Git

https://github.com/it-web-pro

git clone --depth 1 --no-checkout https://github.com/Pleng-np221/WP
git sparse-checkout init --cone
git sparse-checkout set mysite myshop myblogs note
git checkout

SetUp

Install virtualenv
pip install virtualenv

Create a virtual environment
py -m venv myvenv

Activate virtual environment
myvenv\Scripts\activate.bat

Install Django
pip install django

---

pip install psycopg2
OR
pip install psycopg2-binary

#เปิด shell postgress
psql -U postgres

---

pip install django-extensions ipython jupyter notebook
pip install ipython==8.25.0 jupyter_server==2.14.1 jupyterlab==4.2.2 jupyterlab_server==2.27.2
pip install notebook==6.5.7

```
python manage.py shell_plus --notebook

import os
os.environ["DJANGO_ALLOW_ASYNC_UNSAFE"] = "true"
from blogs.models import Blog
```

6.1 ไปที่ folder ที่มีไฟล์ pgAdmin4.exe

6.2 Copy path ที่อยู่ของไฟล์ psql.exe เพื่อนำไป add ใน path

6.3 เพิ่ม path to psql ใน Windows environment variables
C:\Program Files\PostgreSQL\17\pgAdmin 4\runtime

HINT: ไปสร้าง DB ใน postgres ก่อนนะครับ สร้าง DB ชื่อ "blogs"

```
#projectname/settings.py
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django_extensions",
    "blogs",
]

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

Create project "myblogs"
django-admin startproject myblogs

cd myblogs

Create the "blogs" app
python manage.py startapp blogs

Start server
python manage.py runserver
โดย port default จะเป็น port 8000 แต่ถ้าคุณต้องการเปลี่ยน port สามารถใช้ command
python manage.py runserver 8080
python manage.py makemigrations blogs
python manage.py migrate
python manage.py shell
python manage.py shell_plus --notebook

Shell

Model fields
https://docs.djangoproject.com/en/5.0/ref/models/fields/

```
from datetime import datetime
datenow = datetime.now().date()
print(datenow)
```

```
b = Blog(name="Beatles Blog", tagline="All the latest Beatles news.")
b.save()
#create then save immediately
cheese_blog = Blog.objects.create(name="Cheddar Talk", tagline="Greate
cheese!")

b2 = Blog.objects.create(name="Beatles Blog", tagline="All the latest
Beatles news.")
entry2 = Entry.objects.create(blog=b2, headline="Test entry",
body_text="Bla bla bla", pub_date=date(2010, 1, 1))
```

```
#F("name") ใช้เปรียบเทียบ
products = Product.objects.filter(
    Q(name__icontains="Smart") | Q(name__icontains="Electric")
).exclude(
    Q(description__icontains="compact") |
Q(description__icontains="portable")
).filter(description__icontains=F("name"))
for p in products:
    print(f"PRODUCT ID: {p.id}, NAME: {p.name}, DESCRIPTION:
{p.description}")

for o in Order.objects.filter(orderitem__product__name="Dog Bed"):
#     c = Customer.objects.get(id = o.customer_id)
    print(f"ORDER ID:{o.id}, DATE: {o.order_date}, CUSTOMER:
{o.customer.first_name} {o.customer.last_name}, REMARK: {o.remark}")
```

Model

```
#ex
from django.db import models

class Customer(models.Model):
    first_name = models.CharField(max_length=150)
    last_name = models.CharField(max_length=200)
    email = models.CharField(max_length=150)
    address = models.JSONField(null=True)

class ProductCategory(models.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)

class CartItem(models.Model):
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    amount = models.PositiveIntegerField(default=1)

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    order_date = models.DateField()
    remark = models.TextField(null=True, blank=True)

class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    amount = models.PositiveIntegerField(default=1)

class Payment(models.Model):
    order = models.OneToOneField(Order, on_delete=models.PROTECT)
    payment_date = models.DateField()
    remark = models.TextField(null=True, blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    discount = models.DecimalField(max_digits=10, decimal_places=2,
default=0)

class PaymentItem(models.Model):
    payment = models.ForeignKey(Payment, on_delete=models.CASCADE)
    order_item = models.OneToOneField(OrderItem, on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    discount = models.DecimalField(max_digits=10, decimal_places=2,
default=0)

class PaymentMethod(models.Model):
    class MethodChoices(models.Choices):
        QR = "QR"
        CREDIT = "CREDIT"

    payment = models.ForeignKey(Payment, on_delete=models.CASCADE)
```

```
    method = models.CharField(max_length=15,
choices=MethodChoices.choices)
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

Fields types

- BooleanField(**options)
- CharField(max_length=None, **options)
- EmailField(max_length=254, **options)
- URLField(max_length=200, **options)
- UUIDField(**options)
- TextField(**options)
- DateField(auto_now=False, auto_now_add=False, **options)
    - auto_now = True คือจะบันทึกค่า datetime.now() ทุกครั้งที่มีการแก้ไขค่า (INSERT + UPDATE)
    - auto_now_add = True คือจะบันทึกค่า datetime.now() ตอนที่สร้างใหม่ (INSERT)
- DateTimeField(auto_now=False, auto_now_add=False, **options)
- TimeField(auto_now=False, auto_now_add=False, **options)
- FileField(upload_to='', storage=None, max_length=100, **options)
    - upload_to คือกำหนด path ที่จะ save file

```
class MyModel(models.Model):
    # file will be uploaded to MEDIA_ROOT/uploads
    upload = models.FileField(upload_to="uploads/")
    # or...
    # file will be saved to MEDIA_ROOT/uploads/2015/01/30
    upload = models.FileField(upload_to="uploads/%Y/%m/%d/")
```

- ImageField(upload_to=None, height_field=None, width_field=None, max_length=100, **options)
    - สืบทอด attributes และ methods ทั้งหมดจาก FileField
    - ทำการ validate ให้ว่าเป็น object ของ image ที่เหมาะสม และ สามารถกำหนด height_field และ width_field
- DecimalField(max_digits=None, decimal_places=None, **options)

```
models.DecimalField(max_digits=5, decimal_places=2)
```

- IntegerField(**options)
    - ค่าตั้งแต่ -2147483648 ถึง 2147483647 รองรับใน database ทุกตัวที่ supported โดย Django.
- PositiveIntegerField(**options)
- JSONField(encoder=None, decoder=None, **options)

Field options

- primary_key: ถ้ามีค่าเป็น True คือ column นี้เป็น primary key ของ table (ถ้าไม่กำหนด Django จะสร้าง column ชื่อ id ให้อัตโนมัติเป็น primary key)
- unique: ถ้ามีค่าเป็น True คือ ค่าใน column นี้ห้ามซ้ำ
- null: ถ้ามีค่าเป็น True คือ column นี้มีค่าเป็น null ได้
- blank: ถ้ามีค่าเป็น True คือ column นี้มีค่าเป็น "" หรือ empty string ได้

```
from django.db import models

class Student(models.Model):
    code = models.CharField(max_length=20, primary_key=True)
    full_name = models.CharField(max_length=200, null=False, blank=False,
unique=True)
```

- default: กำหนดค่า default
- choices: กำหนด ENUM ให้เลือกเฉพาะค่าที่กำหนด

```
from django.db import models
from django.utils.translation import gettext_lazy as _

class Student(models.Model):
    class YearInSchool(models.TextChoices):
        FRESHMAN = "FR", _("Freshman")
        SOPHOMORE = "SO", _("Sophomore")
        JUNIOR = "JR", _("Junior")
        SENIOR = "SR", _("Senior")
        GRADUATE = "GR", _("Graduate")
```

```
    year_in_school = models.CharField(
        max_length=2,
        choices=YearInSchool,
        default=YearInSchool.FRESHMAN,
    )

    def is_upperclass(self):
        return self.year_in_school in {
            self.YearInSchool.JUNIOR,
            self.YearInSchool.SENIOR,
        }
```

- db_index: ถ้ามีค่าเป็น True คือจะสร้าง index ใน database สำหรับ column นี้

Week2

```
#polls/views.py
from datetime import datetime
from django.shortcuts import render
from django.http import HttpResponse
from .models import Question, Choice

def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    context = {
        "page_title" : "Latest 5 questions",
        "questions": latest_question_list,
        "current_date" : datetime.now().strftime("%d/%m/%Y, %H:%M:%S")
        }
    return render(request, "index.html", context)

def detail(request, question_id):
    question = Question.objects.get(pk=question_id)
    choices = Choice.objects.filter(question_id=question)
    context = {
        "question": question,
        "choices" : choices
        }
    return render(request, "detail.html", context)
```

```python
def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)


def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```

```python
# polls/urls.py
from django.urls import path

from . import views

urlpatterns = [
    # แมพจากบนลงล่าง
    # ex: /polls/
    path("", views.index, name="index"),
    # ex: /polls/5/
    path("<int:question_id>/", views.detail, name="detail"),
    # ex: /polls/5/results/
    path("<int:question_id>/results/", views.results, name="results"),
    # ex: /polls/5/vote/
    path("<int:question_id>/vote/", views.vote, name="vote"),
]
```

```python
# mysite/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path("polls/", include("polls.urls")),
    path("admin/", admin.site.urls),
]
```

```python
# polls/templates/detail.html

<html>
    <head>
    </head>
```

```
    <body>
        <h1>Q{{ question.id }} : {{ question.question_text }}</h1>
        <h2>{{ question.pub_date }}</h2>
        <h3>Description: {{ question.description }}</h3>
        {% if choices %}
            <ul>
            {% for choice in choices %}
                <li>{{ choice.choice_text }} ( {{choice.votes}} )</li>
            {% endfor %}
            </ul>
        {% else %}
            <p>No polls are available.</p>
        {% endif %}
    </body>
</html>
```

```
# polls/templates/index.html

<html>
    <head>
    </head>
    <body>
        <h1>{{page_title}}</h1>
        <h2>{{current_date}}</h2>
        {% if questions %}
            <ul>
            {% for question in questions %}
                <li><a href="{% url 'detail' question.id %}">{{
question.question_text }}</a></li>
            {% endfor %}
            </ul>
        {% else %}
            <p>No polls are available.</p>
        {% endif %}
    </body>
</html>
```

Week3

```
#week3
import shop.models
c1 = Customer(first_name="Django", last_name="Reinhardt",
email="dj_rein@mail.com", address="{Liberchies, Pont-à-Celles, Belgium}")
c1.save()

c1.first_name = "Darwin"
c1.last_name = "Nunez"
c1.email = "660xxxxx@kmitl.ac.th"
c1.save()

p1 = Product(name="USB-C Charger", description="20W fast charging USB-C
adapter, compact and efficient.", remaining_amount = 100, price=299.50)
p2 = Product(name="Noise Cancelling Earbuds", description="Wireless
earbuds with advanced noise cancelling technology.", remaining_amount =
50, price=1890.00)
p3 = Product(name="Mechanical Keyboard", description="RGB mechanical
keyboard designed for gamers with tactile feedback.", remaining_amount =
25, price=2499.99)
p1.save()
p2.save()
p3.save()
Product.objects.filter(price__gt=500)

o1 = Order(customer=c1, remark="This is order for Darwin Nunez")
o1.save()
oi1 = OrderItem(order=o1, product=p3, amount = 1)
oi2 = OrderItem(order=o1, product=p1, amount = 2)
oi1.save()
oi2.save()

Product.objects.filter(name__icontains="cha")


____

from datetime import datetime, timedelta
from django.utils import timezone
from zoneinfo import ZoneInfo
```

```
>>> datetime.now()
datetime.datetime(2025, 7, 16, 17, 16, 17, 933738)
>>> dn = datetime.now()
>>> df = dn + timedelta(days=500)
>>> print(df)
2026-11-28 17:17:01.614086
>>> print(dn)
2025-07-16 17:17:01.614086
>>> df.date()
datetime.date(2026, 11, 28)
>>> df.weekday()
5
>>> df.strftime("%A")
'Saturday'
```

Week4

```
-- Entry.objects.filter(headline__contains='Lennon')
SELECT ... WHERE headline LIKE '%Lennon%';

-- Entry.objects.filter(headline__icontains='Lennon')
SELECT ... WHERE headline ILIKE '%Lennon%';

-- Entry.objects.filter(headline__in=('a', 'b', 'c'))
SELECT ... WHERE headline IN ('a', 'b', 'c');

-- Entry.objects.filter(pub_date__range=(start_date, end_date))
SELECT ... WHERE pub_date BETWEEN '2005-01-01' AND '2005-03-31';

-- Entry.objects.filter(pub_date__year=2005)
-- Entry.objects.filter(pub_date__year__gte=2005)
SELECT ... WHERE pub_date BETWEEN '2005-01-01' AND '2005-12-31';
SELECT ... WHERE pub_date >= '2005-01-01';

-- Entry.objects.filter(pub_date__isnull=True)
SELECT ... WHERE pub_date IS NULL;

-- Entry.objects.get(title__regex=r"^(An?|The) +")
SELECT ... WHERE title ~ '^(An?|The) +'; -- PostgreSQL
```

HINT: ถ้าอยากลองพิมพ์ SQL query ออกมาดูสามารถทำได้โดยใช้ `.query`

```
print(q.query)
```

```
#lab
c1 = ProductCategory.objects.get(name="Books and Media")
p1 = Product.objects.create(name="Philosopher's Stone (1997)",
remaining_amount=20, description = "By J. K. Rowling.", price = 790)
p2 = Product.objects.create(name="Me Before You", remaining_amount=40,
description = "A romance novel written by Jojo", price = 390)
c2 = ProductCategory.objects.get(name="Information Technology")
c3 = ProductCategory.objects.get(name="Electronics")
p3 = Product.objects.create(name="Notebook HP Pavilion Silver",
remaining_amount=10, description="Display Screen. 16.0", price = 20000)

p1.categories.add(c1)
p2.categories.add(c1)
p3.categories.add(c2, c3)

p1.save()
p2.save()
p3.save()

p1.name = "Half-Blood Prince (2005)"
p1.save()

Product.objects.filter(categories__name="Books").delete()

for p in Product.objects.filter(price__lt=200, price__gt=100):
    print(f"PRODUCT ID: {p.id}, NAME: {p.name}, PRICE: {p.price}")
```

week5

```
# import modules
from companies.models import *
```

```
from django.db.models import Count, F, Value
from django.db.models.functions import Length, Upper
```

```
from django.db.models.lookups import GreaterThan
Company.objects.filter(num_employees__gt=F("num_chairs") * 2)
company = (Company.objects.filter(num_employees__gt=F("num_chairs"))
     .annotate(chairs_needed=F("num_employees") - F("num_chairs"))
     .first()
)
Company.num_employees

# find greatest
from django.db.models.functions import Greatest
blog = Blog.objects.create(body="Greatest is the best.")
comment = Comment.objects.create(body="No, Least is better.", blog=blog)
comments = Comment.objects.annotate(last_updated=Greatest("modified",
"blog__modified"))

# find diff + filter + order
for p in (Payment.objects.annotate(after_discount_price= F("price") -
F("discount")).filter(after_discount_price__gt =
500000).order_by("-after_discount_price")):
    print(f"ID: {p.id}, PRICE: {p.price}, DISCOUNT {p.discount},
AFTER_DISCOUNT {p.after_discount_price}")

# concat
from django.db.models import CharField, Value as V
from django.db.models.functions import Concat
import json
for c in (Customer.objects.annotate(full_name = Concat("first_name", V("
"), "last_name", output_field=CharField())).values("id", "email",
"address", "full_name").order_by("full_name")[:5]):
    print(json.dumps(c, indent=4, sort_keys=False))

from django.db.models import Avg, Count, Sum
print(Product.objects.filter(remaining_amount__gt = 0).aggregate(avg =
Avg("price")))

# filter month -> __month
import datetime
print(CartItem.objects.filter(cart__create_date__month=5).aggregate(sum =
Sum("product__price")))
```

```
# OR, find num in filter
from django.db.models import Q
pc = ProductCategory.objects.filter(Q(name="Electronics") |
Q(name="Jewelry")).annotate(count =
    Count("product", filter = (Q(product__price__gte = 8000) &
Q(product__price__lte = 50000))))
for i in pc:
    print(f"PRODUCT CATEGORY NAME: {i.name}, PRODUCT COUNT: {i.count}")

# find latest order by each customer + arranged by date
from django.db.models import OuterRef, Subquery
latest = Order.objects.filter(customer =
OuterRef('pk')).order_by('-order_date')
ctm = Customer.objects.annotate(orderdate =
Subquery(latest.values("order_date")[:1])).order_by("orderdate")
for i in ctm:
    print(f"CUSTOMER NAME: {i.first_name} {i.last_name}, ORDER DATE:
{i.orderdate}")

# filter and
for p in (Product.objects.filter(categories__name = "Information
Technology"
                            ).filter(categories__name = "Electronics"
                            ).order_by("id")
        ):
    cset = p.categories.all()
    print(f"PRODUCT ID: {p.id}, PRODUCT NAME: {p.name}, PRODUCT CATEGORY:
", end="")
    for c in cset:
        print(f"{c.name}, ", end="")
    print("")


#ex1
for o in OrderItem.objects.filter(order__customer__first_name="Sek",
order__customer__last_name="Loso"):
    print(f"CUSTOMER: {o.order.customer.first_name}, ORDER ID:
{o.order.id}, PRICE: {o.order.payment.price}, DISCOUNT:
{o.order.payment.discount}, PRODUCT: {o.product.name}")
```

```python
#ex2
pc_e = ProductCategory.objects.get(name="Electronics")
pc_e.name = "Electronics and Toys"
pc_e.save()

pc_tg = ProductCategory.objects.get(name="Toys and Games")

for p in (Product.objects.filter(categories__name="Toys and Games")):
    p.categories.add(pc_e)
    p.categories.remove(pc_tg)

pc_tg.delete()

print(Product.objects.filter(categories=pc_e).count())
```

JSON EX:


```
{
    "id": 17,
    "email": "anantaya.deena@gmail.com",
    "address": {
        "district": "Yan Nawa",
        "location": "60 Thanon Chan Road",
        "province": "Bangkok",
        "postal_code": 10120
    },
    "full_name": "Anantaya Tontong"
}
```