Setting

0.pip install virtualenv
1.py -m venv myvenv สร้าง venv
2.myvenv\Scripts\activate.bat
3.pip install django
4.pip install psycopg2
5.django-admin startproject ชื่อproject
6.python manage.py startapp ชื่อapp

7.Set db

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql",
        "NAME": "blogs",
        "USER": "postgres",
        "PASSWORD": "password",
        "HOST": "localhost",
        "PORT": "5432",
    }
}
```

8.python manage.py makemigrations
9.python manage.py migrate

Setting notebook

1.pip install django-extensions ipython jupyter notebook
2.pip install ipython==8.25.0 jupyter_server==2.14.1 jupyterlab==4.2.2 jupyterlab_server==2.27.2

3.pip install notebook==6.5.7
4.mkdir notebooks
5.

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django_extensions",
    "blogs",
]
```

6.python manage.py shell_plus --notebook
7.

```
import
osos.environ["DJANGO_ALLOW_ASYNC_UNSAFE"] =
"true"
```

ป.ล.อย่าลืมเปลี่ยน kernel เป็น shell_plus

import

```
from django.db.models import *
from django.db.models.functions import *
from django.db.models.lookups import *

import json (print(json.dumps(dictionary, indent=4,
sort_keys=False))
)
from ชื่อapp.models import *
```

Function

```
test_pro = Product.objects.filter(name__startswith = "Air").values()
for i in test_pro:
    print(i['name'])
```

```
Air Conditioner
```

```
]: test_pro = Product.objects.filter(name__endswith = "ner").values()
   for i in test_pro:
       print(i['name'])
```

```
Air Conditioner
Vacuum Cleaner
```

```
: test_pro = Product.objects.filter(name__icontains = "Ner").values()
  for i in test_pro:
      print(i['name'])
```

```
Air Conditioner
Vacuum Cleaner
Nerf N-Strike Elite Strongarm Blaster
```

```
test_pro = Product.objects.filter(categories__name__in = ("Information Technology", "Pet Supplies")).values()
for i in test_pro:
    print(i['name'])
```

```
Smartphone
Laptop
Smart TV
Bluetooth Earphones
Tablet
Gaming Console
Digital Camera
Wireless Router
Portable Power Bank
Smartwatch
Dog Bed
```

Entry.objects.filter(pub_date__year__gte=2005)

Entry.objects.filter(pub_date__isnull=True)

.query

```
test_pro = Product.objects.filter(price__range = (8000, 50000)).values()
for i in test_pro:
    print(i['name']+" "+str(i['price']))
```

```
Laptop 25999.00
Smart TV 8900.00
Tablet 12900.00
Digital Camera 32000.00
Refrigerator 9000.00
Air Conditioner 18900.00
Gemstone Ring 9000.00
```

คำนวน filed ในตารางเดียวกัน(F)

```
]: test_pro = Payment.objects.annotate(discount_price = F('price')-F('discount')).filter(discount_price__gt=5000).values()

for i in test_pro:
    print(str(i['discount_price']))
```

```
8868.22
1087753.80
19897.65
6030.04
6001.07
1080267.36
56582.84
```

Q

```
]: test_pro = Product.objects.filter(Q(categories__name = "Pet Supplies") | Q(categories__name = "Clothing and Apparel"))

for i in test_pro:
    print(i.name)
```

```
Men's T-Shirt
Women's Jeans
Unisex Hoodie
Women's Dress
Men's Formal Shirt
Women's Athletic Shorts
Men's Polo Shirt
Women's Sweater
Unisex Cap
Men's Shorts
Dog Bed
```

Exclude

```
product = Product.objects.exclude(id=10)[:11]
```

สร้าง, ลบ, แก้ไข ข้อมูล

สร้าง

```
cate_book = ProductCategory.objects.get(name='Books and Toys')

pro1 = Product.objects.create(name="Philosopher's Stone (1997)", remaining_amount= 20, price=790, description="By J. K. Rowling.'
pro1.categories.add(cate_book)
pro1.save()

cate_book = ProductCategory.objects.get(name='Books and Toys')

pro2 = Product.objects.create(name="Me Before You", remaining_amount= 40, price=390, description="A romance novel written by Jojc
pro2.categories.add(cate_book)
pro2.save()


cate_tech = ProductCategory.objects.get(name='Information Technology')
cate_elect = ProductCategory.objects.get(name='Electronics')

pro3 = Product.objects.create(name="Notebook HP Pavilion Silver", remaining_amount= 10, price=20000, description="Display Screen.
pro3.categories.add(cate_tech, cate_elect)
pro3.save()
```

แก้ไข

2.2 แก้ไขชื่อสินค้า จาก `Philosopher's Stone (1997)` เป็น `Half-Blood Prince (2005)` (0.5 คะแนน)

```
book = Product.objects.filter(name="Philosopher's Stone (1997)")

book.name = "Half-Blood Prince (2005)"


for i in book:
    i.name = "Half-Blood Prince (2005)"
    i.save()
```

```
cate = ProductCategory.objects.get(name="Books")

cate.name = "Books and Toys"

cate.save()
```

ลบ

```
product = Product.objects.filter(categories__name = "Books and Toys")

product.delete()
```

Join

1.One to Many

-อ้างอิงใน filter

-ฝั่ง many (อ้างอิงโดยใช้ field ที่ เป็น Fk)

test_pro = Cart.objects.filter(customer__first_name = "Kittitorn")

-ฝั่ง one (อ้างอิงโดยใช้ชื่อ model ของ many ได้เลย)

```
test_pro1 = Customer.objects.filter(order__remark = "EIEI") #เอาลูกค้าที่มี remark = EIEI

# print(test_pro['order__remark'])
for i in test_pro1:
    for j in i.order_set.all():
        print(j.remark)
```

-อ้างอิงเพื่อเอาค่า

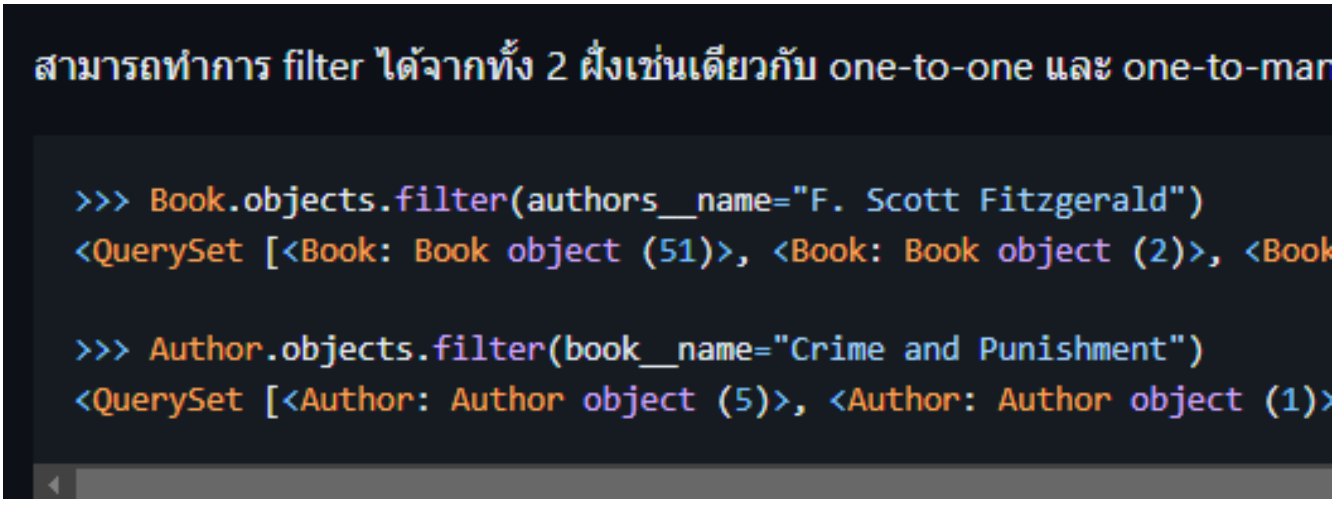
-ฝั่ง many

print(i.customer.first_name)

-ฝั่ง one (อ้างอิงโดยใช้ชื่อ model ของ many และใช้ _set)

cus.order_set.all()

2.Many to Many



เพิ่ม ค.ส.พ ให้ many to many

```
> from books.models import Book, Author
> a1 = Author.objects.get(pk=1)
> a2 = Author.objects.get(pk=2)

> book = Book.objects.get(pk=10)
> book.authors.add(a1, a2)
```

```
>>> from books.models import Book, Author
>>> b1 = Book.objects.get(pk=11)
>>> b2 = Book.objects.get(pk=12)

>>> author = Author.objects.get(pk=10)
>>> author.book_set.add(b1, b2)

>>> author.book_set.all()
<QuerySet [<Book: Book object (11)>, <Boo
```

สามารถทำการ ยกเลิก ความสัมพันธ์ ได้โดยใช้ `remove()` หรือ `clear()` ถ้าต้องการลบความสัมพันธ์ทั้งหมด

```
>>> book = Book.objects.get(pk=10)

>>> book.authors.remove(a1)
>>> book.authors.all()
<QuerySet [<Author: Author object (2)>, <Author: Author object (5)>]>

>>> book.authors.clear()
>>> book.authors.all()
<QuerySet []>
```

โจทย์

1. annotate(), F()

1.1 ให้นักศึกษาค้นหาข้อมูล `Payment` โดยให้เพิ่ม field ราคาที่ลบกับส่วนลดแล้ว โดยกำหนดให้ชื่อ field ว่า "after_discount_price" โดยใช้แสดงข้อมูล 10 ตัวแรกเรียงตาม "after_discount_price" จากมากไปน้อย (0.25 คะแนน)

```
payment1 = Payment.objects.annotate(after_discount_price=F("price")-F("discount")).order_by("-after_discount_price")[:10]

for i in payment1:
    print("ID: "+str(i.id)+", PRICE: "+ str(i.price)+", DISCOUNT "+str(i.discount)+", AFTER_DISCOUNT "+str(i.after_discount_price
```

payment1 = Payment.objects.annotate(after_discount_price=F("price")-F("discount")).order_by("-after_discount_price")[:10]

for i in payment1:
print("ID: "+str(i.id)+", PRICE: "+ str(i.price)+", DISCOUNT "+str(i.discount)+", AFTER_DISCOUNT "+str(i.after_discount_price))

1.2 ต่อเนื่องจากข้อ 1.1 ให้ filter เฉพาะข้อมูล `Payment` ที่มี "after_discount_price" มากกว่า 500,000 (0.25 คะแนน)

```
]: payment2 = Payment.objects.annotate(after_discount_price=F("price")-F("discount")).filter(after_discount_price__gt = 500000)
   .order_by("-after_discount_price")

   for i in payment2:
       print("ID: "+str(i.id)+", PRICE: "+ str(i.price)+", DISCOUNT "+str(i.discount)+", AFTER_DISCOUNT "+str(i.after_discount_price
```

payment2 = Payment.objects.annotate(after_discount_price=F("price")-F("discount")).filter(after_discount_price__gt = 500000).order_by("-after_discount_price")

for i in payment2:
print("ID: "+str(i.id)+", PRICE: "+ str(i.price)+", DISCOUNT "+str(i.discount)+", AFTER_DISCOUNT "+str(i.after_discount_price))

1.3 ให้นักศึกษาเรียงลำดับข้อมูลลูกค้า (`Customer`) โดยเรียงลำดับตามลำดับตัวอักษร `น้อยไปมาก` จากชื่อเต็มของลูกค้า (`full_name`) โดยแสดง 5 คนแรก (0.5 คะแนน)

Hint: Field `full_name` นั้นจะต้องถูก annotate ขึ้นมาโดยการนำ `first_name` มาต่อกับ `last_name` โดยใช้ `Concat(*expressions, **extra)`

Hint: แปลง object เป็น dict ใช้ `values()` doc

```
>>> Blog.objects.filter(name__startswith="Beatles").values()
<QuerySet [{'id': 1, 'name': 'Beatles Blog', 'tagline': 'All the latest Beatles news.'}]>
```

Hint: อยาก print dictionary สวยๆ ใช้ `json.dumps`

```
print(json.dumps(dictionary, indent=4, sort_keys=False))
```

```
import json

customer = Customer.objects.annotate(full_name = Concat("first_name", Value(" "),"last_name"))
.order_by("full_name").values("id", "email", "address", "full_name")[:5]

for i in customer:
    print(json.dumps(i, indent=4, sort_keys=False))
```

import json

customer = Customer.objects.annotate(full_name = Concat("first_name", Value(" "),"last_name"))
.order_by("full_name").values("id", "email", "address", "full_name")[:5]

for i in customer:
    print(json.dumps(i, indent=4, sort_keys=False))

3. aggregation - count(), sum(), AVG()

3.1 ให้นักศึกษาหาค่าเฉลี่ยของราคาสินค้า (`Product.price`) ที่มีจำนวนคงเหลือ (`Product.remaining_amount`) มากกว่า 0 (0

```
{'avg': Decimal('29308.000000000000')}
```

```
: Product.objects.filter(remaining_amount__gt = 0).aggregate(avg=Avg("price"))
```

Product.objects.filter(remaining_amount__gt = 0).aggregate(avg=Avg("price"))

3.2 จงหาผลรวมราคา (`CartItem.product.price`) ที่อยู่ในตระกร้าสินค้าของเดือน `พฤษภาคม` (ดูจาก `Cart.create_date`) (0.

```
{'sum': Decimal('9912555.00')}
```

```
: CartItem.objects.filter(cart__create_date__month = 5).aggregate(sum=Sum("product__price"))
```

CartItem.objects.filter(cart__create_date__month = 5).aggregate(sum=Sum("product__price"))

.3 ให้นักศึกษานับจำนวนสินค้าที่อยู่ประเภท `Electronics` , `Jewelry` และ ราคาของสินค้าอยู่ในช่วง 8,000.00 - 50,000.00 (

```
PRODUCT CATEGORY NAME: Electronics, PRODUCT COUNT: 6
PRODUCT CATEGORY NAME: Jewelry, PRODUCT COUNT: 1
```

```
: product = Product.objects.filter(Q(categories__name='Electronics') | Q(categories__name='Jewelry'), Q(price__range=(8000,50000))
.values('categories__name').annotate(count=Count('id'))

for i in product:
    print(i['categories__name'], i['count'])
```

product = Product.objects.filter(Q(categories__name='Electronics') | Q(categories__name='Jewelry'), Q(price__range=(8000,50000))).values('categories__name').annotate(count=Count('id'))

for i in product:
    print(i['categories__name'], i['count'])

5. many-to-many

5.1 ให้นักศึกษาค้นหาข้อมูลสินค้า (`Product`) ที่อยู่ในประเภทสินค้า "Information Technology" 10 รายการแรก (เรียงลำดับด้วย `Product.id`) และแสดงชื่อประเภทสินค้า (`ProductCategory`) (0.25 คะแนน)

ตัวอย่าง Output บางส่วน

```
PRODUCT ID: 1, PRODUCT NAME: Smartphone, PRODUCT CATEGORY: Information technology, Electronics
PRODUCT ID: 2, PRODUCT NAME: Laptop, PRODUCT CATEGORY: Information technology, Electronics
PRODUCT ID: 3, PRODUCT NAME: Smart TV, PRODUCT CATEGORY: Information technology, Electronics
```

```
pro = Product.objects.filter(categories__name = "Information Technology").order_by("id")[:10]

for i in pro:
    print("PRODUCT ID: "+str(i.id)+", PRODUCT NAME: "+i.name, end =", PRODUCT CATEGORY: ")
    count = i.categories.all().count()
    for j in i.categories.all():
        print(j.name, end=" ")
        count -= 1
        if count != 0:
            print(",", end=" ")
    print("\n")
```

pro = Product.objects.filter(categories__name = "Information Technology").order_by("id")[:10]

for i in pro:
    print("PRODUCT ID: "+str(i.id)+", PRODUCT NAME: "+i.name, end =", PRODUCT CATEGORY: ")
    count = i.categories.all().count()
    for j in i.categories.all():
        print(j.name, end=" ")
        count -= 1
        if count != 0:
            print(",", end=" ")
    print("\n")

5.2 ให้นักศึกษาทำตามขั้นตอนดังนี้ (0.25 คะแนน)

หมายเหตุ: ถ้าใช้ DB จาก WEEK4 `Books and Media` อาจจะถูกเปลี่ยนเป็น `Books` แล้ว

1. เปลี่ยนชื่อประเภทสินค้า `Books and Media` เป็น `Books and Toys`
2. ลบประเภท `Toys and Games` ออกโดยให้ใช้เป็น `Books and Toys` แทน
3. ค้นหาว่าสินค้าที่มีประเภทสินค้าเป็น `Books and Toys` ทั้งหมดมีจำนวนเท่าไหร่

In [55]:
```
book_media = ProductCategory.objects.get(name = "Books")
book_media.name = "Books and Toys"
book_media.save()
```

In [72]:
```
book_toy_cate = ProductCategory.objects.get(name = "Books and Toys")

product = Product.objects.filter(categories__name = "Toys and Games")

for i in product:
    book_toy_cate.product_set.add(i)
```

In [73]:
```
#ลบ ความสัมพันธ์ และลบประเภท Toys and Games
game_toy_cate = ProductCategory.objects.get(name = "Toys and Games")
game_toy_cate.product_set.clear()

game_toy_cate.delete()
```

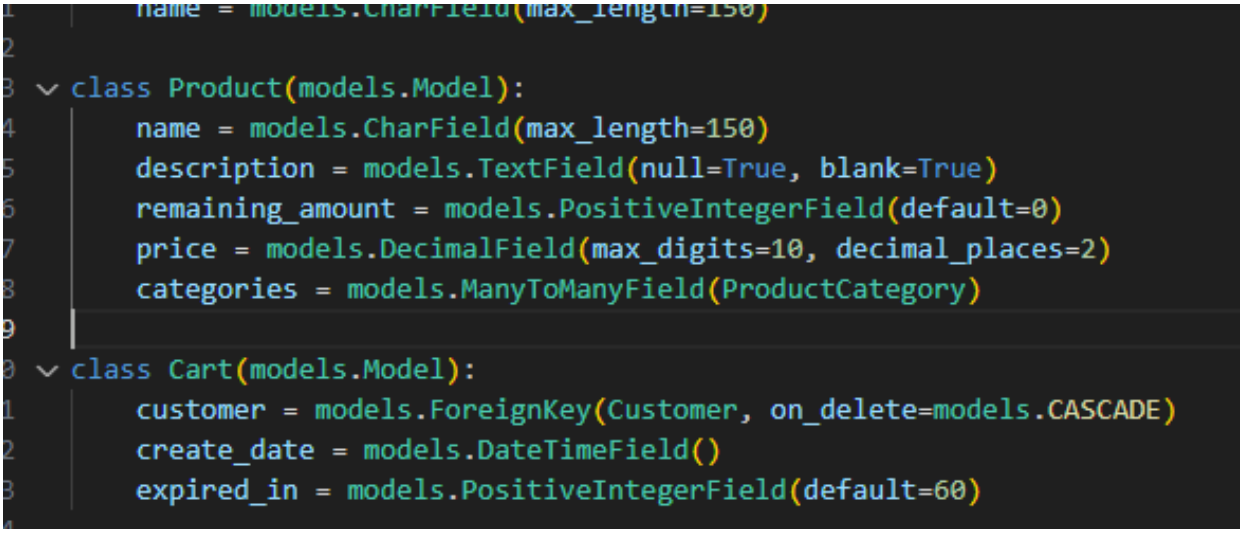
Out[73]:
```
(1, {'shop.ProductCategory': 1})
```

In [74]:
```
bt_product = Product.objects.filter(categories__name = "Books and Toys").count()


print(bt_product)
```

```
11
```

Ex model



```
    name = models.CharField(max_length=150)

class Product(models.Model):
    name = models.CharField(max_length=150)
    description = models.TextField(null=True, blank=True)
    remaining_amount = models.PositiveIntegerField(default=0)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    categories = models.ManyToManyField(ProductCategory)

class Cart(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    create_date = models.DateTimeField()
    expired_in = models.PositiveIntegerField(default=60)
```