Kod:

all_requests = AssistanceRequest.objects.all()

total_count = len(all_requests)

1. Tüm kayıtları çekip Python'da saymak

- Bütün tabloyu RAM'e alıyoruz, sonra len() diyoruz.
- Data büyüdükçe hem gereksiz IO hem de bellek tüketimi yaratır.
- Bunun işi DB'nin:

total_count = AssistanceRequest.objects.count()

---

providers = Provider.objects.all()

for p in providers:

if p.is_active and p.last_ping > datetime.now() - timedelta(minutes=5):

active_providers.append(p)

2. Filtreyi DB yerine Python'da yapmak + datetime.now() sıkıntısı

- Provider.objects.all() ile herkesi çekip for içinde filtrelemek pahalı.
  Direkt DB'ye şartı verip tek sorguda halletmek lazım:

  from django.utils import timezone

  cutoff = timezone.now() - timedelta(minutes=5)

  active_count = Provider.objects.filter(

  is_active=True,

  last_ping__gt=cutoff

).count()

- Bir de Django projelerinde genelde USE_TZ = True.
  last_ping timezone-aware, datetime.now() naive → ya hata alırız ya da yanlış karşılaştırma olur.
  Burada timezone.now() kullanmak gerekiyor.

logs = Log.objects.filter(level='ERROR')[:5]

log_messages = [l.message for l in logs]

3. Logları ham haliyle döndürmek + “son log” garantisi olmaması

- Güvenlik tarafı: ERROR log’un içinde stack trace, internal URL, bazen hassas veri olabilir.
  Bunları olduğu gibi API’den dökmek güzel fikir değil. En azından özetlemek lazım.
- Mantık tarafı: “son loglar” diyoruz ama order_by yok.
  [:5] şu an “rastgele ilk 5 kayıt” gibi davranabilir.

Beklenen daha çok şöyle olur:

logs = Log.objects.filter(level='ERROR').order_by('-created_at')[:5]

Kısaca 3 kritik nokta:

- count() yerine tabloyu komple çekmek,
- Provider sayımını Python’da yapıp üstüne datetime.now() kullanmak,
- Hata loglarını sıralamadan ve filtrelemeden direkt dışarıya açmak.