Group 3 : Final Project Proposal – Part II

Members: 鄭仲軒、王彥翔、劉李洋

1. Non-trivial Functional Dependencies & Normalization Analysis

本章節針對系統設計的六張資料表進行功能依賴 (Functional Dependency, FD) 分析，並驗證其正規化程度。

- 商品

Product(ProductID, ProductName, BrandName, Status, Category)

主鍵（PK）: ProductID

1. Non-trivial FD：
    - ○ ProductID → ProductName, BrandName, Status, Category

2. Normal Form：
    - ○ 所有屬性為單一值，故為 1NF
    - ○ 主鍵為單一屬性，所有非鍵屬性都完全依賴 ProductID，無部分依賴，故為 2NF
    - ○ 不存在非鍵屬性經由其他非鍵屬性的傳遞依賴，且唯一的 non-trivial FD 左側為主鍵，故為 3NF
    - ○ 對每個 non-trivial FD，左側 ProductID 為 super key，故滿足 BCNF

- 型號

SKU(SKU_ID, ProductID, CPU, GPU, Storage, ScreenSize, Weight, Price, Stock)

主鍵（PK）: SKU_ID
外鍵（FK）: ProductID → Product.ProductID

1. Non-trivial FD：
    - ○ SKU_ID → ProductID, CPU, GPU, Storage, ScreenSize, Weight, Price, Stock

2. Normal Form ：

○ 由於所有欄位皆為單一值，故為 1NF。

○ 主鍵為單一屬性 SKU_ID，所有非鍵屬性皆完全依賴主鍵，無部分依賴，故為 2NF。

○ 不存在非鍵屬性決定其他非鍵屬性的傳遞依賴，且唯一的 non-trivial FD 左側為主鍵，故為 3NF。

○ 關係中的每一個 non-trivial FD，其左側 SKU_ID 為 super key，因此滿足 BCNF。

- 訂單

Order(Order_ID, Customer_ID, Address_ID, OrderDate, Status, TotalAmount)

主鍵（PK）： Order_ID

外鍵（FK）： Customer_ID → Customer.Customer_ID

Address_ID → AddressBook.Address_ID

1. Non-trivial FD：
   ○ Order_ID → Customer_ID, Address_ID, OrderDate, Status, TotalAmount

2. Normal Form：

   ○ 由於所有欄位皆為單一值，故為 1NF
   ○ 主鍵為單一屬性 OrderID，無部分依賴，故為 2NF
   ○ 不存在非鍵屬性決定其他非鍵屬性的傳遞依賴，且唯一的 non-trivial FD 左側為主鍵，故為 3NF。雖然符合 3NF 結構，但 TotalAmount 屬性在理論上為衍生屬性，存在資料冗餘風險
   ○ 為了效能與資料獨立性，我們在Customer_ID處進行了適度的反正規化設計，雖然形式上存在對Address_ID的依賴，但這是為了確保當地址被刪除時，訂單的主人不會遺失。

- 訂單品項

OrderItem(OrderItemID, OrderID, SKUID, Quantity)

主鍵（PK）： OrderItemID

外鍵（FK）： OrderID → Order.OrderID

SKUID → SKU.SKUID

1. Non-trivial FD：
   - ○ OrderItemID → OrderID, SKUID, Quantity
2. Normal Form：
   - ○ 由於所有欄位皆為單一值，故為 1NF
   - ○ 主鍵為單一屬性 OrderItemID（代理鍵），所有非鍵屬性皆完全依賴主鍵，無部分依賴，故為 2NF
   - ○ 不存在非鍵屬性決定其他非鍵屬性的傳遞依賴，所有屬性皆直接依賴於 OrderItemID，且唯一的 non-trivial FD 左側為主鍵，故為 3NF
   - ○ 關係中的每一個 non-trivial FD，其左側 OrderItemID 為 super key，因此滿足 BCNF。

- 顧客

Customer(CustomerID, Email, Password, Name, Phone)

主鍵（PK）： CustomerID

- Non-trivial FD： CustomerID → Email, Password, Name, Phone

- Normal Form：
  - ○ 由於所有欄位皆為單一原子值，故為 1NF。
  - ○ 主鍵為單一屬性 CustomerID，所有非鍵屬性皆完全依賴主鍵，無部分依賴，故為 2NF。
  - ○ 不存在非鍵屬性決定其他非鍵屬性的傳遞依賴，所有的個人資料屬性皆直接相依於 CustomerID，且唯一的 non-trivial FD 左側為主鍵，故為 3NF。

○ 關係中的每一個 non-trivial FD，其左側 CustomerID 為 super key，因此滿足 BCNF

- 收件資訊

AddressBook(AddressID, CustomerID, ReceiverName, Phone, Address, PaymentMethod)

主鍵（PK）： AddressID

外鍵（FK）： CustomerID → Customer.CustomerID

1. Non-trivial FD： AddressID → CustomerID, ReceiverName, Phone, Address, PaymentMethod

2. Normal Form：
   ○ 由於所有欄位皆為單一值，地址視為單一的配送單元，故為 1NF。
   ○ 主鍵為單一屬性 AddressID，所有非鍵屬性皆完全依賴主鍵，無部分依賴，故為 2NF。
   ○ 所有非鍵屬性，包含外鍵 CustomerID 皆直接依賴於 AddressID，不存在非鍵屬性間的傳遞依賴，且唯一的 non-trivial FD 左側為主鍵，故為 3NF。
   ○ 關係中的每一個 non-trivial FD，其左側 AddressID 為 super key，因此滿足 BCNF。

—----------

有更動關聯表

- 訂單 (Order)

Order(Order_ID, Customer_ID, Address_ID, OrderDate, Status) (已移除 TotalAmount 因為其為衍伸屬性 可以從 SKU.Price * OrderItem.Quantity 得到)

主鍵（PK）： Order_ID

外鍵（FK）： Customer_ID → Customer.Customer_ID

Address_ID → AddressBook.Address_ID

1. Non-trivial FD：
   ○ Order_ID → Customer_ID, Address_ID, OrderDate, Status

2. Normal Form：

- 由於所有欄位皆為單一值，故為 1NF
- 主鍵為單一屬性 Order_ID，無部分依賴，故符合 2NF。
- 我們移除了原有的 TotalAmount (總金額) 欄位，因為它屬於可被計算的衍生屬性。剩餘的所有屬性皆直接依賴於 Order_ID，不存在非鍵屬性決定其他非鍵屬性的情況，故符合 3NF。
- FD 左側 Order_ID 為 super key，因此滿足 BCNF。

2.Real-World Data Search

整理好的資料連結：

https://github.com/SunnyTee2005/Junior-Year-Data-Base-Management-Final-Project.git

1. 資料來源識別 (Source Identification)

- 資料集名稱: Laptop Price & Specifications Dataset
- 來源平台: Kaggle (基於真實電商網站爬蟲)
- 檔案格式: CSV (Comma-Separated Values)
- 資料規模: 3,976 筆 (Rows)
- 關聯性: 包含 Brand, CPU, GPU, RAM, Storage, Screen, Price，完整覆蓋本專案 Product 與 SKU 實體所需屬性。

2. 資料處理策略 (ETL Strategy)

原始資料為非正規化 (Denormalized) 的扁平表格。為符合本組設計的 BCNF Schema，我們使用 Python (Pandas) 執行以下轉換：

- 實體拆分 (Entity Splitting): 將單一 CSV 拆解為Product(商品系列)與SKU(規格型號) 兩張表，並透過 Foreign Key 重新建立關聯。
- 屬性清洗 (Data Cleaning): 將 RAM 由字串 ("16 GB") 轉換為整數 (16)；標準化 SSD/HDD 儲存空間描述，以利 SQL 數值查詢。

3. AI 協作與數據增強 (AI-Assisted Enrichment with Gemini)

為解決原始資料的缺失值與格式問題，本專案導入 Gemini 協助開發 Python ETL 腳本，實現高擬真數據處理：

- 真實料號提取 (Natural Key Extraction): 利用 Gemini 優化的 Regex 邏輯，從雜亂的商品名稱中精準提取廠商真實料號 (e.g., 82KU017KIN)，取代無意義的流水號作為 SKU 主鍵。
- 啟發式重量估算 (Heuristic Weight Estimation): 針對缺失的重量欄位，不採用隨機生成。我們採用 Gemini 建議的物理邏輯演算法，依據「螢幕尺寸」與「GPU 等級 (是否為電競顯卡)」自動推算逼真的重量數值 (e.g., 電競機型自動加權)。

4. 交易資料生成 (Transactional Mock Data)

考量個資隱私與資料庫邏輯完整性，Customer, AddressBook, Order 等表單不使用真實數據。 改採 Python Faker 套件生成符合台灣在地格式 (zh_TW) 的測試資料，並確保訂單與地址的參照完整性 (Referential Integrity)。