專案企劃書：Hybrid Search Optimizer (混合搜尋優化器)

—— 基於成本模型 (Cost-Based) 的多模態動態檢索系統

1. 專案背景與動機 (Background & Motivation)

在現代電商與多媒體檢索場景中，單一的搜尋模式已無法滿足需求：

- **關鍵字搜尋 (Keyword Search)**：能精確篩選價格區間或品牌（如 SQL `WHERE`），但無法理解圖片的視覺特徵。
- **向量搜尋 (Vector Search)**：能透過 AI 模型（如 CLIP）理解語意與視覺相似度，但難以處理硬性條件（如「價格 < 500」），容易產生「漏斗效應」導致結果不精準。

當系統需要同時處理「語意相似度」與「結構化篩選」時，工程師面臨經典的**「執行路徑兩難 (Trade-off)」**：

1. **先篩選 (Plan A - SQL First)**：精準但慢（資料量大時需進行大量向量運算）。
2. **先向量 (Plan B - Vector First)**：快但不準（HNSW 索引可能在第一階段就漏掉符合篩選條件的稀有資料）。

本專案旨在開發一套 **「智慧型中介層 (Intelligent Middleware)」**，解決此效能與準確度的平衡問題。

2. 系統建置與運作流程 (System Architecture & Workflow)

本系統分為 **「資料庫建置階段」** 與 **「即時查詢階段」**，採用 **Late Fusion (後期融合)** 策略以保留最大的查詢靈活性。

2.1 資料庫建置階段 (Phase 1: Database Construction)

此階段目標為將非結構化圖片轉換為可被數學運算的向量，並建立雙重索引結構。

1. **特徵提取 (Feature Extraction)**：
    - 採用 **OpenAI CLIP (ViT-L/14)** 模型作為 Embedding Backbone。

      可以考慮別的模型
      也可以考慮將圖片的特徵或描述當成文字一起餵入資料庫
    - 將每一張商品圖片映射至 **768維** 的高維向量空間。
    - *策略備註*：僅儲存純圖片向量，不預先混入文字描述，確保後續能進行任意的文字微調 (Text Modification)。
      可以考慮將圖片被去景或用傅立葉來降躁
2. **資料注入 (Data Ingestion)**：
    - 將商品 Metadata（品牌、價格、類別）存入 PostgreSQL 結構化欄位。
    - 將 768維向量存入 `vector` 類型欄位 (透過 `pgvector` 擴充)。
3. **雙重索引建構 (Dual Indexing)**：
    - **B-Tree Index**：針對 Metadata（如 `sales_price`）建立索引，加速 SQL 篩選（服務 Plan A）。
    - **HNSW Index** (Hierarchical Navigable Small World)：針對向量欄位建立圖形索引，加速近似最近鄰搜尋（服務 Plan B）。

2.2 即時查詢階段 (Phase 2: Runtime Execution Workflow)

此階段為系統核心，負責處理使用者請求並動態決定執行路徑。

- **Step 1: 多模態向量合成 (Query Composition)**
    - 接收輸入：基準圖片 + 微調文字（例如：「紅色」）。
    - **幾何運算**：使用 **球面線性插值 (Slerp)** 將圖片向量與文字向量進行加權融合。
    - *數學優勢*：Slerp 沿著單位球體表面路徑移動，能避免傳統線性插值 (Linear Interpolation) 導致向量模長萎縮與特徵失真的問題。

      多維度了話為什麼是球形？
- **Step 2: 成本預估 (Cost Estimation)**
    - 系統攔截 SQL 篩選條件（例如 `WHERE price < 1000`）。
      這邊要解釋一下 為什麼我們只模擬電商的價格搜尋
      因為資料量太少
    - **詢問統計層**：執行 `EXPLAIN` 指令查詢 PostgreSQL 系統表 (** `pg_statistic` **)。
      這個總表是怎麼來的 可以介紹一下，比方說用哪種資料結構
    - 預期筆數 **(Cardinality)**：取得該條件預期會篩選出的資料筆數 (Plan Rows)，此過程不涉及實際資料讀取，延遲極低。

- **Step 3: CBO 決策 (Smart Routing)**
    - 將「預期筆數」代入成本模型 (Cost Model) 比較兩條路徑的預估耗時：
        - **路徑 A (Plan A - Exact Search)**：先執行 SQL 篩選，再對剩餘資料進行 KNN (Exact) 暴力向量運算。
            - *適用情境*：稀有查詢 (Rare Query)，資料筆數少，追求 100% 準確率。
              這部分我們要怎麼算時間，我們可以模擬有10,100,200,500筆資料的時候，所花的時間，然後用回歸去算模型大概要多花多久
        - **路徑 B (Plan B - ANN Search)**：先執行 HNSW 向量索引(ANN)搜尋 Top-K，再進行 SQL 過濾。
            - *適用情境*：大眾查詢 (Broad Query)，資料筆數多，追求極致效能。
- **Step 4: 執行與回傳 (Execution)**
    - 系統依據決策結果，生成對應的 SQL 語句 (CTE 或 Subquery) 並發送至資料庫執行，最終回傳排序後的 Top-N 商品。

3. 技術亮點 (Technical Highlights)

- **雙路徑檢索機制 (Dual-Path Retrieval)**：同時實作 KNN (Exact) 與 HNSW (ANN)，並證明兩者在不同資料規模下的優劣。
- **成本導向決策 (Cost-Based Logic)**：不依賴人工規則 (Heuristic)，而是基於資料庫即時統計數據進行科學決策。
- **幾何特徵融合 (Geometric Fusion)**：使用 Slerp 處理多模態向量，確保「以圖搜圖」加上「文字微調」後的語意精確度。

4. 驗證與評估計畫 (Evaluation Plan)

為證明優化器的有效性，我們設計了包含「稀有查詢」與「大眾查詢」的實驗組：

- **基準 (Ground Truth)**：以 Plan A (全表掃描) 的結果作為標準答案。
- **評估指標**：
    1. **延遲 (Latency)**：比較 Plan A、Plan B 與 Optimizer 的執行時間。預期 Optimizer 的時間曲線應始終貼近兩者中的最小值。
    2. **召回率 (Recall of Top-5 in Top-20)**：
        - **設計邏輯**：考量電商使用者的「瀏覽行為 (Browsing Behavior)」，使用者通常會滑動頁面查看多筆結果。因此，我們不追求絕對的排序一

> 致 (Rank Exactness)，而是追求「高品質商品的可見度 (Visibility)」。

- **定義 (Definition)**：
  - **標準答案 (Ground Truth)**：由 Plan A (暴力掃描) 算出的 **Top 5** 最相似商品。
  - **檢索範圍 (Candidate Window)**：由 Plan B (向量索引) 或 Optimizer 回傳的 **Top 20** 商品。
- **評分方式**：計算「Plan A 的 Top 5 商品」有多少比例落入了「Plan B 的 Top 20 範圍」內。
  - *公式*：$Recall = \frac{\text{Count}(A_{top5} \cap B_{top20})}{5}$
  - *範例*：若 Plan A 認為最好的 5 個商品中，有 4 個出現在 Plan B 的前 20 名內，則召回率為 **80%** (視為高分)。
- **預期結果**：證明 Optimizer 在大幅縮短搜尋時間的同時，仍能保持極高的商品召回率，確保使用者不會錯過最相關的商品。

3. **定性盲測 (Human Evaluation)**：針對檢索結果進行人工滿意度評分，評估 AI 模型與使用者意圖的契合度。
   我可以餵到ai，然後去給他對是否符合圖片＋微調 去做評分？