```
import os,django

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

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "Jango_lession_157.settings")
django.setup()


from modelapp.models import User,Employee

#
# result = User.objects.all()
# print(result)

'''
# 排序 order_by(列名)   是manger的方法也是queryset的方法

result = User.objects.order_by("-pk")
print(result)

result = User.objects.order_by("age")
print(result)

result = User.objects.all().order_by("age")
print(result)

result = User.objects.filter(age=18).order_by("-name")
print(result)

#按年龄排序，年龄相同，则按姓名降序排列
result = User.objects.order_by("age","-name")
print(result)

# result = User.objects.order_by("age").order_by("-name")
# print(result)

'''


# 限制操作：取结果集中的某一部分数据
'''
result = User.objects.all()[1]
print(result)

users = User.objects.all()
for u in users:
    print(u)

result = User.objects.all()[1:3]
print(result)
```

```
result = User.objects.all()[1:]
print(result)

User.objects.filter(age=18)[1]

'''


'''
# 条件查询
# 下面两个方法中的条件查询 只能做等值查询

result = User.objects.get(pk=1)
print(result)

result=User.objects.filter(age=18)
print(result)


# result = User.objects.filter(id>3)   # 参数名=值     错误的
# print(result)

result=User.objects.filter(id__gt=3)   #id gt   select * from user where id>3
print(result)

result = User.objects.filter(age__gte=18)
print(result)
'''


# 模糊查询
# where name like "%a%" 可以查到Andy 不区分大小写
'''
result1 = User.objects.filter(name__contains="A")  #区分大小写
print(result1)

result2 = User.objects.filter(name__icontains="A")
print(result2)

result3 = User.objects.filter(name__istartswith="a")
print(result3)


result4 = User.objects.filter(name__iendswith="A")
print(result4)

'''


# 范围
'''

result = User.objects.filter(id__in=(1,2,4))
```

```
print(result)

result2 = User.objects.filter(age__in=(18,20,22))   # 表示非连续的多个值
print(result2)

result3 = User.objects.filter(age__range=(20,25))  # between 20 and 25
print(result3)


# result4 = User.objects.all(age__range=(20,25))  # 不能在all中加条件
# print(result4)

'''

#空值判断

# result = User.objects.filter(age__isnull=False)   # where age is null / age is not null
# print(result)


# result = User.objects.filter(birthday__year="1999")
# print(result)


# User.objects.filter()

# 映射查询： 查询部分列 values
'''
result = User.objects.values()  # 返回值不再是model对象的QuerySet 而是字典
print(result)

# 查所有行的id、name和age
result = User.objects.values("id","name","age") # 只查询id,name和age列
print(result)

# User.objects.all("id","name","age")  # 错误的 all方法不能接参数

# 查询部分行的id、name、age  部分行部分列
result = User.objects.filter(id__gt=3).values("id","name","age")
print(result)

'''


# result = User.objects.only("id","name")   #率先查询id和name列  不是只查询id name列
# print(result)



# 聚合函数
'''
result = User.objects.aggregate(Max("salary"),Avg("salary"))  # 返回值是字典

print(result)
```

```
result = User.objects.aggregate(ms = Max("salary"),avs = Avg("salary"))  # 返回值是字典
print(result)
# print(result["ms"])

'''


# 分组查询
'''
#以年龄分组，查询每组年龄中的最高薪水
result = User.objects.values("age").annotate(Max("salary"))
print(result)


#以年龄分组。查询每组中的最高薪水（要求id>3）    MySQL       where

result = User.objects.filter(id__gt=3).values("age").annotate(Max("salary"))
print(result)

#加条件：having条件-平均薪水>10000
result = User.objects.values("age").annotate(Max("salary"),avg =
Avg("salary")).filter(avg__gt=5000)
print(result)

'''

# 加排序
# result = User.objects.values("age").annotate(Max("salary")).order_by("age")
# print(result)


# F()和Q()函数

# 查询id大于年龄的用户

# result = User.objects.filter(id__gt=F("age"))  # 查询条件需要另外的列
# print(result)


# blog        发表时间    修改时间

# blog     修改__gt = F(发表)


# result = User.objects.filter(id__lte=4,name__icontains="a")
# print(result)
#
# # 查询id<=4或名字包含a
#
# result = User.objects.filter(Q(id__lte=4)  |  Q(name__icontains="a"))
# print(result)

#
```

```
# result = User.objects.filter(Q(id__lte=4)  |  ~Q(name__icontains="a"))
# print(result)


# result = User.objects.raw("select * from modelapp_user where id>2")[1]
# print(result)









# ----------------------auto_now  auto_now_add区别------------------------ #
# 添加一条数据
'''
 name = models.CharField(max_length=20,default="Jingjing")  #字段 列
    age = models.SmallIntegerField(null=True,unique=True,db_index=True)
    gender = models.BooleanField(default=True,db_column="sex")
    salary = models.FloatField()
    salary2 = models.DecimalField(max_digits=7,decimal_places=2)
    birthday = models.DateTimeField(auto_now_add=True) #首次添加时间
    birthday2 = models.DateTimeField(auto_now=True)  #更新时间
    commont = models.TextField()
'''

Employee.objects.create(age=38,gender=True,salary=1241,salary2=23421,commont="BBBBB")

# emp = Employee.objects.get(pk=2)
# emp.name="Mr_lee"
# emp.save()
```