```
# 安裝 AI 模型需要的套件
!pip install -q transformers accelerate bitsandbytes pandas
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.1/59.1 MB 12.2 MB/s eta 0:00:00
```

```
# ==========================================
# 第一步：環境準備 & 下載資料 (Run once)
# ==========================================
import os

# 2. 上傳你的 kaggle.json 鑰匙
from google.colab import files
if not os.path.exists('kaggle.json'):
    print("請上傳 kaggle.json 檔案...")
    files.upload() # 這裡會跳出按鈕讓你選檔案

# 3. 設定權限並下載 eCommerce 資料集
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

# 下載 2019-Oct.csv (約 2GB，Kaggle API 下載速度很快)
if not os.path.exists('2019-Oct.csv'):
    print("🚀 正在從 Kaggle 下載資料集...")
    !kaggle datasets download -d mkechinov/ecommerce-behavior-data-from-multi-category-store
    !unzip -o ecommerce-behavior-data-from-multi-category-store.zip 2019-Oct.csv
    print("✅ 資料下載並解壓縮完成！")
else:
    print("✅ 資料已存在，跳過下載。")
```

```
請上傳 kaggle.json 檔案...
選擇檔案 未選擇任何檔案  Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.
Saving kaggle.json to kaggle.json
🚀 正在從 Kaggle 下載資料集...
Dataset URL: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store
License(s): copyright-authors
Downloading ecommerce-behavior-data-from-multi-category-store.zip to /content
100% 4.28G/4.29G [01:01<00:00, 34.8MB/s]
100% 4.29G/4.29G [01:01<00:00, 75.4MB/s]
Archive:  ecommerce-behavior-data-from-multi-category-store.zip
  inflating: 2019-Oct.csv
✅ 資料下載並解壓縮完成！
```

```
# ==========================================
# 第二步：Agent 核心程式碼 (Logic & LLM)
# ==========================================
import pandas as pd
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

# 1. 載入模型 (Agent) - 使用 Qwen2.5-7B
# ------------------------------------------------------
# model_id = "Qwen/Qwen2.5-7B-Instruct"
model_id = "Qwen/Qwen2.5-1.5B-Instruct"
print("🧠 正在載入 LLM 模型 (這會用到 GPU)...")
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    torch_dtype=torch.float16,
    load_in_4bit=True
)

# 2. 定義策略庫 (Strategy Bank)
# 這些內容來自你上傳的 Word 檔筆記
# ------------------------------------------------------
STRATEGY_BANK = {
    # 對應筆記：加入購物車後一直不結帳 (拖延)
    "cart_abandonment": {
        "insight": "現在偏誤 / 延遲折扣偏誤",
        "tactic": "創造焦慮感，給予下單壓力",
        "example": "你放在購物車裡的優惠只剩今天。逾時，連我們都幫不了你。"
    },
    # 對應筆記：多次瀏覽同商品卻不買 (決策疲乏)
    "repetitive_view": {
        "insight": "決策疲乏 / 選擇麻痺",
```

```
        "tactic": "提供社會認同，簡化選擇",
        "example": "你已經看了好多次，不如讓它成為你每天的輕鬆選擇。"
    },
    # 對應筆記：一直比較競品（價格敏感）
    "price_comparison": {
        "insight": "價格敏感 / 損失厭惡",
        "tactic": "主動提供比價，強調現在買最划算",
        "example": "現在下單比全年 95% 的價格都低，錯過等一年。"
    },
    # 對應筆記：只是逛逛（興趣探索）
    "window_shopping": {
        "insight": "探索階段 / 動機低",
        "tactic": "建立關係，提供首購優惠",
        "example": "清單裡的熱門款即將售完，你等的不是降價，是缺貨。"
    }
}
```

```
🧠 正在載入 LLM 模型（這會用到 GPU）...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/setti
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:        7.30k/? [00:00<00:00, 218kB/s]
vocab.json:        2.78M/? [00:00<00:00, 43.1MB/s]
merges.txt:        1.67M/? [00:00<00:00, 35.5MB/s]
tokenizer.json:        7.03M/? [00:00<00:00, 58.1MB/s]
config.json: 100%                                                660/660 [00:00<00:00, 50.6kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be removed in the future versions. P
model.safetensors: 100%                                          3.09G/3.09G [00:25<00:00, 196MB/s]
generation_config.json: 100%                                       242/242 [00:00<00:00, 20.6kB/s]
```

```
import pandas as pd

# ------------------------------------------
# 3. Agent 思考函式 (The Hook)
# ------------------------------------------
def detect_scenario(user_data: pd.Series) -> str:
    """根據單筆行為資料判斷情境"""
    scenario = "window_shopping"

    # 有 cart 沒 purchase = 購物車遺棄（你這裡是單筆資料，先用 event_type 近似）
    if user_data.get("event_type") == "cart":
        scenario = "cart_abandonment"
    # 重複看同一商品要累計才知道；這裡先用「看高價」當作比價情境
    elif user_data.get("event_type") == "view" and float(user_data.get("price", 0)) > 500:
        scenario = "price_comparison"

    return scenario


def run_agent(user_data: pd.Series) -> str:
    """
    輸入: Pandas Series (單一用戶數據)
    輸出: AI 生成的廣告
    """
    # --- A. 規則判斷 (感知層) ---
    scenario = detect_scenario(user_data)

    # strategy bank 防呆：若 scenario 不在字典就回退
    strategy = STRATEGY_BANK.get(scenario, STRATEGY_BANK["window_shopping"])

    # 欄位防呆（避免 NaN / 欄位不存在）
    category = user_data.get("category_code", "未知類別")
    brand = user_data.get("brand", "未知品牌")
    price = user_data.get("price", "")
    try:
        price_str = f"${float(price):.0f}"
    except Exception:
        price_str = f"${price}" if price != "" else "未標價"

    # --- B. 組合 Prompt (決策層) ---
```

```
    prompt = f"""
    你是一個專業的電商行銷 Agent。請根據【用戶行為】與【策略指引】，寫一則繁體中文短訊廣告。

    【用戶行為】
    - 關注商品: {category} (品牌: {brand})
    - 價格: {price_str}
    - 偵測狀態: {scenario}

    【策略指引】
    - 心理洞察: {strategy['insight']}
    - 策略重點: {strategy['tactic']}
    - 參考文案: "{strategy['example']}"

    請生成一則 50 字以內的吸睛文案，語氣自然，不要完全照抄參考文案。
    """.strip()

    # --- C. LLM 生成 (執行層) ---
    messages = [{"role": "user", "content": prompt}]
    text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)

    # pad_token 保險
    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    inputs = tokenizer([text], return_tensors="pt", padding=True).to(model.device)

    generated_ids = model.generate(
        input_ids=inputs.input_ids,
        attention_mask=inputs.attention_mask,
        max_new_tokens=150,
        temperature=0.7,
        do_sample=True,
        pad_token_id=tokenizer.eos_token_id,
    )

    # 只取「新生成」的部分，避免把 prompt 一起 decode
    input_len = inputs.input_ids.shape[1]
    new_tokens = generated_ids[0][input_len:]
    response = tokenizer.decode(new_tokens, skip_special_tokens=True).strip()

    # 不要用 split("assistant")，因為模型不一定吐出這個字
    final_ad = response
    return final_ad


# ==========================================
# 第三步：開始執行 (Run Agent)
# ==========================================

print("📂 正在讀取 CSV 資料...")
df = pd.read_csv("2019-Oct.csv", nrows=1000)

# 清洗：去掉缺 brand/category_code 的資料
df_clean = df.dropna(subset=["category_code", "brand"]).reset_index(drop=True)

print(f"🤖 Agent 啟動！正在分析 {len(df_clean)} 筆行為數據...\n")

sample_indices = [10, 50, 100]
for idx in sample_indices:
    if idx >= len(df_clean):
        continue

    user_row = df_clean.iloc[idx]

    # 先用同一套規則推 scenario + strategy（因為 df 裡本來沒有那些欄位）
    scenario = detect_scenario(user_row)
    strategy = STRATEGY_BANK.get(scenario, STRATEGY_BANK["window_shopping"])

    print(f"👤 用戶 ID: {user_row.get('user_id', 'N/A')}")
    print(f"🛒 行為: {user_row.get('event_type', 'N/A')} | 商品: {user_row.get('brand', 'N/A')} - {user_row.
    print(f"📦 價格: ${user_row.get('price', 'N/A')}")
    print(f"🕵️ 偵測場景: {scenario}")
    print(f"🧠 心理原理: {strategy['insight']}")

    ad_copy = run_agent(user_row)

    print(f"📢 Agent 生成廣告:\n{ad_copy}")
    print("-" * 50)
```

```
📂 正在讀取 CSV 資料...
🤖 Agent 啟動！正在分析 598 筆行為數據...

👤 用戶 ID: 537192226
🛒 行為: view | 商品: haier - electronics.video.tv
📦 價格: $193.03
🕵️ 偵測場景: window_shopping
🧠 心理原理: 探索階段 / 動機低
🦾 Agent 生成廣告:
消費者們！在這個動力十足的消費季中，Haier的電子產品正在緊俏之中，僅剩最後一批，等待您的是前所未有的價格優惠和無比期待。立即加入我
--------------------------------------------------
👤 用戶 ID: 550050854
🛒 行為: view | 商品: hp - computers.notebook
📦 價格: $1512.78
🕵️ 偵測場景: price_comparison
🧠 心理原理: 價格敏感 / 損失厭惡
🦾 Agent 生成廣告:
消費者們注意！hp-notebook電腦，現正以驚喜優惠價$1513，是時候行動了！錯過就再無機遇。立即下單，享受超值購物體驗，並能享受到比全年
--------------------------------------------------
👤 用戶 ID: 552783882
🛒 行為: view | 商品: samsung - electronics.smartphone
📦 價格: $254.82
🕵️ 偵測場景: window_shopping
🧠 心理原理: 探索階段 / 動機低
🦾 Agent 生成廣告:
"想擁有最新款的SAMSUNG手機嗎? 別錯過這個難得的機遇！在「Samsung Electronics Smart Phone」中，我們為您精心選擇了「SAMSUNG
--------------------------------------------------
```

```
# ==========================================
# 第二步：Agent 核心程式碼 (Logic & LLM)
# ==========================================
import pandas as pd
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

# 1. 載入模型 (Agent) - 使用 Qwen2.5-7B
# ------------------------------------------------------
model_id = "Qwen/Qwen2.5-7B-Instruct"
# model_id = "Qwen/Qwen2.5-1.5B-Instruct"
print("🧠 正在載入 LLM 模型 (這會用到 GPU)...")
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    torch_dtype=torch.float16,
    load_in_4bit=True
)

# 2. 定義策略庫 (Strategy Bank)
# 這些內容來自你上傳的 Word 檔筆記
# ------------------------------------------------------
STRATEGY_BANK = {
    # 對應筆記：加入購物車後一直不結帳 (拖延)
    "cart_abandonment": {
        "insight": "現在偏誤 / 延遲折扣偏誤",
        "tactic": "創造焦慮感，給予下單壓力",
        "example": "你放在購物車裡的優惠只剩今天。逾時，連我們都幫不了你。"
    },
    # 對應筆記：多次瀏覽同商品卻不買 (決策疲乏)
    "repetitive_view": {
        "insight": "決策疲乏 / 選擇麻痺",
        "tactic": "提供社會認同，簡化選擇",
        "example": "你已經看了好多次，不如讓它成為你每天的輕鬆選擇。"
    },
    # 對應筆記：一直比較競品 (價格敏感)
    "price_comparison": {
        "insight": "價格敏感 / 損失厭惡",
        "tactic": "主動提供比價，強調現在買最划算",
        "example": "現在下單比全年 95% 的價格都低，錯過等一年。"
    },
    # 對應筆記：只是逛逛 (興趣探索)
    "window_shopping": {
        "insight": "探索階段 / 動機低",
        "tactic": "建立關係，提供首購優惠",
        "example": "清單裡的熱門款即將售完，你等的不是降價，是缺貨。"
    }
}
```

```
🧠 正在載入 LLM 模型 (這會用到 GPU)...
tokenizer_config.json:        7.30k/? [00:00<00:00, 292kB/s]
vocab.json:        2.78M/? [00:00<00:00, 58.1MB/s]
merges.txt:        1.67M/? [00:00<00:00, 55.8MB/s]
tokenizer.json:        7.03M/? [00:00<00:00, 90.5MB/s]
config.json: 100%                                                  663/663 [00:00<00:00, 61.0kB/s]
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be removed in the future versions. P
model.safetensors.index.json:        27.8k/? [00:00<00:00, 2.06MB/s]
Fetching 4 files: 100%                                             4/4 [05:48<00:00, 348.46s/it]
model-00001-of-00004.safetensors: 100%                                    3.95G/3.95G [05:48<00:00, 59.4MB/s]
model-00004-of-00004.safetensors: 100%                                    3.56G/3.56G [04:32<00:00, 6.40MB/s]
model-00002-of-00004.safetensors: 100%                                    3.86G/3.86G [04:01<00:00, 16.3MB/s]
model-00003-of-00004.safetensors: 100%                                    3.86G/3.86G [02:52<00:00, 9.25MB/s]
Loading checkpoint shards: 100%                                        4/4 [02:05<00:00, 30.59s/it]
generation_config.json: 100%                                       243/243 [00:00<00:00, 19.9kB/s]
```

```python
import pandas as pd

# ------------------------------------------
# 3. Agent 思考函式 (The Hook)
# ------------------------------------------
def detect_scenario(user_data: pd.Series) -> str:
    """根據單筆行為資料判斷情境"""
    scenario = "window_shopping"

    # 有 cart 沒 purchase = 購物車遺棄（你這裡是單筆資料，先用 event_type 近似）
    if user_data.get("event_type") == "cart":
        scenario = "cart_abandonment"
    # 重複看同一商品要累計才知道；這裡先用「看高價」當作比價情境
    elif user_data.get("event_type") == "view" and float(user_data.get("price", 0)) > 500:
        scenario = "price_comparison"

    return scenario


def run_agent(user_data: pd.Series) -> str:
    """
    輸入: Pandas Series (單一用戶數據)
    輸出: AI 生成的廣告
    """
    # --- A. 規則判斷 (感知層) ---
    scenario = detect_scenario(user_data)

    # strategy bank 防呆：若 scenario 不在字典就回退
    strategy = STRATEGY_BANK.get(scenario, STRATEGY_BANK["window_shopping"])

    # 欄位防呆（避免 NaN / 欄位不存在）
    category = user_data.get("category_code", "未知類別")
    brand = user_data.get("brand", "未知品牌")
    price = user_data.get("price", "")
    try:
        price_str = f"${float(price):.0f}"
    except Exception:
        price_str = f"${price}" if price != "" else "未標價"

    # --- B. 組合 Prompt (決策層) ---
    prompt = f"""
你是一個專業的電商行銷 Agent。請根據【用戶行為】與【策略指引】，寫一則繁體中文短訊廣告。

【用戶行為】
- 關注商品: {category} (品牌: {brand})
- 價格: {price_str}
- 偵測狀態: {scenario}

【策略指引】
- 心理洞察: {strategy['insight']}
- 策略重點: {strategy['tactic']}
- 參考文案: "{strategy['example']}"
```

```
請生成一則 50 字以內的吸睛文案，語氣自然，不要完全照抄參考文案。
""".strip()

    # --- C. LLM 生成 (執行層) ---
    messages = [{"role": "user", "content": prompt}]
    text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)

    # pad_token 保險
    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    inputs = tokenizer([text], return_tensors="pt", padding=True).to(model.device)

    generated_ids = model.generate(
        input_ids=inputs.input_ids,
        attention_mask=inputs.attention_mask,
        max_new_tokens=150,
        temperature=0.7,
        do_sample=True,
        pad_token_id=tokenizer.eos_token_id,
    )

    # 只取「新生成」的部分，避免把 prompt 一起 decode
    input_len = inputs.input_ids.shape[1]
    new_tokens = generated_ids[0][input_len:]
    response = tokenizer.decode(new_tokens, skip_special_tokens=True).strip()

    # 不要用 split("assistant")，因為模型不一定吐出這個字
    final_ad = response
    return final_ad


# ========================================
# 第三步：開始執行 (Run Agent)
# ========================================

print("📂 正在讀取 CSV 資料...")
df = pd.read_csv("2019-Oct.csv", nrows=1000)

# 清洗：去掉缺 brand/category_code 的資料
df_clean = df.dropna(subset=["category_code", "brand"]).reset_index(drop=True)

print(f"🤖 Agent 啟動！正在分析 {len(df_clean)} 筆行為數據...\n")

sample_indices = [10, 50, 100]
for idx in sample_indices:
    if idx >= len(df_clean):
        continue

    user_row = df_clean.iloc[idx]

    # 先用同一套規則推 scenario + strategy（因為 df 裡本來沒有那些欄位）
    scenario = detect_scenario(user_row)
    strategy = STRATEGY_BANK.get(scenario, STRATEGY_BANK["window_shopping"])

    print(f"👤 用戶 ID: {user_row.get('user_id', 'N/A')}")
    print(f"🛒 行為: {user_row.get('event_type', 'N/A')} | 商品: {user_row.get('brand', 'N/A')} - {user_row.
    print(f"📦 價格: ${user_row.get('price', 'N/A')}")
    print(f"🕵️ 偵測場景: {scenario}")
    print(f"🧠 心理原理: {strategy['insight']}")

    ad_copy = run_agent(user_row)

    print(f"📢 Agent 生成廣告:\n{ad_copy}")
    print("-" * 50)
```

```
📂 正在讀取 CSV 資料...
🤖 Agent 啟動！正在分析 598 筆行為數據...

👤 用戶 ID: 537192226
🛒 行為: view | 商品: haier - electronics.video.tv
📦 價格: $193.03
🕵️ 偵測場景: window_shopping
🧠 心理原理: 探索階段 / 動機低
📢 Agent 生成廣告:
Haier智慧電視，$193超值首購專享！探索新科技，不等人！早鳥特惠，機不可失！
--------------------------------------------------
👤 用戶 ID: 550050854
```

```
🛒 行為: view | 商品: hp - computers.notebook
📦 價格: $1512.78
👮 偵測場景: price_comparison
🧠 心理原理: 價格敏感 / 損失厭惡
🦾 Agent 生成廣告:
HP筆記本電腦現價$1513，比全年95%的日子都便宜，錯過等一年！快來鎖定 Deals!
--------------------------------------------------
👤 用戶 ID: 552783882
🛒 行為: view | 商品: samsung - electronics.smartphone
📦 價格: $254.82
👮 偵測場景: window_shopping
🧠 心理原理: 探索階段 / 動機低
🦾 Agent 生成廣告:
Samsung新機在清單裡嗎? 首購享獨家折扣，機不可失，時不再來!
--------------------------------------------------
```

```python
import pandas as pd
import torch
import os
from datetime import datetime

# ==========================================
# 1. 策略庫 (7 大策略)
# ==========================================
STRATEGY_BANK = {
    "cart_abandonment": {
        "insight": "現在偏誤 / 延遲折扣偏誤",
        "tactic": "創造焦慮感，給予下單壓力",
        "example": "你放在購物車裡的優惠只剩今天。逾時，連我們都幫不了你。"
    },
    "price_comparison": {
        "insight": "價格敏感 / 損失厭惡",
        "tactic": "主動提供比價，強調現在買最划算",
        "example": "現在下單比全年 95% 的價格都低，錯過等一年。"
    },
    "repetitive_view": {
        "insight": "決策疲乏 / 選擇麻痺",
        "tactic": "提供社會認同，簡化選擇",
        "example": "你已經看了好多次，不如讓它成為你每天的輕鬆選擇。"
    },
    "window_shopping": {
        "insight": "探索階段 / 動機低",
        "tactic": "建立關係，提供首購優惠",
        "example": "清單裡的熱門款即將售完，你等的不是降價，是缺貨。"
    },
    "late_night_browsing": {
        "insight": "行為拖延 / 罪惡感 / 衝動控制低",
        "tactic": "感性訴求，強調自我獎勵",
        "example": "我知道你在猶豫。睡前買下最想要的，讓明天的你更開心。"
    },
    "size_anxiety": {
        "insight": "訊息不足偏誤 / 風險趨避",
        "tactic": "提供具體對照，降低決策風險",
        "example": "不確定尺寸? 我們有 1:1 真人對照圖，讓你 0 猜測，不合包退。"
    },
    "regret_aversion": {
        "insight": "後悔厭惡 / 風險放大偏誤",
        "tactic": "強調鑑賞期與退貨保證，消除後顧之憂",
        "example": "30 天試用不滿意全額退，我們替你擋住所有後悔。"
    }
}

# ==========================================
# 2. Agent 思考函式
# ==========================================
def run_agent(user_data: pd.Series):
    # --- A. 規則判斷 ---
    scenario = "window_shopping"
    status_desc = "只是逛逛"

    # 解析時間: 更穩健 (避免格式怪掉)
    event_time_str = str(user_data.get("event_time", "")).replace(" UTC", "")
    hour = 12
    try:
        dt = datetime.strptime(event_time_str, "%Y-%m-%d %H:%M:%S")
        hour = dt.hour
    except Exception:
        # 解析失敗就維持 12
        pass
```

```
        category = str(user_data.get("category_code", ""))
        try:
            price = float(user_data.get("price", 0))
        except Exception:
            price = 0.0

        # 預先算好的瀏覽次數（若沒有欄位，預設 1）
        view_count = user_data.get("view_count", 1)
        try:
            view_count = int(view_count)
        except Exception:
            view_count = 1

        event_type = user_data.get("event_type", "")

        # --- 邏輯判斷樹（注意優先順序） ---
        if event_type == "cart":
            scenario = "cart_abandonment"
            status_desc = "🛒 放入購物車未結帳"

        elif event_type == "view" and view_count >= 3:
            scenario = "repetitive_view"
            status_desc = f"👀 重複瀏覽 ({view_count}次) - 猶豫不決"

        elif event_type == "view" and (hour >= 23 or hour <= 4):
            scenario = "late_night_browsing"
            status_desc = f"🌙 深夜瀏覽 ({hour}點)"

        elif event_type == "view" and (("apparel" in category) or ("shoes" in category)):
            scenario = "size_anxiety"
            status_desc = "👕 瀏覽服飾/鞋類 (擔心尺寸)"

        elif event_type == "view" and price > 800:
            scenario = "regret_aversion"
            status_desc = "💎 極高價商品 (怕後悔)"

        elif event_type == "view" and price > 300:
            scenario = "price_comparison"
            status_desc = "💰 中高價商品 (比價中)"

        strategy = STRATEGY_BANK.get(scenario, STRATEGY_BANK["window_shopping"])

        brand = user_data.get("brand", "未知品牌")
        cat = user_data.get("category_code", "未知類別")

        # --- B. 組合 Prompt（加上「禁止只回你好」） ---
        prompt = f"""
請將「參考句子」改寫成一則給台灣客戶的繁體中文廣告短訊。

商品名稱：{brand} ({cat})
參考句子：{strategy['example']}
行銷策略：{strategy['tactic']}

限制：
- 20～50 字
- 請勿只回「你好」或任何招呼語
- 語氣自然、有行動誘因（例如：免運/限時/保障/熱賣其一）

改寫後的繁體中文廣告：
""".strip()

        # --- C. 生成（修復：避免秒停只吐「你好」） ---
        messages = [{"role": "user", "content": prompt}]
        text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)

        if tokenizer.pad_token is None:
            tokenizer.pad_token = tokenizer.eos_token

        inputs = tokenizer(
            text,
            return_tensors="pt",
            padding=True,
            truncation=True
        ).to(model.device)

        generated = model.generate(
            input_ids=inputs.input_ids,
            attention_mask=inputs.attention_mask,
            max_new_tokens=120,
```

```
            min_new_tokens=25,          # ✅ 重要：避免只生成幾個字就 EOS
            do_sample=True,
            temperature=0.9,
            top_p=0.9,
            repetition_penalty=1.05,
            eos_token_id=tokenizer.eos_token_id,
            pad_token_id=tokenizer.eos_token_id
        )

        input_len = inputs.input_ids.shape[1]
        response = tokenizer.decode(generated[0, input_len:], skip_special_tokens=True).strip()

        # ✅ 防呆：太短就再生一次（更強約束）
        if len(response) < 10:
            fallback = prompt + "\n⚠️ 請輸出 20～50 字完整廣告，不要招呼語。"
            messages = [{"role": "user", "content": fallback}]
            text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
            inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True).to(model.device)

            generated = model.generate(
                input_ids=inputs.input_ids,
                attention_mask=inputs.attention_mask,
                max_new_tokens=140,
                min_new_tokens=35,
                do_sample=True,
                temperature=0.85,
                top_p=0.9,
                repetition_penalty=1.08,
                eos_token_id=tokenizer.eos_token_id,
                pad_token_id=tokenizer.eos_token_id
            )
            input_len = inputs.input_ids.shape[1]
            response = tokenizer.decode(generated[0, input_len:], skip_special_tokens=True).strip()

        return {
            "user_id": user_data.get("user_id", "N/A"),
            "product_info": f"{brand} - {cat}",
            "price": user_data.get("price", "N/A"),
            "detected_scenario": status_desc,
            "strategy_insight": strategy["insight"],
            "ai_ad_copy": response
        }


    # ==========================================
    # 3. 執行並搜尋特定案例 (含資料預處理)
    # ==========================================
    csv_file = "2019-Oct.csv"

    if os.path.exists(csv_file):
        print("📂 正在讀取資料並計算瀏覽次數...")
        df = pd.read_csv(csv_file, nrows=20000)
        df_clean = df.dropna(subset=["category_code", "brand", "event_time"]).reset_index(drop=True)

        # 計算每個用戶看每個商品的 view 次數 (每筆資料都會得到相同 view_count)
        df_clean["view_count"] = df_clean.groupby(["user_id", "product_id"])["event_type"] \
            .transform(lambda x: (x == "view").sum())

        # 小心時間解析錯誤
        df_clean["hour"] = pd.to_datetime(
            df_clean["event_time"].astype(str).str.replace(" UTC", ""),
            errors="coerce"
        ).dt.hour.fillna(12).astype(int)

        test_users = []

        # 1) 重複瀏覽
        repetitive_users = df_clean[df_clean["view_count"] >= 3]
        if not repetitive_users.empty:
            test_users.append(repetitive_users.iloc[0])
        else:
            print("⚠️ 沒抓到重複瀏覽的用戶，可能資料量不夠")

        # 2) 深夜滑手機 (最好也限制 view)
        night_owls = df_clean[((df_clean["hour"] >= 23) | (df_clean["hour"] <= 4)) & (df_clean["event_type"] ==
        if not night_owls.empty:
            test_users.append(night_owls.iloc[0])

        # 3) 看衣服/鞋子
```

```
    fashion_users = df_clean[df_clean["category_code"].astype(str).str.contains("apparel|shoes", na=False)
    if not fashion_users.empty:
        test_users.append(fashion_users.iloc[0])

    # 4) 極高價
    rich_users = df_clean[(df_clean["price"] > 1000) & (df_clean["event_type"] == "view")]
    if not rich_users.empty:
        test_users.append(rich_users.iloc[0])

    # 5) 購物車遺棄
    cart_users = df_clean[df_clean["event_type"] == "cart"]
    if not cart_users.empty:
        test_users.append(cart_users.iloc[0])

    print(f"🤖 Agent 啟動！共找到 {len(test_users)} 種不同情境的用戶...\n")

    for idx, user_row in enumerate(test_users):
        result = run_agent(user_row)

        print(f"🧪 案例測試 #{idx+1}")
        print(f"👤 用戶 ID: {result['user_id']}")
        print(f"🛒 行為: {user_row.get('event_type', 'N/A')} | 商品: {user_row.get('brand', 'N/A')} - {user_
        print(f"📦 價格: ${user_row.get('price', 'N/A')}")
        print(f"🕵️ 偵測場景: {result['detected_scenario']}")
        print(f"🧠 心理原理: {result['strategy_insight']}")
        print(f"📢 Agent 生成廣告:\n{result['ai_ad_copy']}")
        print("-" * 50)

else:
    print("❌ 找不到 2019-Oct.csv")
```

```
📂 正在讀取資料並計算瀏覽次數...
🤖 Agent 啟動！共找到 5 種不同情境的用戶...

🧪 案例測試 #1
👤 用戶 ID: 545323115
🛒 行為: view | 商品: respect - apparel.shoes.keds
📦 價格: $66.67
🕵️ 偵測場景: 👀 重複瀏覽 (4次) - 猶豫不決
🧠 心理原理: 決策疲乏 / 選擇麻痺
📢 Agent 生成廣告:
Already seen it多次? 何不每天穿上Keds，展現獨特風格? 即購即送运费， LIMITED TIME OFFER!
--------------------------------------------------
🧪 案例測試 #2
👤 用戶 ID: 554748717
🛒 行為: view | 商品: aqua - appliances.environment.water_heater
📦 價格: $33.2
🕵️ 偵測場景: 🌙 深夜瀏覽 (0點)
🧠 心理原理: 行為拖延 / 罪惡感 / 衝動控制低
📢 Agent 生成廣告:
睡前享受即熱水，明日心情更美好! 把握機會，今夜即購即送運費，熱 SALE 中!
--------------------------------------------------
🧪 案例測試 #3
👤 用戶 ID: 520571932
🛒 行為: view | 商品: baden - apparel.shoes.keds
📦 價格: $102.71
🕵️ 偵測場景: 🌙 深夜瀏覽 (0點)
🧠 心理原理: 行為拖延 / 罪惡感 / 衝動控制低
📢 Agent 生成廣告:
我知道你考慮著，夜深了就讓自己放鬆吧! 晚上下單Baden Keds，享受免運費优惠， Discounts Ends Soon!
--------------------------------------------------
🧪 案例測試 #4
👤 用戶 ID: 535871217
🛒 行為: view | 商品: apple - electronics.smartphone
📦 價格: $1081.98
🕵️ 偵測場景: 🌙 深夜瀏覽 (0點)
🧠 心理原理: 行為拖延 / 罪惡感 / 衝動控制低
📢 Agent 生成廣告:
APPLE智慧型手機，今夜就買，享受科技樂趣，明日的你會更開心! 限量特惠，速點購不鏽运费。
--------------------------------------------------
🧪 案例測試 #5
👤 用戶 ID: 524325294
🛒 行為: cart | 商品: apple - electronics.smartphone
📦 價格: $515.67
🕵️ 偵測場景: 🛒 放入購物車未結帳
🧠 心理原理: 現在偏誤 / 延遲折扣偏誤
📢 Agent 生成廣告:
你的Apple智慧型手機已在購物車中等著你，今天最後機會! 逾時免運享优惠即 Ends。
--------------------------------------------------
```