In []:
```python
import redis

r = redis.Redis(host="localhost", port=6379, decode_responses=True)
```

Étude de cas 1

In []:
```python
from typing import Any


def update_user_profile(user_id: int, preferences: dict[str, Any]):
    r.hset(f"user:{user_id}", mapping=preferences)

def add_recent_view(user_id: int, product_id: int):
    r.lpush(f"recent_views:{user_id}", product_id)
    r.ltrim(f"recent_views:{user_id}", 0, 20)

def update_recommendations(user_id: int, product_id: int, score: int):
    r.zadd(f"recommendations:{user_id}", {str(product_id): score})

def get_top_recommendations(user_id: int, count: int = 5) -> set[int]:
    recs = r.zrevrange(f"recommendations:{user_id}", 0, count - 1)

    assert isinstance(recs, list)

    return set(int(i) for i in recs)

def cache_product_info(product_id: int, product_info: dict[str, Any]):
    r.hset(f"product:{product_id}", mapping=product_info)

async def get_product_info(product_id: int) -> dict[str, Any]:
    res = r.hgetall(f"product:{product_id}")

    if isinstance(res, dict):
        return res

    return await res
```

Étude de cas 2

In []:
```python
import time
from uuid import uuid4


def create_session(user_id: int, data: str) -> str:
    gen_sess_id = str(uuid4())

    r.hset(
        f"session:{gen_sess_id}",
        mapping={
            "user_id": user_id,
            "timestamp": int(time.time()),
            "data": data
        }
    )

    r.expire(f"session:{gen_sess_id}", 3600)

    return gen_sess_id

async def get_session(session_id: str) -> dict[str, Any] | None:
    session = r.hgetall(f"session:{session_id}")

    if not isinstance(session, dict):
        session = await session

    if len(session) > 0:
        r.expire(f"session:{session_id}", 3600)
        return session

def update_session(session_id: str, data):
    r.hset(f"session:{session_id}", mapping={"data": data})
    r.expire(f"session:{session_id}", 3600)

def delete_session(session_id: str):
    r.delete(f"session:{session_id}")
```

Étude de cas 3

In []:
```python
import datetime


def call_counter_basic(user_id: int) -> bool:
    current_hour = datetime.datetime.now().strftime("%Y%m%d%H")

    calls = r.get(f"rate_limit:{user_id}:{current_hour}")

    if calls is None:
        r.set(f"rate_limit:{user_id}:{current_hour}", 0)
        calls = 0
    else:
        calls = int(str(calls))

    r.incr(f"rate_limit:{user_id}:{current_hour}")

    return calls < 20

def call_counter_sliding(user_id: int) -> bool:
    calls = r.keys(f"rate_limit:{user_id}:*")
    assert isinstance(calls, list)

    if len(calls) < 10:
        now = datetime.datetime.now().isoformat()
        r.set(f"rate_limit:{user_id}:{now}", 0)
        r.expire(f"rate_limit:{user_id}:{now}", 60)
        return True
    return False
```