SI 201 Foodies - Final Project Report

Group Name: SI201 Foodies
Members:
Vittorio Centore (vcentore@umich.edu), 83197025
Jack Miller (miljack@umich.edu)
Date: December 2025

Project Goals

APIs/Websites Planned

We planned to work with two APIs:

- **Kroger API** (https://api.kroger.com) - To gather grocery product data including prices, brands, and availability
- **Spoonacular API** (https://spoonacular.com/food-api) - To gather recipe and nutrition data including calories, macronutrients, and health scores

Data Collection Goals

Kroger API:

- Collect 100+ grocery products with pricing information
- Store product details (UPC, description, brand)
- Track location-specific data (prices, stock levels, sizes)

Spoonacular API:

- Collect 100+ meal recipes with nutrition information
- Store calories, protein, fat, and carbohydrates for each meal
- Capture cuisine type and health scores
- Record diet labels and ingredient lists

Goals Achieved

APIs Actually Used

Kroger API - Successfully implemented
Spoonacular API - Successfully implemented

Data Actually Collected

Kroger API:

- Collected 125 grocery products from store near ZIP code 48104
- Stored data in 3 related tables:

- products, `items`, `price_history`
- Successfully tracked prices, brands, stock levels, and product sizes
- Implemented foreign key relationship between products and items tables

Spoonacular API:

- Collected 100 meal recipes (20 each for: pasta, chicken, salad, soup, vegetarian)
- Stored complete nutrition data in `meals` table
- Captured all planned fields: calories, macronutrients, cuisine types, health scores
- Successfully parsed ingredients and diet labels

Database Implementation

- Created single SQLite database:
- foodproject.db
- Implemented 4 tables with proper relationships
- Products and items share `product_id` foreign key (avoiding duplicate data)
- Successfully limited data storage to 25 items per execution

Problems Faced

Problem 1: Data Parsing

- Issue: Some Kroger products had missing price data or used "nationalPrice" instead of "price"
- Solution: Implemented fallback logic: `price_obj = item.get("price") or item.get("nationalPrice") or {}`

Problem 2: Unit Conversions

- Issue: Product sizes came in different units (oz, lb, gal, qt, pt, liters)
- Solution: Created conversion functions to normalize all sizes to ounces for fair comparison

Problem 3: Spoonacular API Structure

- Issue: Nutrition data nested deep in JSON response (nutrition → nutrients → array)
- Solution: Used list comprehension with `next()` to extract specific nutrients by name

Calculations from Database

Kroger Calculations:

--- Price Per Unit Calculation ---
Computed price per unit for 105 products

--- Cheapest Item Calculation ---
The cheapest item found out of 105 products is Imperial Sugar Packets ($0.0379)

--- Average Price Calculation ---
Average price of -1 products: 6.29

--- Brand Price Comparison ---
Brand comparison for 31 brands complete.
Boar's Head: Average Price = $14.99
Bob Evans: Average Price = $12.49
Bob's Red Mill: Average Price = $7.73
Cook's: Average Price = $3.66
Cumberland Gap: Average Price = $4.24
Domino: Average Price = $6.39
Double L Ranch: Average Price = $7.49
Eggland's Best: Average Price = $4.54
Fairlife: Average Price = $5.99
Gold Medal: Average Price = $5.51
Happy Egg Co.: Average Price = $8.59
Hatfield: Average Price = $3.29
Horizon Organic: Average Price = $11.1
Imperial Sugar: Average Price = $3.54
In The Raw: Average Price = $5.49
King Arthur Baking Company: Average Price = $8.61
Kodiak: Average Price = $6.99
Kroger: Average Price = $2.84
Lakanto: Average Price = $11.49
Mentos: Average Price = $5.49
Nellie's: Average Price = $4.79
Pete and Gerrys: Average Price = $5.99
Pillsbury: Average Price = $4.99
Pioneer: Average Price = $4.49
Private Selection: Average Price = $7.62
Simple Truth: Average Price = $5.75
Simple Truth Organic: Average Price = $5.34
Smart Way: Average Price = $8.76
Smithfield: Average Price = $8.78
Vital Farms: Average Price = $8.92
Zulka: Average Price = $4.99

Spoonacular Calculations

=== SPOONACULAR NUTRITION ANALYSIS ===

Total meals analyzed: 91

--- Average Nutrition by Cuisine ---
Middle Eastern: Avg Calories: 345.2, Avg Protein: 17.4g, Avg Health Score: 100.0 (1 meals)

European: Avg Calories: 186.5, Avg Protein: 5.6g, Avg Health Score: 100.0 (1 meals)
Cajun: Avg Calories: 392.8, Avg Protein: 18.1g, Avg Health Score: 96.0 (1 meals)
Asian: Avg Calories: 114.1, Avg Protein: 5.3g, Avg Health Score: 86.0 (1 meals)
South American: Avg Calories: 474.9, Avg Protein: 29.0g, Avg Health Score: 85.0 (1 meals)
Mediterranean: Avg Calories: 595.1, Avg Protein: 28.4g, Avg Health Score: 77.7 (3 meals)
Unknown: Avg Calories: 448.0, Avg Protein: 22.3g, Avg Health Score: 76.7 (72 meals)
Chinese: Avg Calories: 306.0, Avg Protein: 19.0g, Avg Health Score: 74.5 (2 meals)
Indian: Avg Calories: 128.7, Avg Protein: 6.7g, Avg Health Score: 71.0 (1 meals)
American: Avg Calories: 609.9, Avg Protein: 36.6g, Avg Health Score: 66.0 (1 meals)
Thai: Avg Calories: 556.7, Avg Protein: 19.7g, Avg Health Score: 63.0 (1 meals)
Mexican: Avg Calories: 426.5, Avg Protein: 28.1g, Avg Health Score: 59.0 (5 meals)
Southern: Avg Calories: 662.5, Avg Protein: 65.6g, Avg Health Score: 47.0 (1 meals)

--- Health Category Distribution ---
Moderately Healthy: 29 meals (31.9%)
Healthy: 37 meals (40.7%)
Less Healthy: 25 meals (27.5%)

--- Top 5 Meals by Nutrition Index ---
1. Slow Cooker Beef Stew (Score: 64.08)
2. Italian Tuna Pasta (Score: 63.25)
3. Colorful Tomato and Spinach Seafood Pasta (Score: 62.93)
4. Moosewood Lentil Soup (Score: 62.59)
5. Red Lentil Soup with Chicken and Turnips (Score: 62.26)

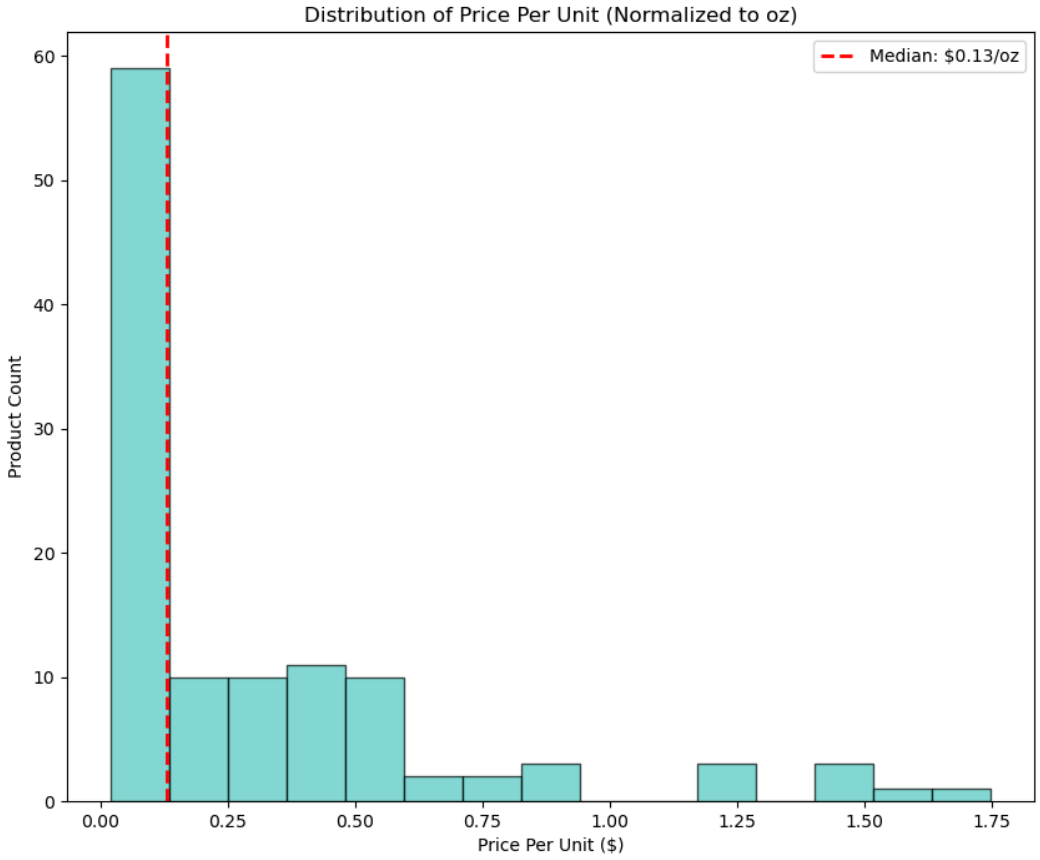
--- Average Macronutrient Breakdown ---
Protein: 19.3%
Fat: 31.1%
Carbs: 49.6%

\

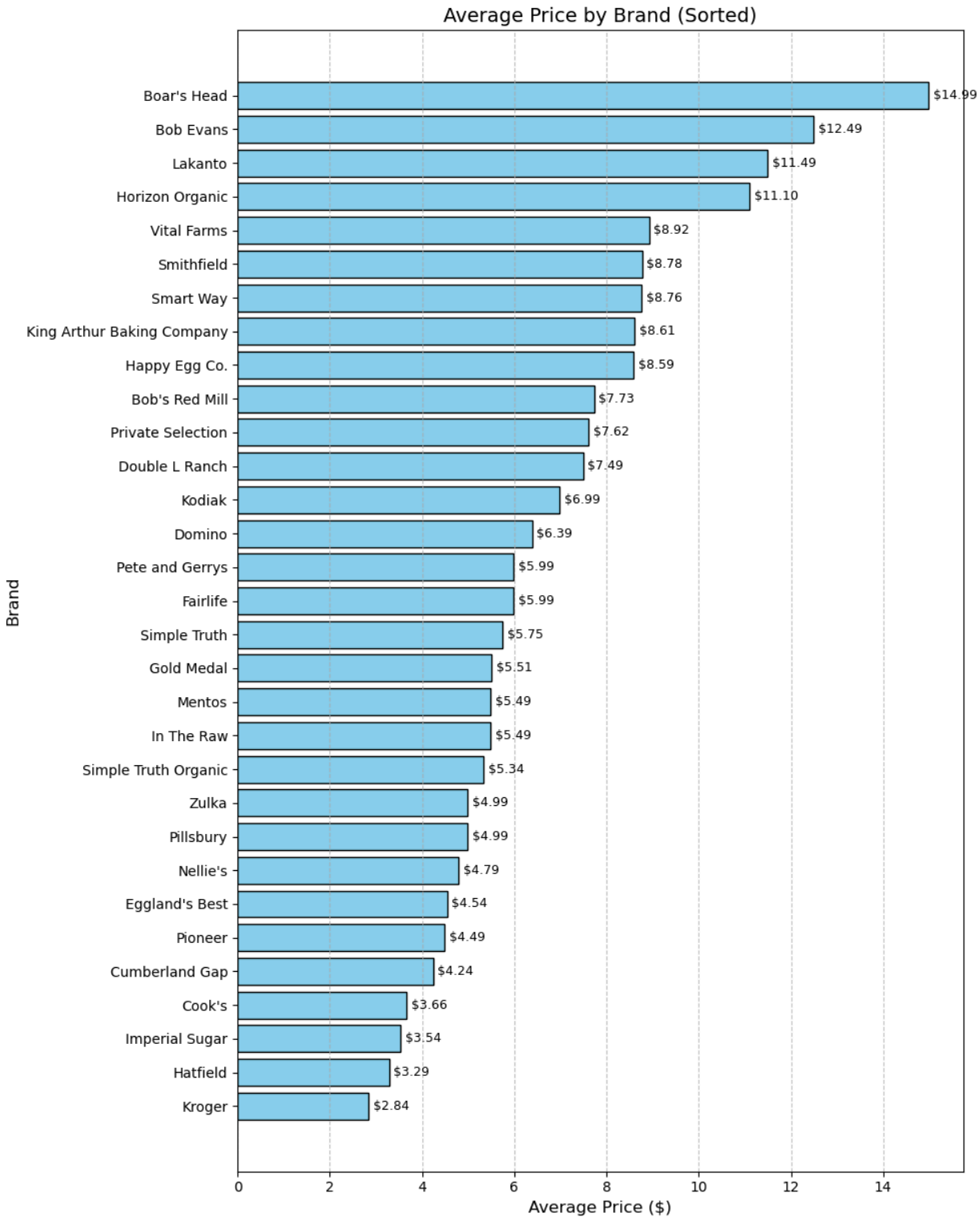
Visualizations Created (After 4 Main.Py Runs)

Kroger Visualizations

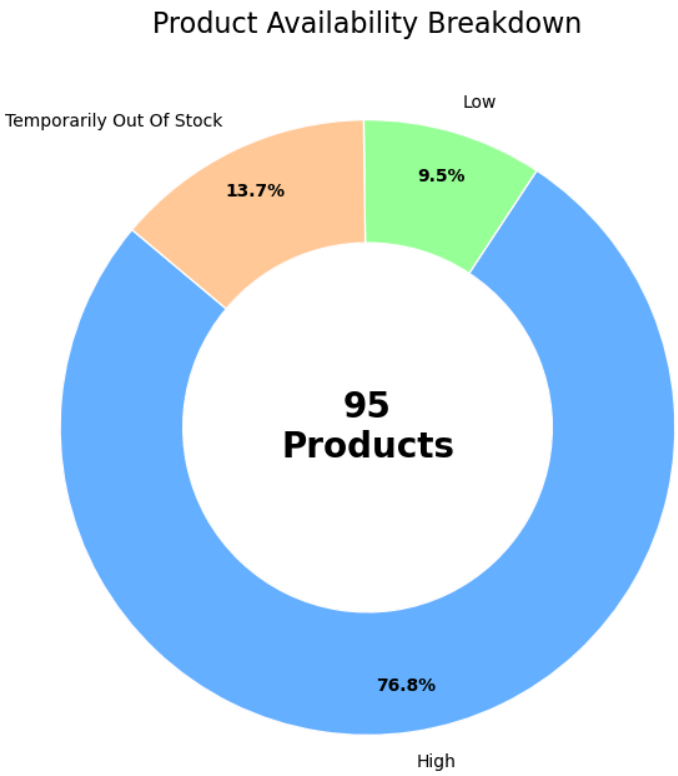
1. **hist_PPU.png** - Histogram showing distribution of price per unit



2. **avgp_brand_bar.png** - Bar chart comparing average prices by brand

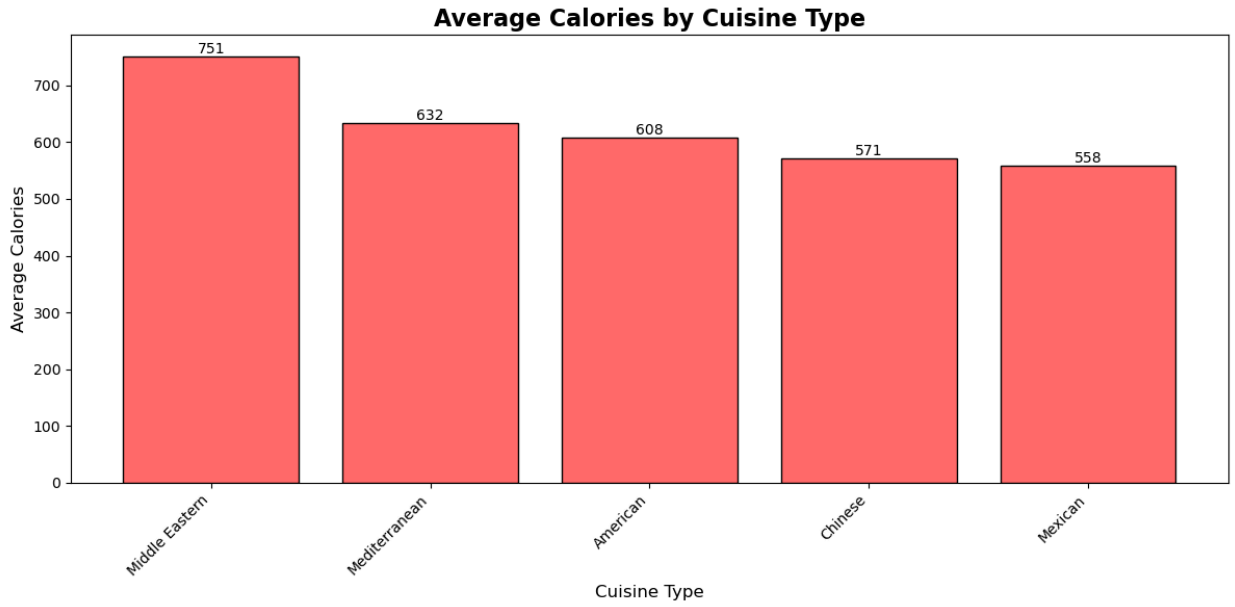


3. **inventory_pie.png** - Pie chart showing product availability levels

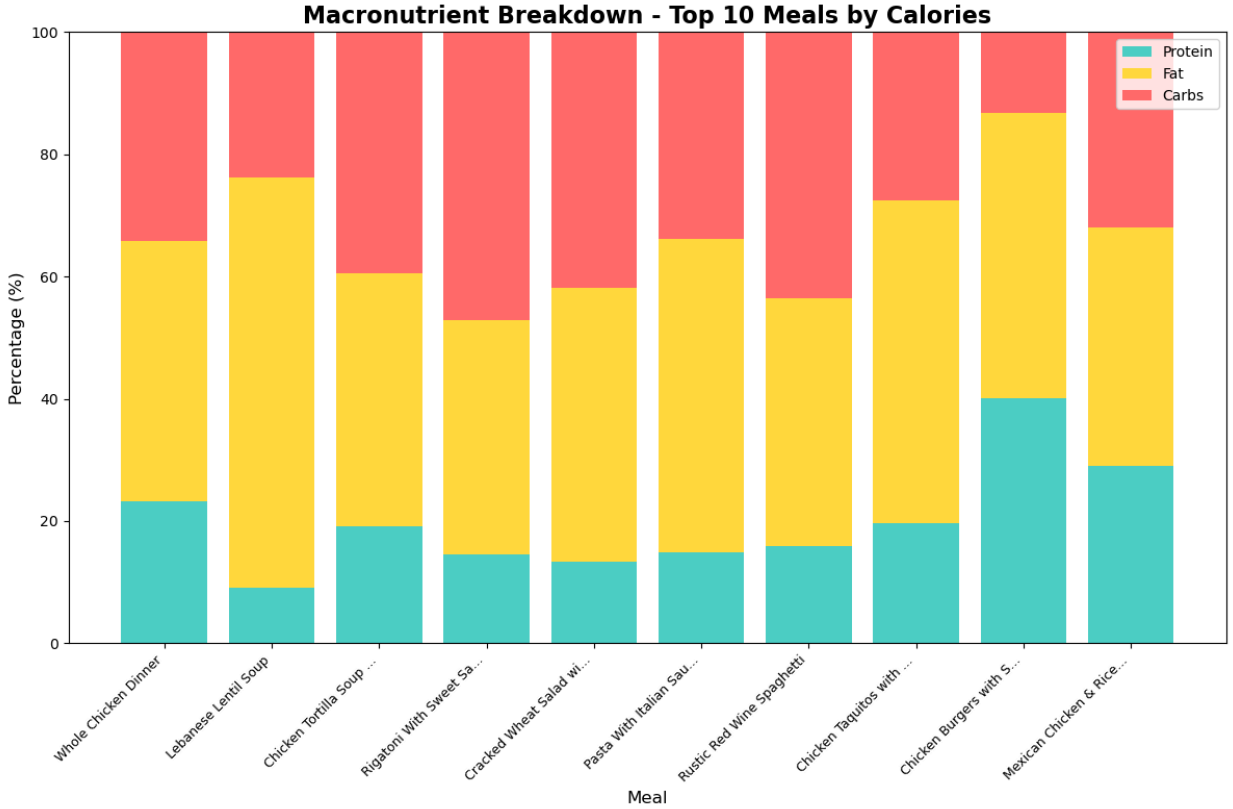


Spoonacular Visualizations

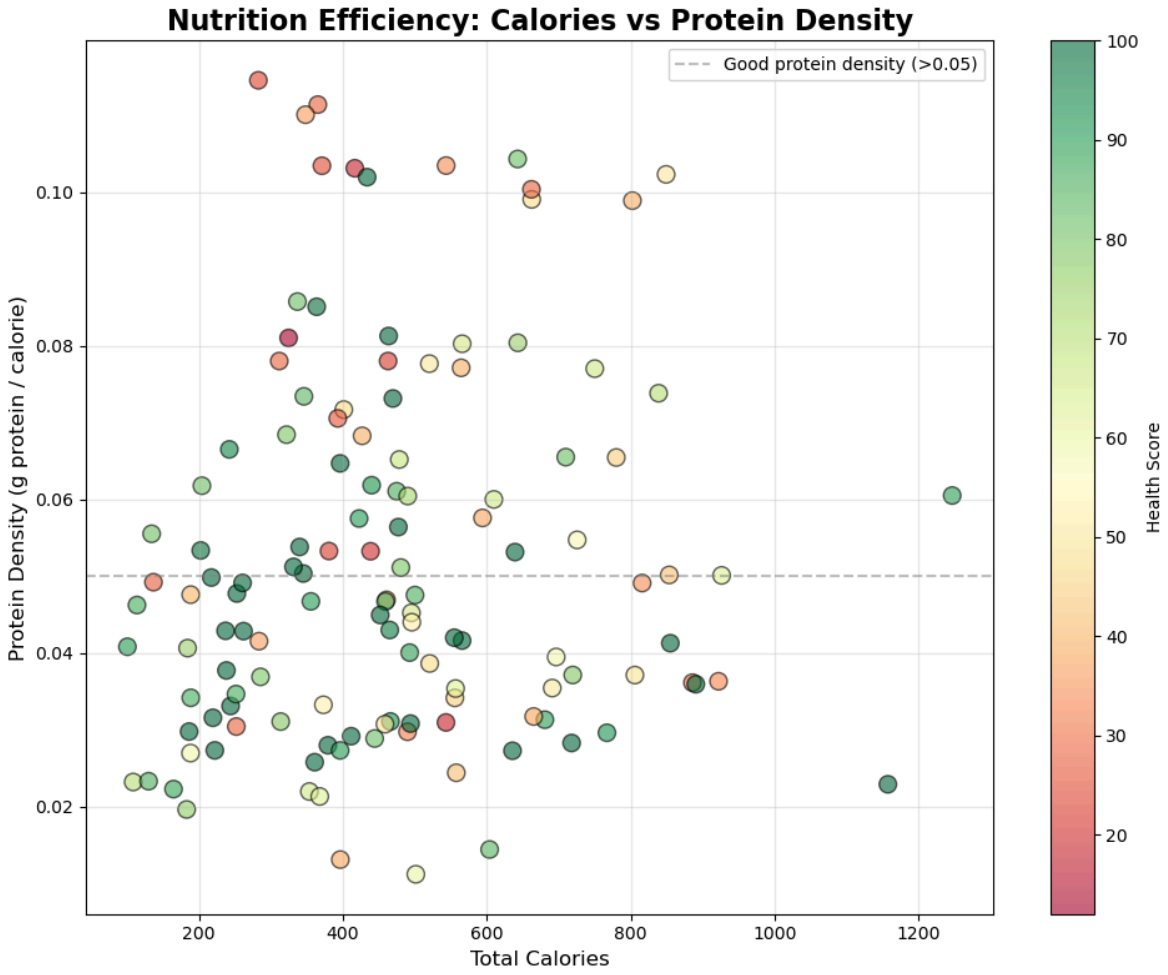
4. **calories_by_cuisine.png** - Bar chart of average calories by cuisine type



5. **macronutrient_breakdown.png** - Stacked bar chart of protein/fat/carbs for top 10 meals



6. **calories_vs_protein_density.png** - Scatter plot showing nutrition efficiency



Instructions for Running Code

Prerequisites

- Python 3.7 or higher
- Spoonacular API key (free tier: 150 requests/day)
- Kroger Developer ID and Secret, Free

Setup Steps

1. **Create virtual environment**
    - python3 -m venv venv
    - source venv/bin/activate
2. **Install dependencies**
    - pip install -r requirements.txt
3. **Configure Spoonacular and Kroger API Key**
    - Get API key from https://spoonacular.com/food-api/console#Dashboard
    - Kroger from https://developer.kroger.com/manage/apps

        i. Note, you need to sign up for an email account, and create an application.
            1. This is how you get the ID and Secret key to use.
        ii. Select “production” environment and select “locations”, “cart”, and “products” as public
    - Calcs_and_sql.py
    - Line 9 and 10: Replace “CLIENT_ID” and “CLIENT_SECRET” with your actual application ID and Secret
    - Line 13: Replace `"YOUR_API_KEY_HERE"` with your actual key
4. **Run the program**
    - python main.py

Output Files

After running, you'll find:

- foodproject.db - SQLite database with all data
- kroger_results.txt - Kroger analysis results
- `spoonacular_results.txt` - Spoonacular analysis results
- 6 PNG visualization files

Function Diagram and Team ContributionS

Work Division

- **Jack Miller**: All Kroger API functions (authentication, data fetching, database operations, calculations)
- **Vittorio Centore**: All Spoonacular API functions (data fetching, parsing, database operations, calculations)
- **Both**: Visualization functions, main pipeline integration, testing

Function Diagram

main.py - main()
├── Input: None
├── Output: None (creates output files)

└── Author: Both

KROGER API FUNCTIONS (Jack):
├── create_kroger_tables()
│   ├── Input: None
│   └── Output: Creates 3 tables in database
├── get_kroger_token(client_id, secret, url)
│   ├── Input: API credentials
│   └── Output: OAuth2 access token
├── get_kroger_loc(token, zipcode)
│   ├── Input: Access token, ZIP code
│   └── Output: Store location ID
├── fetch_kroger_products(token, location_id, term, limit)
│   ├── Input: Token, location, search term, limit
│   └── Output: List of raw product dictionaries
├── clean_and_transf_kroger(raw_products)
│   ├── Input: Raw API data
│   └── Output: Cleaned product dictionaries
├── into_krogerdb(products, location_id)
│   ├── Input: Cleaned products, location ID
│   └── Output: Number of items inserted
├── save_price_hist()
│   ├── Input: None
│   └── Output: Saves price history to database
└── kroger_calculations()
    ├── Input: None
    └── Output: Writes kroger_results.txt

SPOONACULAR API FUNCTIONS (Vittorio):
├── create_spoonacular_table()
│   ├── Input: None
│   └── Output: Creates meals table in database
├── fetch_meal_data_spoonacular(api_key, query, number)
│   ├── Input: API key, search term, limit
│   └── Output: List of raw meal dictionaries
├── clean_and_transform_meal_data(raw_meals)
│   ├── Input: Raw API data
│   └── Output: Cleaned meal dictionaries
├── into_spoonacular_db(meals)
│   ├── Input: Cleaned meals
│   └── Output: Number of meals inserted
└── spoonacular_calculations()
    ├── Input: None
    └── Output: Writes spoonacular_results.txt

VISUALIZATION FUNCTIONS (Both):
├── price_per_unit()
│ ├── Input: None (reads from database)
│ └── Output: hist_PPU.png
├── brand_avg_price()
│ ├── Input: None (reads from database)
│ └── Output: avgp_brand_bar.png
├── pie_inventory()
│ ├── Input: None (reads from database)
│ └── Output: inventory_pie.png
├── calories_by_cuisine()
│ ├── Input: None (reads from database)
│ └── Output: calories_by_cuisine.png
├── macronutrient_breakdown()
│ ├── Input: None (reads from database)
│ └── Output: macronutrient_breakdown.png
└── calories_vs_protein_density()
  ├── Input: None (reads from database)
  └── Output: calories_vs_protein_density.png

8. Resources Used

Date	Issue Description	Location of Resource	Result
12/01/2024	How to use Kroger API	https://developer.kroger.com/api-products	Solved - understood API structure
12/01/2024	OAuth2 authentication	https://requests.readthedocs.io/en/latest/	Solved - implemented Base64 encoding
12/02/2024	Spoonacular API documentation	https://spoonacular.com/food-api/docs	Solved - learned to use complexSearch endpoint
12/02/2024	How to parse nested JSON	https://stackoverflow.com/questions/14048948	Solved - used .get() with fallbacks

12/03/2024	SQLite foreign keys	https://www.sqlitetutorial.net/sqlite-foreign-key/	Solved - created proper relationships
12/04/2024	Matplotlib stacked bar chart	https://matplotlib.org/stable/gallery/lines_bars_and_markers/bar_stacked.html	Solved - created macronutrient visualization
12/05/2024	Unit conversion formulas	Wikipedia	Solved - implemented oz conversion
12/06/2024	Database JOINs in SQLite	https://www.sqlitetutorial.net/sqlite-join/	