AI1030 – Python Programming

Individual Assignment 2

Eldana Ashirova

Market Basket Analysis on Groceries

1 Design & Data Model

1.1 Raw Data Assumptions

- Source file: `groceries.csv`.
- Possible raw schemas:
  (a) One row per basket with comma-separated items, or
  (b) Two columns: `TransactionID, Item` with multiple rows per basket.

1.2 Canonical Transaction Schema

Infer the schema and convert it to a canonical, analysis-friendly form shown in Table 1.

Table 1: Canonical transaction schema.

Column	**Type**	**Description**
`TransactionID`	`int`	Unique identifier per basket (reindexed after cleaning).
`Items`	`list[str]`	Cleaned list of item names per basket.
`Basket_Size`	`int`	Count of items after cleaning.
`Basket_Total`	`float`	Synthetic price sum for items in the basket.

1.3 Auxiliary Structures

- **Price map:** table `product_prices.csv` with columns `item, price`, using a fixed RNG seed in the range [0.50, 15.00].

2 Methodology

2.1 Loading & EDA (Part A)

1. Load `groceries.csv` with pandas.
2. Infer the raw schema; print a compact data dictionary.
3. Perform basic EDA: number of transactions and unique products; basket-size distribution (min/median/95th percentile); top-20 items by frequency.

Figure 1. Top-15 items by frequency.

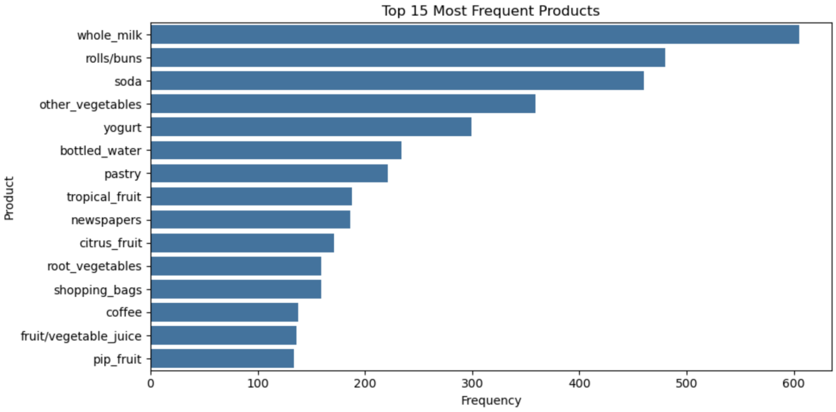

Figure 1: Top-15 items by frequency.

2.2 Cleaning & Transformation (Part B)

- **Standardization:** lowercase, strip whitespace, replace internal spaces by underscores.
- **Invalid removal:** drop empty tokens (e.g., “”, “nan”); normalize duplicates.
- **Basket filter:** drop baskets with fewer than 2 items.
- **Reindex/enrich:** compute `Basket_Size`, reindex `TransactionID`, persist `transactions_clean.csv`.

2.3 Pricing & Enrichment (Part C)

- Build a deterministic price map with a fixed RNG seed (e.g., 42).
- Compute `Basket_Total` by summing per-item prices (assume quantity = 1).
- Persist both `product_prices.csv` and `transactions_priced.csv`.

Figure 2. Distribution of basket totals.

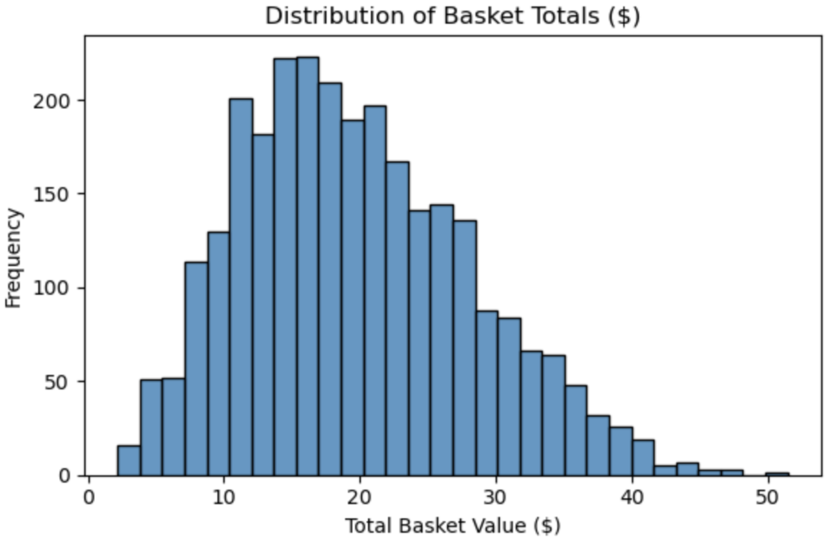

Figure 2: Distribution of basket totals with summary statistics.

2.4 Co-Occurrence Statistics (Part D)

- Generate pairs and triples for unique items per basket using `itertools.combinations`.
- Count with `collections.Counter`.
- Minimum support thresholds (defaults): `min_count_pairs = 20`, `min_count_triples = 5`.
- Compute support count and fraction for each itemset.
- Extract `top_k = 10` pairs and `top_k = 10` triples with deterministic tie-breaks (alphabetical).

Figure 3. Top-k pairs by support fraction.

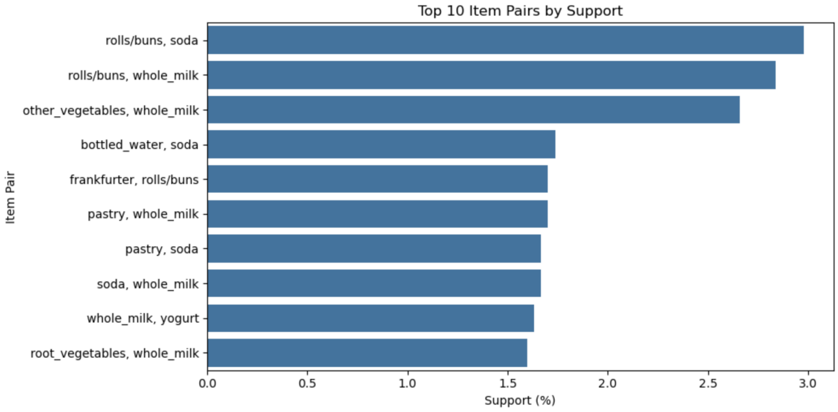

Figure 3: Top-k pairs by support fraction (bar chart).

Figure 4. Top-k triples by support fraction.

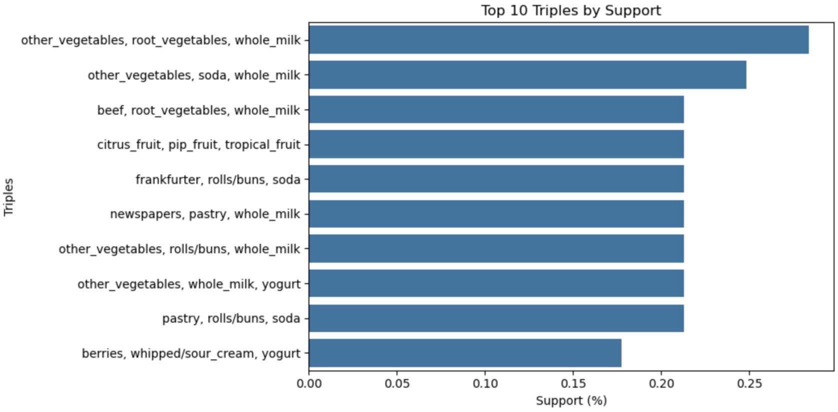

Figure 4: Top-k triples by support fraction (bar chart).

2.5 Visual Analytics (Part E)

- Co-occurrence heatmap for the 25 most frequent items.
- Histogram of `Basket_Size`.

Figure 5. Co-occurrence heatmap (top-25 items).

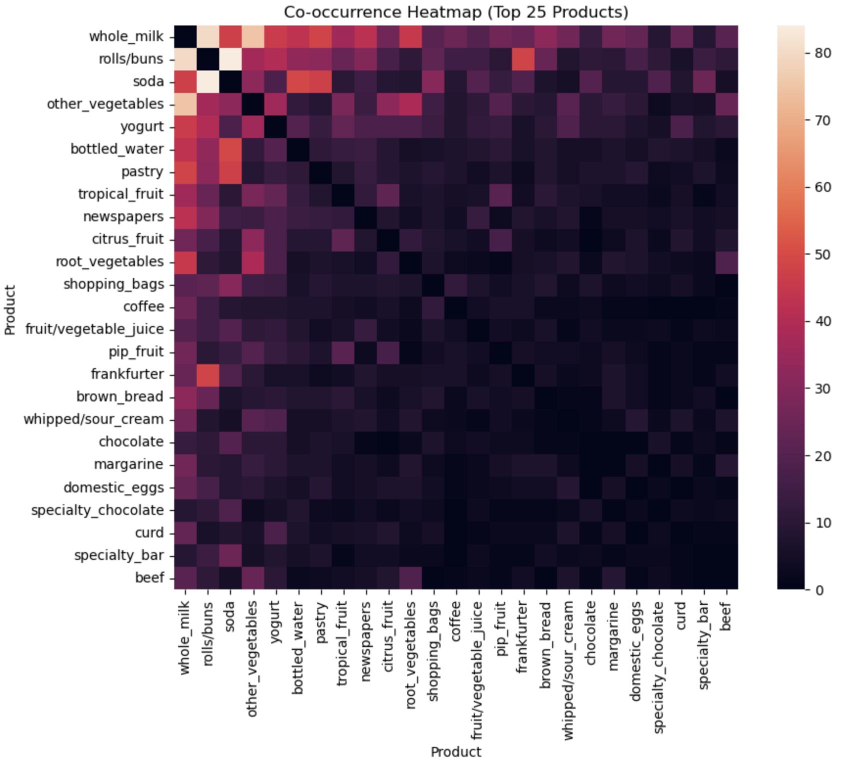

Figure 5: Co-occurrence heatmap for the 25 most frequent items.

Figure 6. Distribution of basket size.

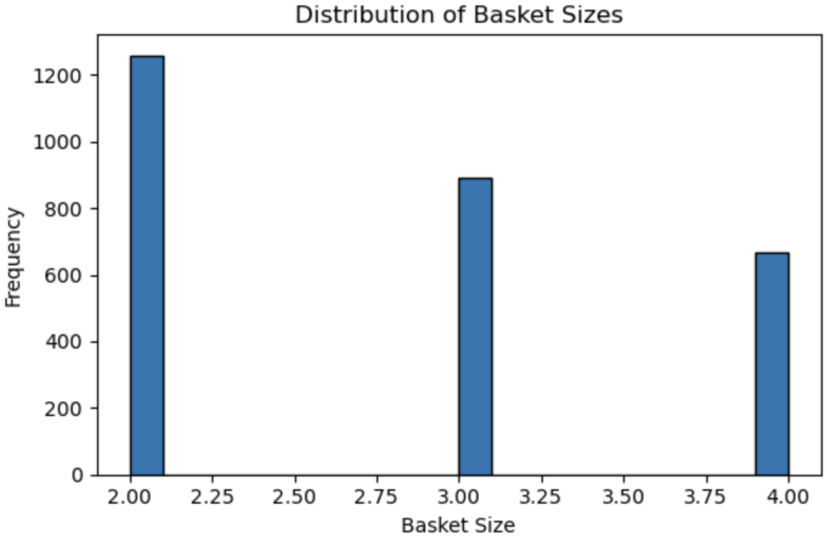

Figure 6: Distribution of basket size.

2.6 Performance & Reproducibility (Part F)

- **Determinism:** All stochasticity controlled via a fixed random seed.
- **Efficiency:** Combination generation only once per basket; counting performed with `Counter`.

- **Parameterization:** thresholds and plotting limits exposed as variables near the top of the notebook; the notebook runs top-to-bottom without manual steps.

Primary Parameters (exact values from the notebook).

- `rng_seed = 42`
- `price_min, price_max = 0.50, 15.00`
- `top_n_items = 15`
- `top_k = 10`
- `min_count_pairs = 20`
- `min_count_triples = 5`

3 Findings & Interpretation

3.1 Basket Structure & Item Popularity

The dataset contains 2819 baskets and 151 unique products. The basket-size distribution is right-skewed with a median around 2 items and a long tail (95th percentile ≈ 4 items). Figure 1 shows a long-tailed item frequency profile typical of retail data. The most frequent items (e.g., whole_milk) suggest staple products that anchor many baskets.

3.2 Pricing & Basket Totals

With synthetic prices in [0.50, 15.00], the `Basket_Total` distribution (Figure 2) is right-skewed. Typical totals fall in 10-30, with outliers reflecting large baskets or concentrations of higher-priced items.

3.3 Co-Occurrence Patterns

Pairs: The top-k pairs (Figure 3) show consistent complementarity (e.g., rolls/buns - soda, rolls/buns - wholemilk). Support fractions indicate meaningful cross-sell potential.
Triples: The top-k triples (Figure 4) reinforce product themes (e.g., produce-oriented or breakfast-oriented groups).
Heatmap: Blocks/stripes in Figure 5 reveal latent product families; dense regions indicate frequently co-purchased groups and can seed planogram decisions or bundle offers.

4 Limitations & Next Steps

Limitations

- Synthetic prices approximate value but do not reflect real profit margins, discounts, or taxation.
- Item multiplicity within baskets is not modeled; co-occurrences are presence/absence counts.
- No temporal segmentation (weekday/weekend, seasonality) or customer-level modeling.

Next Steps (Extensions)

1. **Stability -** Bootstrap baskets (e.g., 20 resamples) and report variability in top-k ranks.
2. **Sparse Structures -** Construct a product×transaction sparse matrix.
3. **Quality Report -** Automated checks for duplicate IDs, malformed rows, and outliers.

5 Reproducibility Checklist

- Single notebook, top-to-bottom execution without manual input.
- Parameters surfaced at the top of the notebook.
- RNG seed fixed.
- Required outputs saved: `product_prices.csv`, `transactions_priced.csv`.
- README documents environment setup and how to run.

6 Figures

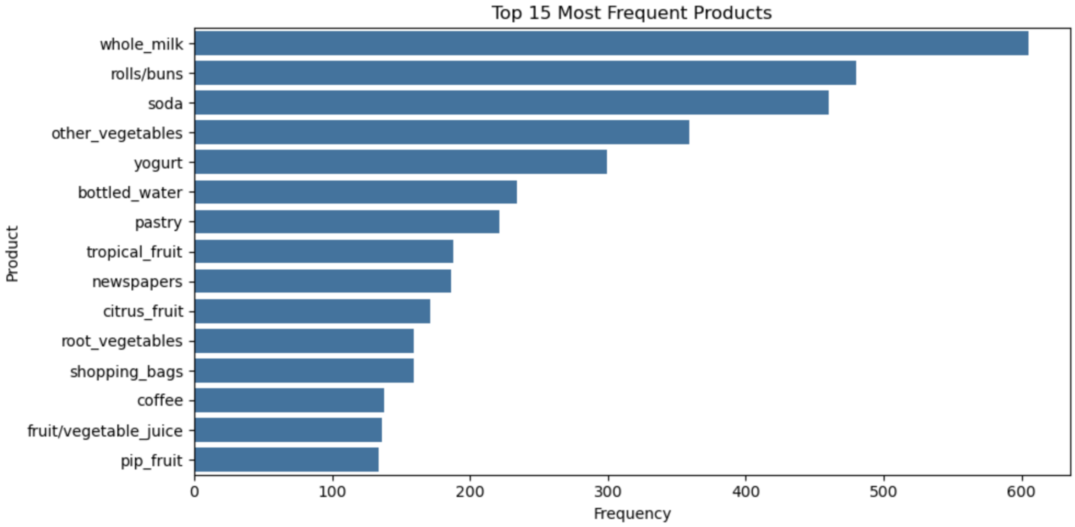

Figure 7: Top-15 individual items by frequency (bar chart).

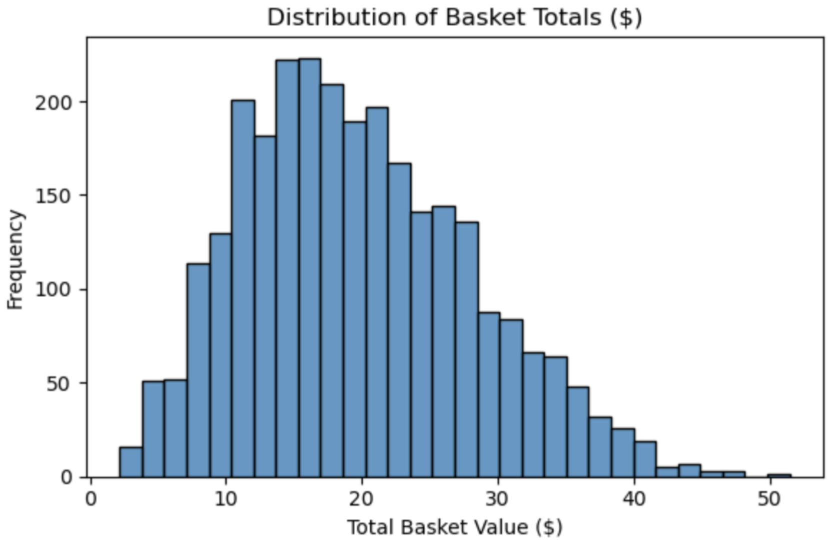

Figure 8: Basket totals.

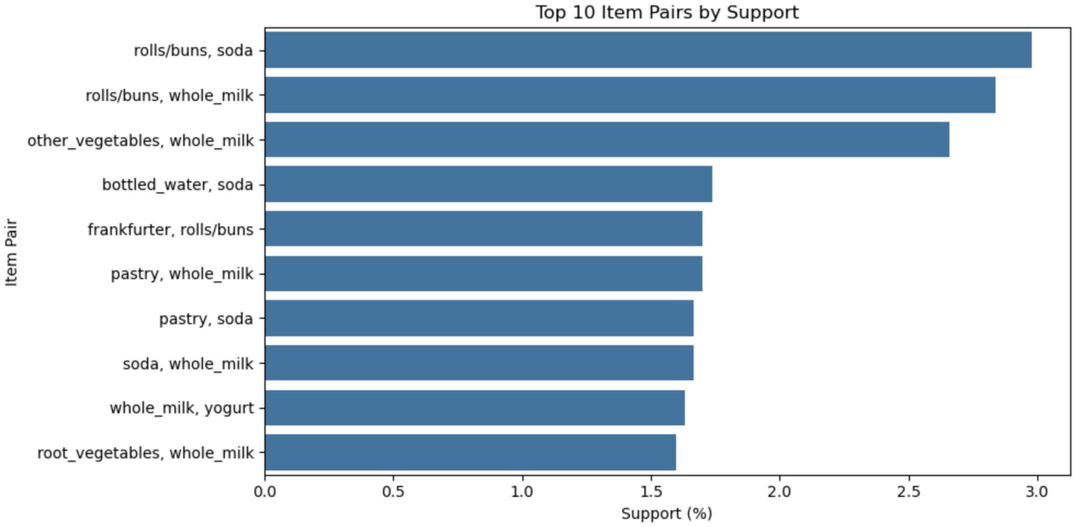

Figure 9: Top-k pairs by support fraction (bar chart).

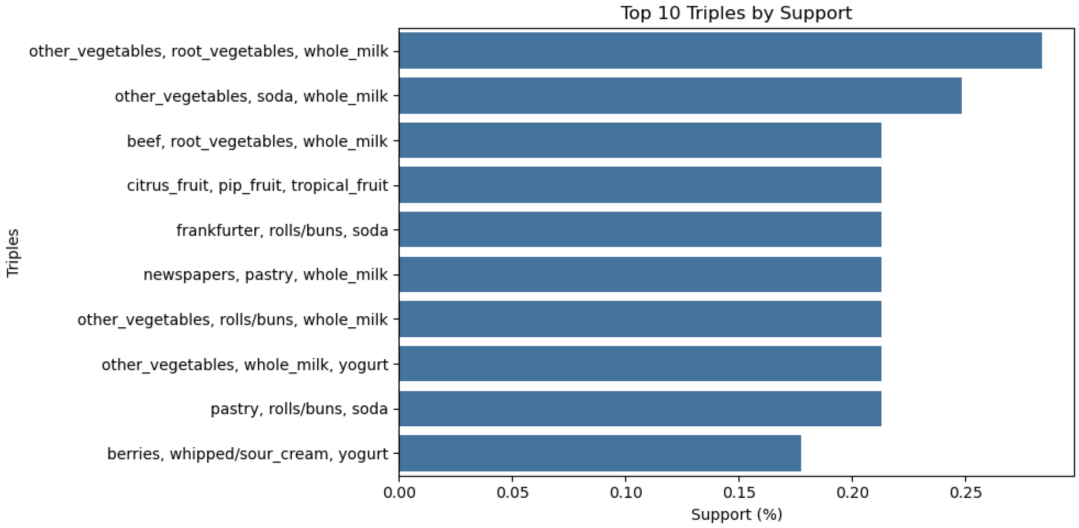

Figure 10: Top-k triples by support fraction (bar chart).

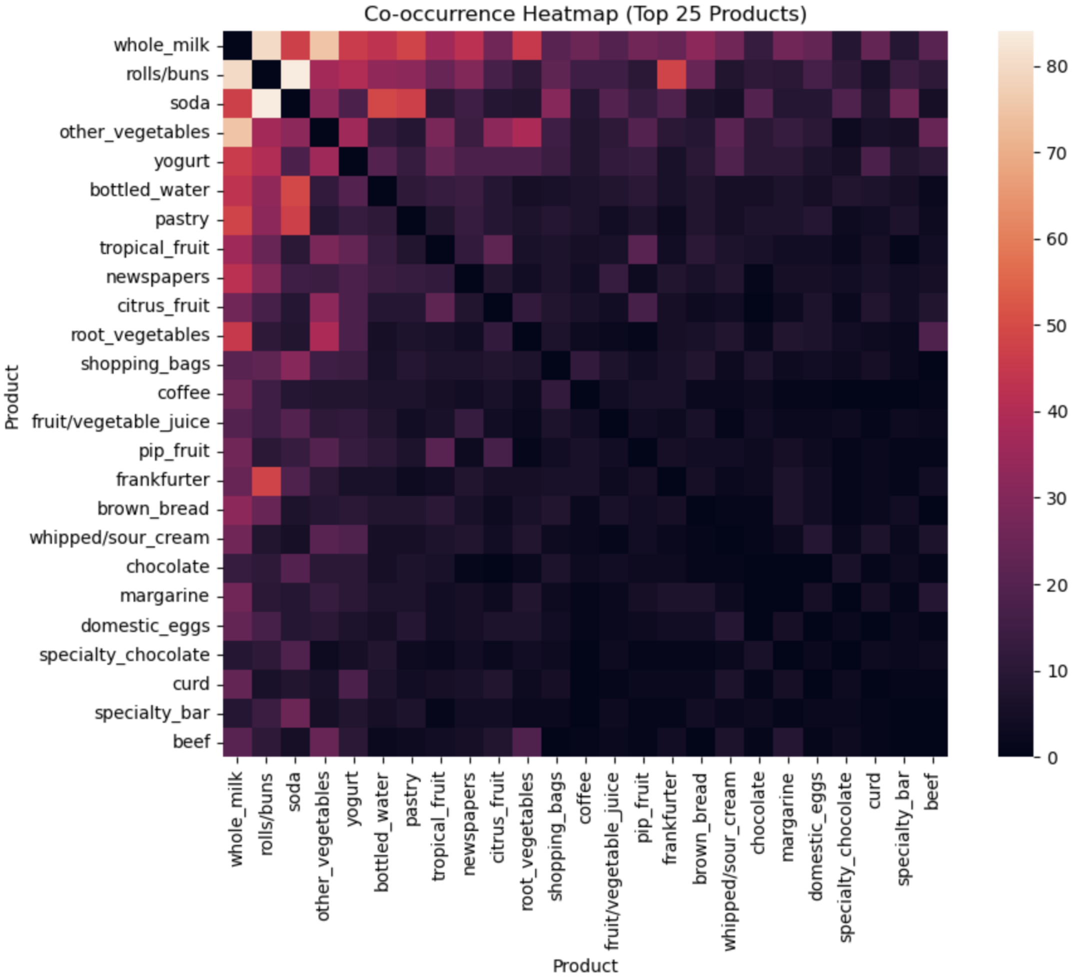

Figure 11: Co-occurrence heatmap for the 25 most frequent items.

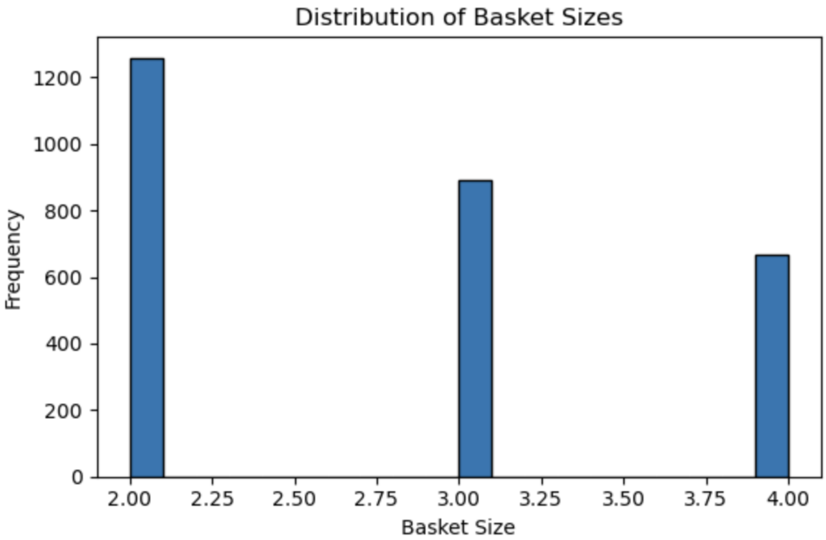

Figure 12: Distribution of basket size.

Declaration of Original Work

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.