Mohamed bin Zayed University of Artificial Intelligence

Individual Assignment 2 - Market Basket Analysis on Groceries

AI1030 - Python Programming

1 Introduction

You will perform an end-to-end market basket analysis on a real-life grocery dataset. You will load and inspect the CSV, clean/transform it into a transaction format, assign prices to products, enrich the data with basket totals, and compute co-occurrence statistics for product pairs and triples. You will also produce clear visualizations and a report explaining your methodology and your findings.

2 Data

File: `groceries.csv` (available in moodle). The schema may vary (e.g. one row per transaction with comma-separated items; or two columns like `TransactionID,Item`). Your first task is to infer and document the schema and transform it to a canonical format as follows:

- TransactionID: string or int,
- Items: list of product strings.

3 Tasks (Required)

A. Load & Understand (Reproducible)

1. Load `groceries.csv` with pandas.
2. Infer schema; print a data dictionary (columns, types, meaning).
3. Basic Exploratory Data Analysis (EDA): number of transactions and unique products; basket-size distribution (min/median/95th percentile); top 20 products by frequency.

B. Clean & Transform

1. Standardize item names (lowercase, strip whitespace; optional: replace spaces with underscores).
2. Remove empty/invalid items; drop baskets with fewer than 2 items (explain your choice).
3. Create a canonical transactions table with columns: `transaction_id`, `items` (list of strings), `basket_size`. Persist as `transactions_clean.parquet` (or CSV).

C. Assign Prices & Enrich

1. Create a product-level price map: assign each unique product a random unit price in a reasonable range (e.g., \$0.50–\$15.00) using a fixed random seed for reproducibility (e.g., `np.random.default_rng(42)`). Save as `product_prices.csv` with columns `product`, `price`.
2. Compute basket totals by summing unit prices for items in each transaction (assume quantity=1 unless specified).
3. Add a `basket_total` column to the transactions table and export as `transactions_priced.csv`.

D. Co-Occurrence Statistics

1. Count pairs and triples of items that occur in the same basket. Define support count (number of baskets containing the itemset).
2. Make `min_count` configurable (default 20). Return all pairs/triples meeting the threshold.
3. Compute top-k pairs and top-k triples by frequency (default k=10), with ties broken deterministically (alphabetical).
4. Report both support count and support fraction for each itemset.

E. Visualizations (at least 4)

1. Bar chart of the top 15 individual items by frequency.
2. Bar chart of top-k pairs by support fraction.
3. Heatmap of a co-occurrence matrix for the 25 most frequent items.
4. Distribution plot of `basket_size` and `basket_total` (histogram or ECDF).
5. (Optional) Network graph for top co-occurring pairs using NetworkX.

F. Performance & Reproducibility

1. Use efficient counting (e.g., `itertools.combinations` with `collections.Counter`).
2. Parameterize `min_count` and `k` so that they can be changed via variables or command-line flags.
3. Ensure that the notebook/script runs top-to-bottom without manual intervention; set a fixed random seed.

4 Extensions (Optional)

- Association rules: compute confidence and lift for top pairs/triples; interpret which are most informative.
- Stability: bootstrap the transaction set (e.g., 20 resamples) to estimate variability of top-k rankings.
- Revenue scenarios: randomly perturb prices ($\pm 10\%$) and simulate effects on basket totals and top-k revenue-contributing pairs.
- Sparse structures: build a product $\times$ transaction sparse matrix and derive co-occurrence via matrix ops.
- Quality report: automatic checks (duplicate IDs, malformed rows, outliers), with a summary table.

5 Deliverables

5.1 Declaration of Original Work

The following piece of text must be present in your submitted files.

I hereby declare that the work presented in the submitted report and the accompanying code is entirely my own. No portion of this submission has been copied, reproduced, or directly generated/refined using the responses or outputs of any AI tools (including, but not limited to, ChatGPT, Copilot, Gemini, DeepSeek, or other automated systems). Any external sources, datasets, or tools that have been used are properly cited and referenced. I understand that any breach of this declaration may result in submission cancellation or significant mark deduction.

5.2 Code & Reproducibility

- **Executable analysis:** A single Jupyter notebook *or* a set of Python scripts that runs top-to-bottom without manual intervention and implements Tasks A–F from the brief (load/EDA, clean/transform, pricing/enrichment, co/occurrence statistics, visualizations, and performance/-reproducibility controls).
- **Parameters:** All thresholds and options (`min_count`, `k`, random seed, figure limits, file paths) should be clearly defined as variables or CLI flags.
- **Outputs saved:**
  - `product_prices.csv` (`product`, `price`).
  - `transactions_priced.csv` (canonical transaction table with basket totals).

- **Efficiency:** Employs `itertools.combinations` and `collections.Counter` (or equivalent) for pair/triple counting; avoids redundant computation.
- **Determinism:** Fixed RNG seed ensures reproducibility across runs.
- **Packaging:** Includes a concise `README.md` detailing how to run, dependency setup, and where outputs are stored.

5.3 Report

- The report should be written in a professional and clear manner.
- Figures should be embedded within the report body, each referenced and discussed in context.
- The report must include:
  1. **Design & Data Model:** Description of the inferred raw schema, canonical transaction schema, and any auxiliary structures (e.g., price maps).
  2. **Methodology:** Cleaning rules (standardization, invalid removal, basket-size filters), parameter choices (`min_count`, k), and performance decisions.
  3. **Findings:** Key results from EDA and co-occurrence analyses, top pairs/triples with support statistics, and insights derived from visualizations.
  4. **Figures:** At least four figures integrated into the discussion:
     - Bar chart of top 15 individual items by frequency.
     - Bar chart of top-k pairs by support fraction.
     - Heatmap of a co-occurrence matrix for the 25 most frequent items.
     - Distribution plot of basket size and basket total (histogram or ECDF).
     - (Optional) Network graph for top co-occurring pairs.
  5. **Limitations & Next Steps:** Discussion of data limitations, threshold sensitivity, and potential extensions (e.g., confidence/lift, stability checks, temporal or segment analysis).

5.4 Submission Checklist

- `.ipynb` or `.py` files, with `README.md`, exported `.csv` artifacts where required, and a professional PDF report including figures and any other visuals that you want to include.
- Optional: `config.yaml` for parameters; `tests/` directory with basic validation scripts.

5.5 Bonus (up to +5%)

Optional extension tasks such as computing association rules (confidence & lift), bootstrap stability checks, temporal/segment analysis, sparse matrix operations, or automated data-quality reports. Include at least three targeted tests for any extension implemented.

6 Grading Rubric

Component	**Criteria**	**Points**
Code & Reproducibility	Implements Tasks A–F; parameters exposed; deterministic runs; efficient counting; modular structure with clear documentation; produces required outputs.	**50**
Report (with Figures)	Contains design, methodology, findings, and figures integrated into the text; demonstrates understanding and interpretation of results; professional structure and clarity.	**50**
Bonus (extra credit)	Well-implemented optional extension with 3 targeted tests (see Deliverables).	*+5*
	Total	**105**

7 Grade Bands

Grade	GPA	Score Range (%)	Performance Description
A+	4.0	95–100	Exceptional work, complete and polished in all aspects; interpreter, report, and presentation exceed expectations.
A	4.0	90–94	Excellent work, very strong across all components with only minor improvements possible.
A-	3.7	85–89	Very good work, complete with small gaps in detail, clarity, or testing.
B+	3.3	80–84	Good work, most requirements met with some issues in implementation or explanation.
B	3.0	75–79	Satisfactory work, functional but with gaps in coverage or clarity.
B-	2.7	70–74	Adequate work, significant issues in one component (e.g., interpreter or report).
C+	2.3	65–69	Passable work, limited testing or reflection, some incomplete elements.
C	2.0	60–64	Barely sufficient, multiple weaknesses across code, report, or presentation.
C-	1.7	55–59	Weak performance, incomplete work with major flaws but some evidence of effort.
D+	1.3	50–54	Very limited work, serious issues across most components, minimal demonstration of understanding.
D	1.0	40–49	Poor work, interpreter largely non-functional or report missing key sections.
F	0.0	0–39	Unsatisfactory, major requirements missing or not attempted.