**نحوه کش کردن با Redis در پروژه Django :**

**۱. نصب و تنظیمات Redis**

اول Redis رو نصب می‌کنیم (روی لینوکس/هاست):

sudo apt install redis-server

سپس توی settings.py جنگو:  
  


**۲. کش کردن ویوهای Class-Based**

جنگو یه دکوریتور داره به اسم cache\_page که میشه روی CBV هم اعمال کرد.

مثال: ✅ این یعنی کل خروجی HTML این ویو برای ۱۵ دقیقه توی Redis ذخیره میشه.



**۳. کش کردن APIهای DRF**

در DRF معمولاً داده JSON برمی‌گرده، پس بهتره روی **queryset** یا روی **response** کش بذاریم.

**روش A – کش کردن کل response API**



**روش B – کش دستی روی داده (دقیق‌تر)**

بعضی وقتا نمی‌خوای کل response کش بشه، می‌خوای فقط داده‌های query سنگین کش بشن.



**۴. Invalidate کردن کش (پاک کردن بعد از تغییر دیتا)**

وقتی دیتای حساس مثلاً Product تغییر کنه، باید کش پاک بشه.  
مثال در سیگنال:



**۵. نکته‌های حرفه‌ای**

* برای APIهای بزرگ بهتره **django-cacheops** یا **django-redis** استفاده بشه که کش QuerySet رو اتوماتیک هندل می‌کنن.
* TTL (زمان انقضا) رو بر اساس نوع دیتا انتخاب کن: داده‌های استاتیک (لیست دسته‌بندی‌ها) → طولانی، داده‌های پویا (سبد خرید) → کوتاه.
* حتماً کش keyها رو namespace کن (مثلاً f"products:{id}" ) تا تداخل پیش نیاد.
* برای performance بالاتر میشه از @cache\_page + Vary headers (مثلاً برای کاربرهای مختلف) استفاده کرد.

**کاربرد و موارد استفاده از caching ؟**

**Caching** یعنی ذخیره‌ی نتایج سنگین ( query ، محاسبه، یا response )در یک حافظه‌ی سریع مثل RAM تا دفعه‌های بعدی به جای محاسبه/خواندن دوباره، مستقیم از کش جواب بدیم.

🔹 هدف:

* افزایش **سرعت پاسخ‌دهی**
* کاهش **بار روی دیتابیس/سرور**
* صرفه‌جویی در **منابع سیستم**

**📌 چرا Redis برای کش؟**

* Redis یه دیتابیس in-memory (توی RAM) هست → سرعتش بسیار بالاست.
* قابلیت TTL (زمان انقضا)داره → کش خودکار بعد از مدتی پاک میشه.
* از کلید-مقدار استفاده می‌کنه → سریع‌ترین حالت بازیابی داده.
* می‌تونه بین چند سرور (distributed cache) استفاده بشه → مناسب برای scale بالا.

**📌 چه زمان‌هایی باید کش کنیم؟**

1. **داده‌های تکراری و پرمصرف**
   * مثال: لیست محصولات در صفحه اصلی فروشگاه.
   * Queryهای سنگینی که کاربرهای زیادی مدام تکرار می‌کنن.
2. **داده‌هایی که به ندرت تغییر می‌کنن**
   * مثال: لیست دسته‌بندی‌ها، منوهای سایت، تنظیمات عمومی.
3. **APIهای سنگین یا کند**
   * مثال: خروجی گزارش‌های آماری، یا API که به سرویس خارجی وصل میشه.
   * مثل همون مثالی که زدی( API که delay داره).
4. **زمانی که نیاز به performance بالا داریم**
   * برای endpointهایی که باید سریع جواب بدن (مثلاً سبد خرید یا جستجو).

**📌 چه زمان‌هایی نباید کش کنیم؟**

* داده‌هایی که **خیلی سریع تغییر می‌کنن** (مثلاً موجودی لحظه‌ای انبار).
* داده‌های **کاملاً شخصی‌سازی‌شده** برای هر کاربر (مگر اینکه per-user cache بزنی)
* زمانی که invalidate کردن کش (پاک‌کردنش بعد از تغییر) خیلی پیچیده و پرریسک میشه.

**📌 انواع caching در Django با Redis**

1. **Page Cache** → کل response ویو رو کش می‌کنه (@cache\_page).
2. **Template Fragment Cache** → فقط بخشی از HTML رو کش می‌کنه.
3. **Low-level Cache API** → خودت با cache.get/set کنترل می‌کنی.
4. **QuerySet Cache مثل (django-cacheops** **)** → نتایج ORM رو کش می‌کنه.

✅ پس کش یه ابزار کمکیه برای **performance**، نه جایگزین دیتابیس.  
قاعده کلی:

* اگه **چیزی گرونه (محاسبه/Query/Call خارجی)** و **تغییراتش کم** → کش کن.
* اگه **چیزی همیشه تغییر می‌کنه** یا **باید همیشه real-time باشه** → کش نکن.