Project Workflow Documentation

This project is organized into **three Jupyter notebooks** and one **utility module (`utils.py`)**. Each file has a well-defined role in the end-to-end semantic shoe recommendation workflow.

The following documentation explains:

- What each notebook does
- Why each step exists
- What utilities are available
- How everything connects together

🔷 1. shoes-data-generation.ipynb — Create Synthetic Shoe Dataset

This notebook is responsible for **building the entire dataset from scratch**.

Purpose

Generate artificial shoe product data and create embedding vectors using OpenAI.

Key Components

Feature	Description
Fake data generation (faker)	Uses `faker` library to build realistic product descriptions, brands, materials, cities, etc.
Generate shoe data (`shoes.csv`)	Creates the base dataset used by all downstream notebooks.
Embedding columns	Adds empty placeholder columns for vectors.
Generate embeddings using OpenAI	Calls OpenAI `text-embedding-3-large` for every shoe description.
Save vectors (`shoes-vectors.csv`)	Exports the enriched dataset containing embeddings.

Output Files

- `shoes.csv`
- `shoes-vectors.csv`

This notebook acts as the **data factory** for the entire project.

🔷 2. shoes-data-partitioning.ipynb — Store-wise Dataset Splitting

After generating the full dataset, the next step is to **simulate multiple store inventories**.

Purpose

Partition the complete shoe dataset by **store location**.

Key Components

Feature	Description
Partition dataset by store	Splits `shoes.csv` into multiple per-store CSVs (e.g., Ottawa, Toronto).

Why This Matters

Real retail systems have **different products available in each store**.
This step enables:

- Location-specific recommendations
- More realistic testing
- Faster vector search (smaller subsets)

Output Files

- `store_101_shoes.csv`
- `store_102_shoes.csv`
- ...
- `stores_summary.csv`

This notebook acts as the **data distribution engine** for the project.

🔷 3. shoes-search.ipynb — Semantic Search + Recommendation Engine

This is the **core notebook** where semantic similarity search happens using the vectors stored in Db2.

Purpose

Perform vector search inside Db2 to find similar shoes.

Key Components

Step	Description
OpenAI API Setup	Reuses OpenAI client for additional embeddings when needed.
Db2 connection (.env)	Connects to Db2 using environment variables.
Create Db2 shoe table	Builds `SQ_SHOES` with vector column.
Import `shoes-vectors.csv`	Loads vectorized data into Db2.
Search "running shoes"	Example semantic query using Db2 VECTOR_DISTANCE.
Verify chosen shoe in Ottawa	Query to check availability in specific stores.
Search similar shoes in Toronto	Store-specific recommendation example.
Compare search results with the preferred shoe	Demonstrates distance-based ranking and evaluation.

User Experience in This Notebook

You get results like:

- "Your selected shoe is available in Ottawa"
- "Top 3 similar shoes in Toronto"
- "Here is the closest match to your preferred shoe"

This notebook is the **brains of the recommendation pipeline**.

🔷 4. utils.py — Helper Functions for SQL, Images, and Visualization

This module contains all reusable helper functions that keep the notebooks clean and modular.

Categories & Functions

🖼️ A. Visualization Utilities

Function	Description
display_sku_images	Show shoe images in a grid layout.
plot_similarity_tsne	T-SNE visualization to plot vector similarity on a 2-D map.

🧮 B. SQL Helpers

These functions dynamically generate SQL queries used inside the notebooks.

Function	Purpose
get_men_shoes_sql	SQL query to retrieve men's shoes.
get_similar_shoes_sql	SQL for similarity-based ranking using vector distance.
get_distance_for_sku_list_sql	Compute VECTOR_DISTANCE between one shoe and a list of SKUs.
get_shoe_by_sku_sql	Fetch a product row for a given SKU.
get_similarity_query	Build complete similarity search query for Db2.

🗂️ C. Table Creation Helpers

Function	Purpose
get_create_sq_shoes_table_sql	SQL to create the Db2 table structure.
get_create_sq_shoes_table_data_only	SQL to create a “data-only” version of the table (without vectors).

🔍 D. Utility Helpers

Function	Purpose
find index of my_choice_sku	Utility to locate SKU position inside a DataFrame.

These utilities power the notebooks and ensure consistent behavior.

🔷 🎯 End-to-End Pipeline Summary

Here's the complete workflow:

1. **shoes-data-generation.ipynb**
   - Create dataset → Generate embeddings → Save vectorized CSV
2. **shoes-data-partitioning.ipynb**
   - Create per-store datasets → Simulate real multi-store environment
3. **shoes-search.ipynb**
   - Load vectors into Db2 → Run semantic vector search → Compare similarity → Visualize results
4. **utils.py**
   - SQL generators → Visualization helpers → Image rendering → T-SNE maps

This structure creates a **clean, production-like RAG (Retrieval-Augmented Generation) pipeline** fully backed by Db2 Vector AI + OpenAI.