這是一份專為 Cursor (或任何 AI 輔助編碼工具) 設計的 Spec-Driven Development (SDD) 文件。

這份文件的撰寫風格是高度技術導向的，我將之前 PRD 中的業務邏輯轉化為精確的「偽代碼（Pseudo-code）」與「數據定義」，讓 Cursor 可以直接讀取並生成高品質的 Python 程式碼。

你可以直接複製下方的 markdown 內容，存成 `DIT_Analysis_Spec.md`，然後拖進 Cursor 的對話框中。

給您的行動指南

1. **複製內容**：將下方分隔線內的內容存為 `.md` 檔。
2. **開啟 Cursor**：打開 Cursor，按 `Ctrl+I` (Composer) 或 `Ctrl+L` (Chat)。
3. **輸入指令**：輸入「@DIT_Analysis_Spec.md 請根據這份 Spec 文件，幫我撰寫 Python 程式碼，使用 Pandas 進行數據處理。」

DIT Analysis System - Specification Document

1. Project Overview

- **Project Name**: DIT (Design-In Tracking) Intelligent Analytics Module
- **Goal**: To parse a specific sales CSV report and generate actionable insights based on rule-based logic.
- **Tech Stack**: Python 3.9+, Pandas, Numpy.
- **Input**: CSV file (`DIT report.csv`).
- **Output**: JSON format actionable insights (Action Cards) and statistical summary.

2. Data Structure (Input Schema)

The input is a CSV file. The system must handle the following key columns. Note that column names may contain leading/trailing spaces, so stripping whitespace is required during pre-processing.

Column Name	Data Type	Description
`Created Date`	Date	Project creation date.
`Account Name`	String	Customer name.
`Stage`	String	Current status (e.g., "Negotiation", "Opportunity Lost", "Won").

Column Name	Data Type	Description
`Application`	String	Primary application field (may be empty, needs fallback).
`Application Detail`	String	Secondary application info (fallback for Application).
`Opportunity Name`	String	Project name (fallback for Application).
`Total Price`	Float	Potential Revenue (EAU * Price).
`Approved date`	Date	The date when the technical design was approved.
`Lost Type`	String	Reason for loss (e.g., "Spec", "Price").

3. Data Pre-processing Rules

1. **Column Cleaning**: Strip whitespace from all column headers.
2. **Date Parsing**: Convert `Created Date`, `Approved date`, `Close Date` to datetime objects. Handle errors (`errors='coerce'`).
3. **Numeric Conversion**: Convert `Total Price` to numeric, filling `NaN` with `0`.
4. **Application Derivation**:
    - Create a new column `Derived_Application`.
    - Logic: If `Application` is null, use `Application Detail`. If that is null, use `Opportunity Name`. If that is null, use "Unknown".
5. **Status Flags**:
    - `Is_Lost`: True if `Stage` contains "Lost".
    - `Is_Active`: True if `Stage` is NOT "Lost" (includes "Negotiation", "Won", "Mass Production").

4. Feature Specifications (Core Logic)

Feature 6.2: High Value Resource Allocation (Resource Allocation Advisor)

- **Objective**: Identify high-potential application areas where the win rate is surprisingly low.

- **Logic Steps**:
    1. Group data by `Derived_Application`.
    2. Calculate metrics for each group:
        - `Sum_Total_Price`: Sum of `Total Price`.
        - `Win_Rate`: Mean of `Is_Active` (Simulation of Win Rate).
    3. **Filter Logic (The Trigger)**:
        - Sort groups by `Sum_Total_Price` descending.
        - Identify the **Top 20%** of applications by value.
        - Within this Top 20%, select applications where `Win_Rate` < **10%** (0.1).
    4. **Output Generation**:
        - Return a list of dictionary objects containing: `{Title, Application, Money, WinRate, Suggestion}`.
        - **Suggestion Template**: "High potential market ($[Money]), but current win rate is low ([WinRate]%). Suggest assigning senior FAE to top accounts."

Feature 6.3: Stagnant Deal Alert (Sales Cycle Booster)

- **Objective**: Alert on deals that have passed technical approval but are stuck in negotiation.

- **Logic Steps**:
    1. Filter dataset for rows where:
        - `Stage` == "Negotiation" (Case insensitive).
        - `Approved date` is NOT Null (NaT).
    2. Calculate `Days_Since_Approved`:

        Formula: `Current_Date` - `Approved date`.
    3. **Filter Logic (The Trigger)**:

        Select rows where `Days_Since_Approved` > **60**.
    4. **Output Generation**:
        - Return a list of dictionary objects containing: `{Title, Account, Project, ApprovedDate, Days_Pending, Suggestion}`.
        - **Suggestion Template**: "Customer [Account] project [Project] approved for [Days] days but not yet ordered. Check for price/inventory issues."

5. Class Architecture Suggestion

```
class DITAnalyzer:
    def __init__(self, file_path):
        self.df = None
        self.load_data(file_path)

    def load_data(self, file_path):
        # Implementation of Section 3 (Pre-processing)
        pass

    def get_high_value_allocation_suggestions(self):
        # Implementation of Feature 6.2
        pass

    def get_stagnant_deal_alerts(self, threshold_days=60):
        # Implementation of Feature 6.3
        pass

    def run_full_analysis(self):
        # Aggregates all insights
        return {
            "allocation_suggestions": self.get_high_value_allocation_suggestions(),
            "stagnant_alerts": self.get_stagnant_deal_alerts()
        }
```

6. Implementation Notes for Developer (Cursor)

- **Library**: Use `pandas` for all data manipulation.
- **Error Handling**: Ensure the code does not crash if `Approved date` column is entirely empty or missing; simply return an empty list for Feature 6.3 in that case.
- **Simulation**: Since the current dataset might be small or lack specific dates, allow a `reference_date` parameter in methods (default to `datetime.now()`) to calculate date differences accurately.

產品需求文件 (PRD) - DIT 智能分析系統

文件名稱：DIT Intelligent Analytics Module (DIT 智能分析模組) **版本**：v1.0 **日期**：2025-12-12 **狀態**：已定案 (Approved)

1. 專案概述 (Project Overview)

1.1 背景

目前的 Design-In Tracking (DIT) 報表包含大量未被挖掘的價值。業務主管與 PM 需要一套自動化工具，能從 Excel/CSV 報表中識別出「高價值但低勝率」的市場機會，以及「卡關過久」的風險案件。

1.2 目標

開發一個 Python 分析模組，讀取 DIT CSV 原始檔，透過規則基礎（Rule-Based）演算法，自動輸出具體的「行動建議卡片（Action Cards）」。

1.3 範圍 (Scope)

本階段 (v1.0) 專注於後端邏輯與數據處理，核心功能為：

1. **資料清洗與預處理**。
2. **高潛力市場偵測 (Feature 6.2)**。
3. **呆滯案件警示 (Feature 6.3)**。

2. 資料需求 (Data Requirements)

系統需讀取 `.csv` 格式檔案，並處理以下關鍵欄位。若欄位缺失或格式錯誤，需進行防呆處理。

欄位名稱 (Header)	資料型態	必填	說明
`Account Name`	String	Yes	客戶名稱
`Stage`	String	Yes	專案階段 (e.g., Negotiation, Opportunity Lost)
`Application`	String	No	應用領域 (若為空，需依照下方邏輯進行替補)
`Application Detail`	String	No	應用細節 (Application 的第一替補)
`Opportunity Name`	String	Yes	專案名稱 (Application 的第二替補)
`Total Price`	Float	Yes	預估總金額 (EAU * Price)，需處理非數字字元
`Approved date`	Date	No	技術承認通過日期，用於計算呆滯天數
`Created Date`	Date	Yes	立案日期

3. 功能需求 (Functional Requirements)

3.1 資料預處理模組 (Data Preprocessor)

- **欄位正規化**：移除欄位名稱前後的空白。
- **應用領域推導 (Derived Application)**：
  - 建立新欄位 `derived_app`。
  - 邏輯：優先使用 `Application`；若為空，使用 `Application Detail`；若仍為空，使用 `Opportunity Name`；最後補上 "Unknown"。
- **狀態標記**：
  - `is_lost`: 當 `Stage` 包含 "Lost" 或 "Design-Lost" 時為 True。
  - `is_active`: 當 `is_lost` 為 False 時為 True (視為潛在 Win)。

3.2 核心功能：高價值資源分配建議 (Feature 6.2)

- **商業目的**：識別「金礦區」，即金額龐大但目前我方勝率極低的領域。

- **觸發邏輯 (Trigger Logic)**：
    1. 將資料依照 `derived_app` 分組。
    2. 計算該分組的總金額 (`sum_total_price`) 與 存活率/勝率 (`win_rate` = `is_active` 的平均值)。
    3. **篩選條件**：
        - 總金額排名在全體的前 **20%**。
        - **且** 存活率 (`win_rate`) 低於 **10%** (0.1)。
- **輸出格式 (Action Card)**：
    - 標題：`高潛力市場攻堅提醒`
    - 內容：`[App名稱] 領域潛在商機巨大 ($[金額])，但目前勝率偏低 ([勝率]%)。`
    - 建議行動：`建議指派資深 FAE 介入該領域的前三大案子（如 [該組最大的 Account Name]）。`

3.3 核心功能：銷售週期推進/呆滯警示 (Feature 6.3)

- **商業目的**：針對技術已承認但商務未轉單的案子進行催單。
- **觸發邏輯 (Trigger Logic)**：
    1. 篩選 `Stage` 為 "Negotiation" 的案件。
    2. 篩選 `Approved date` **不為空值** 的案件。
    3. 計算 `days_pending` = (當前日期 - `Approved date`)。
    4. **篩選條件**：

        `days_pending` > **60** 天。
- **輸出格式 (Action Card)**：
    - 標題：`呆滯案件喚醒 (Stagnant Deal Alert)`
    - 內容：`客戶 [Account Name] 的 [Project Name] 已承認超過 [月份數] 個月，仍未轉單。`
    - 建議行動：`請業務確認是否為「價格」或「庫存」問題。若無下文，應要求客戶給出 Forecast。`

4. 系統架構與實作建議 (System Architecture)

建議採用物件導向 (OOP) 結構設計，以利後續擴充圖表功能。

```
class DITAnalyzer:
    def __init__(self, file_path):
        """初始化並載入 CSV 資料"""
        pass

    def _preprocess(self):
        """執行資料清洗與型態轉換"""
        pass

    def analyze_resource_allocation(self, top_percent=0.2, low_win_rate=0.1):
        """執行 Feature 6.2 邏輯"""
        pass

    def analyze_stagnant_deals(self, threshold_days=60):
        """執行 Feature 6.3 邏輯"""
        pass

    def generate_report(self):
        """彙整所有建議並輸出 JSON 或 Dictionary"""
        pass
```

5. 驗收標準 (Acceptance Criteria)

1. **容錯性**：程式碼在讀取含有空值 (`NaN`) 或格式錯誤日期的 CSV 時不應崩潰（Crash）。
2. **準確性**：對於 Feature 6.3，若 `Approved date` 空白，該筆資料應被自動忽略，不可報錯。
3. **可讀性**：輸出的建議內容需為繁體中文，且數值需包含千分位符號 (e.g., $300,000)。

給您的下一步 (Next Step)

現在您手上有兩份文件：

1. **Spec 文件 (.md)**：給 Cursor 看的，用來生成程式碼。
2. **PRD 文件 (.md)**：給您自己與團隊看的，用來確認需求與驗收成果。

請執行這個動作： 將這份 `DIT_Analysis_PRD.md` 存檔。打開 Cursor，將檔案拖入，並輸入以下指令：

> 「請依照這份 PRD 的架構，撰寫完整的 Python 程式碼。請建立一個 DITAnalyzer 類別，並包含一個 main 函式來測試執行。」

這會讓您立即得到一個可執行的程式原型。我們不空談，直接看結果。