ORM练习题

一 .表结构

```
#出版社
class Publisher(models.Model):
    name = models.CharField(max_length=32)
    city = models.CharField(max_length=32)

    def __str__(self):
        return  "<Publisher object: {} {}>".format(self.id, self.name)
#书籍
class Book(models.Model):
    title = models.CharField(max_length=32)
    publish_date = models.DateField(auto_now_add=True)
    price = models.DecimalField(max_digits=5,decimal_places=2)
    memo = models.TextField(null=True)
    #创建外键,关联Publisher
    Publisher = models.ForeignKey(to='Publisher')

    def __str__(self):
        return "<Book object: {} {}>".format(self.id, self.title

#作者
class Author(models.Model):
    name = models.CharField(max_length =32)
    age = models.IntegerField()
    phone = models.CharField(max_length=11)
    #创建多对多关联
    books = models.ManyToManyField(to='Book')

    def __str__(self):
        return "<Author object: {} {}>".format(self.id, self.name)
```

题目

1.查找所有书名里包含金老板的书

```
ret= models.Book.objects.filter(title__contains = '金老板')
```

2.查找出版日期是2018年的书

```
ret = models.Book.objects.filter(publisher_data__year =2018)
```

3.查找出版日期是2017年的书名

```
ret = models.Book.objects.filter(publisher_data__year =2018).values('title')
```

4.查找价格大于10元的书

```
ret = models.Book.objects.filter(price__gt=10)
```

5.查找价格大于10元的书名和价格

```
ret = models.Book.objects.filter(price__gt=10).values('title','price')
```

6.查找memo字段是空的书

```
ret = models.Book.objects.filter(memo__isnull=True)
```

7.查找在北京的出版社

```
ret = models.Publisher.objects.filter(city='北京')
```

8.查找名字以沙河开头的出版社

```
ret = models.Publisher.objects.filter(name__startswith='沙河')
```

9.查找"沙河出版社"出版的所有书籍

```
#基于对象的查询
    publisher_obj = models.Publisher.objects.filter(name="沙河出版社")
    ret = publisher_obj.book_set.all()
#基于字段的查询
    ret = models.Book.objects.filter(publiser__name = "沙河出版社")
```

10.查找每个出版社出版价格最高的书籍价格

```
ret = models.Publisher.all().annotate(max=Max('book__price')).values('name','max')
```

11.查找每个出版社的名字以及出的书籍数量

```
ret = models.Publisher.all().annotate(count=Count('book__title')).vales('name','count')
```

12.查找作者名字里面带"小"字的作者

```
ret = models.Author.objects.filter(name__contains='小')
```

13.查找年龄大于30岁的作者

```
ret = models.Author.objects.filter(age__gt=30)
```

14.查找手机号是155开头的作者

```
ret = models.Author.objects.filter(phone__startswith=155)
```

15.查找手机号是155开头的作者的姓名和年龄

```
ret = models.Author.objects.filter(phone__startswith=155).values('name','age')
```

16.查找每个作者写的价格最高的书籍价格

```
ret = models.Book.objects.values('author').annotate(max=Max('price')).values('author','max')
```

```
ret= models.Author.objects.all().annotate(max=Max('books__price')).values('name','max')
```

17.查找每个作者的姓名以及出的书籍数量

```
ret=
models.Author.objects.all().annotate(count=Count('books__title')).values('name','count')
```

18.查找书名是“跟金老板学开车”的书的出版社

```
ret = models.Publisher.objects.filter(book__title='跟金老板学开车')
```

19.查找书名是“跟金老板学开车”的书的出版社所在的城市

```
ret = models.Publisher.objects.filter(book__title='跟金老板学开车').values('city')
```

20.查找书名是“跟金老板学开车”的书的出版社的名称

```
ret = models.Publisher.objects.filter(book__title='跟金老板学开车').values('name')
```

21.查找书名是“跟金老板学开车”的书的出版社出版的其他书籍的名字和价格

```
pub_obj = models.Publisher.objects.get(book__title='跟金老板学开车')
ret= pub_obj.book_set.all().exclude(title='跟金老板学开车').values('title','price')
```

22.查找书名是“跟金老板学开车”的书的所有作者

```
ret = models.Author.objects.filter(books__title='跟金老板学开车')
```

23.查找书名是“跟金老板学开车”的书的作者的年龄

```
ret = models.Author.objects.filter(books__title='跟金老板学开车').values('name','age')
```

24.查找书名是“跟金老板学开车”的书的作者的手机号码

```
ret = models.Author.objects.filter(books__title='跟金老板学开车').values('name','phone'
```

25.查找书名是“跟金老板学开车”的书的作者们的姓名以及出版的所有书籍名称和价钱

```
ret =
models.Author.objects.values('name','books__title','books__price').filter(books__title='跟金老板学开车')
```

二.表结构

```
class Student(models.Model):
    s_name = models.CharField(max_length=32)
    my_class = models.ForeignKey('Classes', related_name='students')
    score = models.IntegerField()

class Teacher(models.Model):
    t_name = models.CharField(max_length=32)
    sex = models.IntegerField(choices=((1, 'male'), (2, 'female')))
    age = models.IntegerField()

class Classes(models.Model):
    c_name = models.CharField(max_length=32)
    teachers = models.ManyToManyField('Teacher', related_name='classes'
```

数据:

app01_classes

id	c_name
1	三年一班
2	三年二班
3	三年三班

app01_teacher

id	t_name	sex	age
1	太白	1	30
2	景女神	1	18
3	Alex	2	73

app01_student

id	s_name	score	my_class_id
1	小豪	45	1
2	小明	61	2
3	豪爽	94	3
4	明亮	86	1
5	小洋	55	2
6	小亮	98	3

app_01_classes_teachers

id	classes_id	teacher_id
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	3	1

题目

1.查询所有三年级的班级对象

```
ret = models.Classes.objects.filter(cname__startswith = '三年')
```

2.查询三年三班的所有同学的名字

```
ret = models.Student.objects.filter(my_class__cname = '三年三班').values('s_name')
```

```
ret = models.Classes.objects.filter(cname = '三年三班').student.set.all().values('s_name')
```

3.查询分数大于60分的同学的成绩和姓名

```
ret = models.Student.objects.filter(score__gte=60).values('s_name', 'score')
```

4.查询每个班级的名称和的学生人数

```
ret = models.Classes.objects.all().annotate(count=Count('student')).values('c_name',
'count')
```

5.查询年纪最大的老师姓名和年龄

```
age = models.Teacher.objects.aggregate(max=Max('age'))['max']

models.Teacher.objects.filter(age=models.Teacher.objects.aggregate(max=Max('age'))
['max']).values('t_name', 'age')
```

```
ret = models.Teacher.objects.order_by('-age').values('t_name', 'age').first()
```

6.分别查询出男女老师的个数

```
ret = models.Teacher.objects.values('sex').annotate(Count('sex'))
```

7.查询名字中有“豪”的学生姓名和成绩

```
ret = models.Student.objects.filter(s_name__contains='豪').values('s_name', 'score')
```

8.查询三年二班的平均成绩

```
ret1 = models.Student.objects.filter(my_class__c_name = "三年二班").aggregate(Avg('score'))
```

9.新增一个名为“三年四班”的班级

```
models.Classes.objects.create(c_name='三年四班')
```

10.给三年二班新增一个名为“小青”的同学（三种方法，不给score字段赋值也可）

```
models.Student.objects.create(s_name='小青', my_class_id=2)
```

```
models.Student.objects.create(s_name='小青',my_class=models.Classes.objects.get(c_name='三年二
班'))
```

```
models.Classes.objects.get(c_name='三年二班').student_set.create(s_name='小青', score=99)
```

11.给三年二班，新增一个名为“苑局”的30岁男老师

```
models.Classes.objects.get(c_name='三年三班')
```

```
a） 小红转班了，转到了三年四班
b） 给所有学生的成绩都加2分
```