django

1.pip install virtualenv

2.py -m venv myvenv

3.myvenv\Scripts\activate.bat

4.pip install django psycopg2-binary

5.django-admin startproject ชื่อโปรเจค

cd เข้าไปด้วย

6.python manage.py startapp ชื่อแอป

7.# Database setting

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


# Add app blogs to INSTALLED_APPS
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
```

"django.contrib.messages",

"django.contrib.staticfiles",

Add your apps here

"ชื่อแอป",

]

8.ใส่model ลงใน ไฟล model.py

9.python manage.py makemigrations

10.python manage.py migrate

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.เปลี่ยนในsetting

INSTALLED_APPS = [

"django.contrib.admin",

"django.contrib.auth",

"django.contrib.contenttypes",

"django.contrib.sessions",

"django.contrib.messages",

"django.contrib.staticfiles",

"django_extensions",

“ชื่อแอป”

]

6.python manage.py shell_plus --notebook

7.import os

os.environ["DJANGO_ALLOW_ASYNC_UNSAFE"] = "true"

ใช้ทุกครั้งที่เปิด

ทำงาน

จากนั้นให้ทำการ migrate และ run คำสั่งในไฟล์ `shop.sql` ใน PgAdmin4

```
In [1]: import os
        from shop.models import *
        os.environ["DJANGO_ALLOW_ASYNC_UNSAFE"] = "true"
```

from ชื่อแอป.model import *

____.objects คือ ข้างหน้าเป็นชื่อตาราง

1. ให้นักศึกษา Query ค้นหาข้อมูลมาแสดงให้ถูกต้องตามโจทย์

1.1 query หาข้อมูล `Order` ทั้งหมดที่เกิดขึ้นในเดือน `พฤษภาคม` มาแสดงผล 10 รายการแรก และแสดงผลดังตัวอย่าง (0.5 คะแนน)

```
ORDER ID:22, DATE: 2024-05-01, PRICE: 4890.00
ORDER ID:23, DATE: 2024-05-01, PRICE: 2540.00
ORDER ID:24, DATE: 2024-05-01, PRICE: 1720.00
ORDER ID:25, DATE: 2024-05-02, PRICE: 322499.00
ORDER ID:26, DATE: 2024-05-02, PRICE: 3399.00
ORDER ID:27, DATE: 2024-05-02, PRICE: 1190.00
ORDER ID:28, DATE: 2024-05-03, PRICE: 9499.00
ORDER ID:29, DATE: 2024-05-03, PRICE: 700.00
ORDER ID:30, DATE: 2024-05-03, PRICE: 1690.00
ORDER ID:31, DATE: 2024-05-04, PRICE: 3280.00
```

```
In [2]: ordermay = Order.objects.filter(order_date__month=5)[:10]
        for i in ordermay:
            print(f"ORDER ID: {i.id}, DATE: {i.order_date}, PRICE: {i.payment.price}")
```

```
ORDER ID: 22, DATE: 2024-05-01, PRICE: 4890.00
ORDER ID: 23, DATE: 2024-05-01, PRICE: 2540.00
ORDER ID: 24, DATE: 2024-05-01, PRICE: 1720.00
ORDER ID: 25, DATE: 2024-05-02, PRICE: 322499.00
ORDER ID: 26, DATE: 2024-05-02, PRICE: 3399.00
ORDER ID: 27, DATE: 2024-05-02, PRICE: 1190.00
ORDER ID: 28, DATE: 2024-05-03, PRICE: 9499.00
ORDER ID: 29, DATE: 2024-05-03, PRICE: 700.00
ORDER ID: 30, DATE: 2024-05-03, PRICE: 1690.00
ORDER ID: 31, DATE: 2024-05-04, PRICE: 3280.00
```

[:10]คือเอาแค่10ตัวแรก

1.2 query หาข้อมูล `Product` ที่มีคำลงท้ายว่า `features.` ในรายละเอียดสินค้า และแสดงผลดังตัวอย่าง (0.5 คะแนน)

```
PRODUCT ID: 1, DESCRIPTION: A sleek and powerful smartphone with advanced features.
PRODUCT ID: 7, DESCRIPTION: High-resolution digital camera with advanced photography features.
PRODUCT ID: 10, DESCRIPTION: A stylish smartwatch with health monitoring and notification feat
ures.
PRODUCT ID: 14, DESCRIPTION: Split air conditioner with remote control and energy-saving featu
res.
PRODUCT ID: 45, DESCRIPTION: Customizable racing track set with loop and jump features.
```

[3]:
```python
product = Product.objects.filter(description__endswith='features.')
for i in product:
    print(f"PRODUCT ID: {i.id}, DESCRIPTION: {i.description}")
```

```
PRODUCT ID: 1, DESCRIPTION: A sleek and powerful smartphone with advanced features.
PRODUCT ID: 7, DESCRIPTION: High-resolution digital camera with advanced photography features.
PRODUCT ID: 10, DESCRIPTION: A stylish smartwatch with health monitoring and notification features.
PRODUCT ID: 14, DESCRIPTION: Split air conditioner with remote control and energy-saving features.
PRODUCT ID: 45, DESCRIPTION: Customizable racing track set with loop and jump features.
```

1.3 query หาข้อมูล `Product` ที่มีราคาสินค้าตั้งแต่ `5000.00` ขึ้นไป และอยู่ในหมวดหมู่ `Information Technology` และแสดงผลดังตัวอย่าง (0.5 คะแนน)

```
 # ตัวอย่างบางส่วน
PRODUCT ID: 1, NAME: Smartphone, PRICE: 5900.00
```

[6]:
```python
productgt = Product.objects.filter(price__gte='5000',categories__name="Information Technology")
for i in productgt:
    print(f"Product ID {i.id}, NAME: {i.name}, PRICE: {i.price}")
```

```
Product ID 1, NAME: Smartphone, PRICE: 5900.00
Product ID 2, NAME: Laptop, PRICE: 25999.00
Product ID 3, NAME: Smart TV, PRICE: 8900.00
Product ID 5, NAME: Tablet, PRICE: 12900.00
Product ID 6, NAME: Gaming Console, PRICE: 5000.00
Product ID 7, NAME: Digital Camera, PRICE: 32000.00
```

2. เพิ่ม ลบ แก้ไข สินค้า

2.1 ให้เพิ่มสินค้าใหม่จำนวน 3 รายการ (0.5 คะแนน)

```
สินค้าที่ 1
ชื่อ: Philosopher's Stone (1997)
หมวดหมู่สินค้า: Books and Media
จำนวนคงเหลือ: 20
รายละเอียดซ: By J. K. Rowling.
ราคา: 790

สินค้าที่ 2
ชื่อ: Me Before You
หมวดหมู่สินค้า: Books and Media
จำนวนคงเหลือ: 40
รายละเอียดซ: A romance novel written by Jojo
ราคา: 390

สินค้าที่ 3
ชื่อ: Notebook HP Pavilion Silver
หมวดหมู่สินค้า: Information Technology และ Electronics
จำนวนคงเหลือ: 10
รายละเอียดซ: Display Screen. 16.0
ราคา: 20000
```

In [9]:
```
ad1 = Product.objects.create(name="Philosopher's Stone (1997)", remaining_amount=20, description="By J. K. Rowling.", price=790)
ad2 = Product.objects.create(name="Me Before You", remaining_amount=40, description="A romance novel written by Jojo", price=390)
ad3 = Product.objects.create(name="Notebook HP Pavilion Silver", remaining_amount=10, description="Display Screen. 16.0", price=
c1 = ProductCategory.objects.get(name="Books and Media")
c2 = ProductCategory.objects.get(name="Information Technology")
c3 = ProductCategory.objects.get(name="Electronics")
ad1.categories.add(c1)
ad2.categories.add(c1)
ad3.categories.add(c2)
ad3.categories.add(c3)
```

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

:
```
fix = Product.objects.get(name ="Philosopher's Stone (1997)")
fix.name = "Half-Blood Prince (2005)"
fix.save()
```

:

2.3 แก้ไขชื่อหมวดหมู่สินค้า จาก `Books and Media` เป็น `Books` (0.5 คะแนน)

[19]:
```
book = ProductCategory.objects.get(name="Books and Media")
book.name = "Books"
book.save()
```

2.4 ลบสินค้าทุกตัวที่อยู่ในหมวดหมู่ `Books` (0.5 คะแนน)

In [22]:
```
Product.objects.filter(categories__name="Books").delete()
```

Out[22]:
```
(0, {})
```

1. **exact**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ตรงกับค่าที่ระบุอย่างแม่นยำ
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__exact='Hello World')
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE headline = 'Hello World';

2. **iexact**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ตรงกับค่าที่ระบุอย่างแม่นยำ โดยไม่สนใจตัวพิมพ์ใหญ่/เล็ก
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__iexact='hello world')
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE LOWER(headline) = 'hello world';

3. **contains**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่มีค่าที่ระบุอยู่ในฟิลด์ โดยสนใจตัวพิมพ์ใหญ่/เล็ก
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__contains='Lennon')
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE headline LIKE '%Lennon%';

4. **icontains**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่มีค่าที่ระบุอยู่ในฟิลด์ โดยไม่สนใจตัวพิมพ์ใหญ่/เล็ก
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__icontains='Lennon')
- **SQL ที่สร้าง:**

- sql
- Copy code
- SELECT ... WHERE headline ILIKE '%Lennon%';

5. **startswith**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่เริ่มต้นด้วยค่าที่ระบุ
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__startswith='Hello')
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE headline LIKE 'Hello%';

6. **endswith**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ลงท้ายด้วยค่าที่ระบุ
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__endswith='World')
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE headline LIKE '%World';

7. **in**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ค่าของฟิลด์นั้นอยู่ในลิสต์ที่ระบุ
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(headline__in=('a', 'b', 'c'))
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE headline IN ('a', 'b', 'c');

8. **gt, gte, lt, lte**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ค่าของฟิลด์นั้นมากกว่า (gt), มากกว่าหรือเท่ากับ (gte), น้อยกว่า (lt), หรือน้อยกว่าหรือเท่ากับ (lte) ค่าที่ระบุ
- **ตัวอย่าง:**
- python
- Copy code

- Entry.objects.filter(id__gt=4)
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE id > 4;

9. **range**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ค่าของฟิลด์นั้นอยู่ในช่วงที่กำหนด
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(pub_date__range=(start_date, end_date))
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE pub_date BETWEEN '2005-01-01' AND '2005-03-31';

10. **date, year, month, day, week, week_day**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ตามค่าที่ระบุในฟิลด์วันที่
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(pub_date__year=2005)
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE pub_date BETWEEN '2005-01-01' AND '2005-12-31';

11. **isnull**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ฟิลด์นั้นเป็น NULL หรือไม่เป็น NULL
- **ตัวอย่าง:**
- python
- Copy code
- Entry.objects.filter(pub_date__isnull=True)
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE pub_date IS NULL;

12. **regex**

- **ใช้งาน:** ใช้เพื่อกรองผลลัพธ์ที่ค่าของฟิลด์นั้นตรงกับ pattern ของ regular expression ที่ระบุ
- **ตัวอย่าง:**
- python

- Copy code
- Entry.objects.get(title__regex=r"^(An?|The) +")
- **SQL ที่สร้าง:**
- sql
- Copy code
- SELECT ... WHERE title ~ '^(An?|The) +'; -- PostgreSQL

การพิมพ์ SQL Query ด้วย .query

- **HINT:** ถ้าอยากเห็น SQL Query ที่ Django ORM สร้างขึ้นจาก QuerySet สามารถทำได้โดยเพิ่ม .query หลังจากคำสั่ง query
- **ตัวอย่าง:**
- python
- Copy code
- print(Entry.objects.filter(headline__contains='Lennon').query)

ทั้งหมดนี้คือ lookup types หลัก ๆ ที่ใช้กับ filter(), exclude(), และ get() ใน Django ORM ที่ช่วยให้คุณสามารถสร้างคำสั่ง WHERE ใน SQL ได้อย่างง่ายดาย

4o

การor ต้อง import

from django.db.models.functions import *

from django.db.models import *

from django.db.models import Q

entries = Entry.objects.filter(Q(pub_date__year=2004, pub_date__month=5) | Q(pub_date__year=2005))

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)

หมายเหตุ: จะต้องใช้ annotate() นะครับ ให้เอา `Payment.price` - `Payment.discount`

ตัวอย่าง Output

```
ID: 92, PRICE: 1200500.00, DISCOUNT 29433.25, AFTER_DISCOUNT 1171066.75
ID: 82, PRICE: 1200280.00, DISCOUNT 46229.40, AFTER_DISCOUNT 1154050.60
ID: 137, PRICE: 1200690.00, DISCOUNT 71407.25, AFTER_DISCOUNT 1129282.75
ID: 105, PRICE: 1200390.00, DISCOUNT 105019.11, AFTER_DISCOUNT 1095370.89
ID: 45, PRICE: 1218900.00, DISCOUNT 126859.95, AFTER_DISCOUNT 1092040.05
ID: 7, PRICE: 1201200.00, DISCOUNT 113446.20, AFTER_DISCOUNT 1087753.80
ID: 18, PRICE: 1202190.00, DISCOUNT 121922.64, AFTER_DISCOUNT 1080267.36
ID: 77, PRICE: 379000.00, DISCOUNT 19397.00, AFTER_DISCOUNT 359603.00
ID: 127, PRICE: 320450.00, DISCOUNT 14578.90, AFTER_DISCOUNT 305871.10
ID: 125, PRICE: 320399.00, DISCOUNT 17939.55, AFTER_DISCOUNT 302459.45
```

In [2]:
```
from django.db.models import *
after= Payment.objects.annotate(after_discount_price=F('price')-F('discount')).order_by('-after_discount_price')[:10]
for i in after:
    print(f"ID: {i.id}, PRICE: {i.price}, DISCOUNT {i.discount}, AFTER_DISCOUNT {i.after_discount_price}")
```

```
ID: 92, PRICE: 1200500.00, DISCOUNT 29433.25, AFTER_DISCOUNT 1171066.75
ID: 82, PRICE: 1200280.00, DISCOUNT 46229.40, AFTER_DISCOUNT 1154050.60
ID: 137, PRICE: 1200690.00, DISCOUNT 71407.25, AFTER_DISCOUNT 1129282.75
ID: 105, PRICE: 1200390.00, DISCOUNT 105019.11, AFTER_DISCOUNT 1095370.89
ID: 45, PRICE: 1218900.00, DISCOUNT 126859.95, AFTER_DISCOUNT 1092040.05
ID: 7, PRICE: 1201200.00, DISCOUNT 113446.20, AFTER_DISCOUNT 1087753.80
ID: 18, PRICE: 1202190.00, DISCOUNT 121922.64, AFTER_DISCOUNT 1080267.36
ID: 77, PRICE: 379000.00, DISCOUNT 19397.00, AFTER_DISCOUNT 359603.00
ID: 127, PRICE: 320450.00, DISCOUNT 14578.90, AFTER_DISCOUNT 305871.10
ID: 125, PRICE: 320399.00, DISCOUNT 17939.55, AFTER_DISCOUNT 302459.45
```

from django.db.models import *

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

for i in after:

print(f"ID: {i.id}, PRICE: {i.price}, DISCOUNT {i.discount}, AFTER_DISCOUNT {i.after_discount_price}")

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

ตัวอย่าง Output

```
ID: 92, PRICE: 1200500.00, DISCOUNT 29433.25, AFTER_DISCOUNT 1171066.75
ID: 82, PRICE: 1200280.00, DISCOUNT 46229.40, AFTER_DISCOUNT 1154050.60
ID: 137, PRICE: 1200690.00, DISCOUNT 71407.25, AFTER_DISCOUNT 1129282.75
ID: 105, PRICE: 1200390.00, DISCOUNT 105019.11, AFTER_DISCOUNT 1095370.89
ID: 45, PRICE: 1218900.00, DISCOUNT 126859.95, AFTER_DISCOUNT 1092040.05
ID: 7, PRICE: 1201200.00, DISCOUNT 113446.20, AFTER_DISCOUNT 1087753.80
ID: 18, PRICE: 1202190.00, DISCOUNT 121922.64, AFTER_DISCOUNT 1080267.36
```

In [7]:
```
after= Payment.objects.annotate(after_discount_price=F('price')-F('discount')).filter(after_discount_price__gt=500000).order_by('
for i in after:
    print(f"ID: {i.id}, PRICE: {i.price}, DISCOUNT {i.discount}, AFTER_DISCOUNT {i.after_discount_price}")
```

```
ID: 92, PRICE: 1200500.00, DISCOUNT 29433.25, AFTER_DISCOUNT 1171066.75
ID: 82, PRICE: 1200280.00, DISCOUNT 46229.40, AFTER_DISCOUNT 1154050.60
ID: 137, PRICE: 1200690.00, DISCOUNT 71407.25, AFTER_DISCOUNT 1129282.75
ID: 105, PRICE: 1200390.00, DISCOUNT 105019.11, AFTER_DISCOUNT 1095370.89
ID: 45, PRICE: 1218900.00, DISCOUNT 126859.95, AFTER_DISCOUNT 1092040.05
ID: 7, PRICE: 1201200.00, DISCOUNT 113446.20, AFTER_DISCOUNT 1087753.80
ID: 18, PRICE: 1202190.00, DISCOUNT 121922.64, AFTER_DISCOUNT 1080267.36
```

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

```
for i in after:

    print(f"ID: {i.id}, PRICE: {i.price}, DISCOUNT {i.discount}, AFTER_DISCOUNT {i.after_discount_price}")
```

Aggregate Functions

1. **Avg**: หาค่าเฉลี่ยของฟิลด์ที่ระบุ
2. python
3. Copy code
4. from django.db.models import Avg
5.
6. avg_price = Product.objects.aggregate(Avg('price'))
7. # Output: {'price__avg': 200.0}
8. **Sum**: หาผลรวมของฟิลด์ที่ระบุ
9. python
10. Copy code
11. from django.db.models import Sum
12.
13. total_price = Product.objects.aggregate(Sum('price'))
14. # Output: {'price__sum': 1000.0}
15. **Count**: นับจำนวนวัตถุใน QuerySet
16. python
17. Copy code
18. from django.db.models import Count
19.
20. total_products = Product.objects.aggregate(Count('id'))
21. # Output: {'id__count': 5}
22. **Max**: หาค่าสูงสุดของฟิลด์ที่ระบุ
23. python
24. Copy code
25. from django.db.models import Max
26.
27. max_price = Product.objects.aggregate(Max('price'))
28. # Output: {'price__max': 500.0}
29. **Min**: หาค่าต่ำสุดของฟิลด์ที่ระบุ
30. python
31. Copy code
32. from django.db.models import Min
33.
34. min_price = Product.objects.aggregate(Min('price'))
35. # Output: {'price__min': 100.0}

Annotate Functions

annotate() ใช้ในการเพิ่มฟิลด์ใหม่ที่ถูกคำนวณไปใน QuerySet ทำให้สามารถใช้ฟิลด์นี้ในส่วนอื่น ๆ ของ

.

QuerySet ได้ เช่นในการกรองหรือสั่งข้อมูล ตัวอย่าง:

1. **annotate** กับ **Count**
2. python
3. Copy code
4. from django.db.models import Count
5.
6. products_with_order_count = Product.objects.annotate(order_count=Count('order'))
7. for product in products_with_order_count:
8. print(product.name, product.order_count)
9. **annotate** กับ **Sum**
10. python
11. Copy code
12. from django.db.models import Sum
13.
14. products_with_total_sales = Product.objects.annotate(total_sales=Sum('order__total'))
15. for product in products_with_total_sales:
16. print(product.name, product.total_sales)
17. **annotate** กับ **Avg**
18. python
19. Copy code
20. from django.db.models import Avg
21.
22. products_with_avg_rating = Product.objects.annotate(avg_rating=Avg('reviews__rating'))
23. for product in products_with_avg_rating:
24. print(product.name, product.avg_rating)

การใช้ **aggregate()** และ **annotate()** นั้นช่วยให้เราสามารถคำนวณข้อมูลและเพิ่มข้อมูลที่คำนวณไว้ไปใน QuerySet ได้อย่างมีประสิทธิภาพมากขึ้นใน Django ORM