cid=models.AutoField(primary\_key=True)

caption=models.CharField(max\_length=32)

grade=models.ForeignKey(to="Class\_grade",on\_delete=models.CASCADE)

teachers=models.ManyToManyField(to="Teacher")

# 年级

class Class\_grade(models.Model):

gid=models.AutoField(primary\_key=True)

gname=models.CharField(max\_length=32)

# 课程

class Course(models.Model):

cid=models.AutoField(primary\_key=True)

cname=models.CharField(max\_length=32)

teacher=models.ForeignKey(to="Teacher",on\_delete=models.CASCADE)

# 老师

class Teacher(models.Model):

tid=models.AutoField(primary\_key=True)

tname=models.CharField(max\_length=32)

#成绩

class Score(models.Model):

sid=models.AutoField(primary\_key=True)

student=models.ForeignKey(to="Student",on\_delete=models.CASCADE)

course=models.ForeignKey(to="Course",on\_delete=models.CASCADE)

score=models.IntegerField()

复制代码

[复制代码](javascript:void(0);)

习题 :

[复制代码](javascript:void(0);)

复制代码

# 1、自行创建测试数据；

# 2、查询学生总人数；

# 3、查询“生物”课程和“物理”课程成绩都及格的学生id和姓名；

# 4、查询每个年级的班级数，取出班级数最多的前三个年级；

# 5、查询平均成绩最高的学生的id和姓名以及平均成绩；

# 6、查询每个年级的学生人数；

# 7、查询每位学生的学号，姓名, 平均成绩；

# 8、查询学生编号为“2”的学生的姓名、该学生成绩最高的课程名及分数；

# 9、查询姓“李”的老师的个数和所带班级数；

# 10、查询班级数小于5的年级id和年级名；

# 11、查询教过课程超过2门的老师的id和姓名；

# 12、查询学过编号“1”课程和编号“2”课程的同学的学号、姓名；

# 13、查询所带班级数最多的老师id和姓名；

# 14、查询有课程成绩小于60分的同学的学号、姓名；

# 15、查询男生、女生的人数，按倒序排列；

# 16、 查询各个课程及相应的选修人数；

# 17、 查询同时选修了物理课和生物课的学生id和姓名；

# 18、 检索“3”课程分数小于60，按分数降序排列的同学学号；

# 19、 查询每门课程的平均成绩，结果按平均成绩升序排列，平均成绩相同时，按课程号降序排列；

# 20、 查询各科成绩最高和最低的分：以如下形式显示：课程ID，最高分，最低分；

复制代码

[复制代码](javascript:void(0);)

答案 :

[复制代码](javascript:void(0);)

复制代码

from django.shortcuts import render,HttpResponse,redirect

from app01.models import Class,Course,Teacher,Student,Score,Class\_grade

from django.db.models import F, Q

from django.db.models import Avg,Max,Sum,Min,Count

# Create your views here.

# values.annotate() : 按字段分组

# annotate() : 按id,name...分组,属于一列是一组

def query(request):

# 2 . 查询学生总人数

ret=Student.objects.count()

print("------>",ret) # {'c': 3}

# 3 . 查询“生物”课程和“物理”课程成绩都及格的学生id和姓名；

ret = Student.objects.filter(score\_\_course\_\_cid=3,score\_\_score\_\_gt=59).filter(score\_\_course\_\_cid=1,score\_\_score\_\_gt=59).values("sname", "sid")

print("----------------", ret)

# 先去筛选选课在生物和物理之间的,并且创建大于60的学生id,在分组查看选课数,然后筛选选课数等于2的

ret=Score.objects.filter(course\_\_cname\_\_in=["生物","物理"],score\_\_gt=60).values("student\_\_tid").annotate(c=Count("course")).filter(c=2)

# 4.查询每个年级的班级数，取出班级数最多的前三个年级；

ret=Score.objects.values("student").annotate(avg\_score=Avg("score")).order\_by("-avg\_score").values("student\_\_sname","student\_\_pk","avg\_score")[0]

print(ret)

ret=Class.objects.values("grade\_\_gname").annotate(c=Count("caption")).order\_by("-c")[:3]

print("------>",ret)

# 5.查询平均成绩最高的学生的id和姓名以及平均成绩；

ret=Grade.objects.annotate(c=Count("klass\_\_student\_\_pk")).values("gname","c")

print(ret)

ret=Student.objects.values("sid","sname").annotate(scoreAvg = Avg("score\_\_score")).order\_by("-scoreAvg")[:1]

print("------>", ret)

# 6.查询每个年级的学生人数；

ret=Student.objects.values("sid","sname").annotate(avg\_score=Avg("score\_\_score"))

print(ret)

ret=Student.objects.annotate(avg\_score=Avg("score\_\_score")).values("sid","sname","avg\_score")

ret=Class.objects.values("grade\_\_gname").annotate(c=Count("student"))

print("-------->",ret)

# 7 . 查询每位学生的学号，姓名,平均成绩；

ret=Student.objects.values("sid","sname").annotate(scoreAvg=Avg("score\_\_score"))

print("-------->", ret)

ret=Score.objects.filter(student\_\_pk=2).order\_by("-score").values("student\_\_sname","course\_\_cname","score")[0]

print(ret)

# 8、查询学生编号为“2”的学生的姓名、该学生成绩最高的课程名及分数；

ret=Student.objects.filter(sid="2").annotate(scoreMax=Max("score\_\_score")).order\_by('-scoreMax')[0:1].values("sname","score\_\_course\_\_cname","scoreMax")

ret2=Student.objects.filter(sid="2").values("score\_\_course").order\_by("-score\_\_score").values("sname","score\_\_course\_\_cname","score\_\_score")[:1]

print("-------->", ret)

print("-------->", ret2)

# 9、查询每一个姓“李”的老师所带班级数；；

ret=Teacher.objects.filter(tname\_\_istartswith="李").annotate(c=Count("classes")).values("tname","c")

print(ret)

# 10 . 查询班级数小于5的年级id和年级名；

ret=Class\_grade.objects.annotate(c=Count("class")).filter(c\_\_lt=2).values("gid","gname")

print("--------", ret)

ret=Grade.objects.annotate(c=Count("klass")).filter(c\_\_lt=5).values("pk","gname")

# 11 . 查询教过课程超过2门的老师的id和姓名；

ret=Teacher.objects.annotate(c=Count("course")).filter(c\_\_gt=2).values\_list("tid","tname")

print("--------", ret)

# 12 . 查询学过编号“1”课程和编号“2”课程的同学的学号、姓名； ????????

ret=Student.objects.filter(score\_\_course\_\_cid=1).filter(score\_\_course\_\_cid=2).values("sid","sname")

print("-------",ret)

# 13 . 查询所带班级数最多的老师id和姓名；

ret=Teacher.objects.annotate(c=Count("class\_\_cid")).order\_by("-c").values("tid","tname","c")[0]

print(">>>>>>>>>>",ret)

# 14 . 查询有课程成绩小于60分的同学的学号、姓名；

ret=Student.objects.filter(score\_\_score\_\_lt=60).values("sid","sname","score\_\_sid").distinct() #去重

print("-------",ret)

ret=Score.objects.filter(score\_\_lt=60).values("student\_\_sname","student\_\_pk").distinct()

print(ret)

# 15 . 查询男生、女生的人数，按倒序排列；

ret=Student.objects.values("gender").annotate(c=Count("gender")).order\_by("-c")

print("-------",ret)

# 16 . 查询各个课程及相应的选修人数；

ret=Course.objects.annotate(c=Count("score\_\_student")).values("cname","c")

print("-------", ret)

ret=Score.objects.values("course").annotate(c=Count(1)).values("course\_\_cname","c")

print(ret)

# 17 . 查询同时选修了物理课和生物课的学生id和姓名；

ret=Student.objects.filter(score\_\_course\_\_cname="物理").filter(score\_\_course\_\_cname="生物").values("sid","sname")

print("------->",ret)

# 18 . 检索“3”课程分数小于60，按分数降序排列的同学学号；

ret=Student.objects.filter(Q(score\_\_course\_\_cid=3),Q(score\_\_score\_\_lt=60)).order\_by("-score\_\_score").values("sid")

ret=Score.objects.filter(course\_\_cid=3).filter(score\_\_lt=60).order\_by("-score").values("student\_\_sid")

print("------->", ret)

ret=Score.objects.filter(course\_id=3,score\_\_lt=60).order\_by("-score").values("student\_id")

# 19 . 查询每门课程的平均成绩，结果按平均成绩升序排列，平均成绩相同时，按课程号降序排列；

ret=Score.objects.values("course\_id").annotate(scoreAvg=Avg("score")).order\_by("scoreAvg","-course\_id") #可以不用跨表 course\_id

print("------->", ret)

# 20 . 查询各科成绩最高和最低的分：以如下形式显示：课程ID，最高分，最低分；

ret=Score.objects.values("course\_\_cid").annotate(scoreMax=Max("score"),scoreMin=Min("score"))

print(">>>>>>>>>>>",ret)

ret=Course.objects.annotate(max\_score=Max("score\_\_score"),min\_score=Min("score\_\_score")).values("pk","max\_score","min\_score")

return HttpResponse("ok")