Synthetic Retail Sales Generation and Economic Data Integration

# Execution Instructions

**Prerequisites:**  
Run the following in a Jupyter Notebook or Google Colab cell to install required libraries:

|  |  |
| --- | --- |
| * Fredapi * Faker |  |

**Required Libraries:**

|  |  |
| --- | --- |
| * Pandas * Numpy * Random * Datetime * Fredapi * OS * Logging * Faker |  |

**Run Environment:**Recommended: Google Colab or Jupyter Notebook  
Local Python (.py script) is also supported

**File Configuration:**

* Update the DATA\_PATH in the script to a folder you can write to:  
   DATA\_PATH = '../content/'
* The script generates and stores:
  + A master dataset (US\_Historical\_Sales.csv)
  + Monthly sales files (e.g., US\_Monthly\_Sales\_Month\_2025.csv)
  + A log file: data\_check\_log.txt

# Overview of the Approach

This script simulates monthly retail sales data for 50 synthetic Shampoo products in the US. It includes both product-level dynamics and macroeconomic context, such as gasoline prices and CPI.  
  
**The process is:**

1. **Initialization:**
   1. If no master file exists, it generates synthetic weekly sales data for Jan–Dec 2024.
   2. Merges with weekly average gas prices and CPI from the FRED API.
   3. Saves as baseline master dataset.
2. **Monthly Update Logic:**
   1. Checks for already existing monthly sales files.
   2. If the current month is already processed, exits gracefully.
   3. Otherwise, generates new month’s data and updates both monthly and master files.
3. **Economic Indicators:**
   1. Gas Prices (GASREGW): Weekly averages.
   2. Consumer Price Index (CPIAUCSL): Weekly forward-filled values.

# Assumptions Made

|  |  |
| --- | --- |
| **Assumption** | **Description** |
| First-Time Run Logic | If US\_Historical\_Sales.csv is missing, generate full-year data for 2024 |
| Monthly Data Logic | From Jan 2025 onward, monthly files are generated one at a time |
| Product Setup | 50 synthetic products with fixed IDs and generic 'Variant' naming |
| Pricing | Prices randomly range from $4–$15 with variable discounts (0–20%) |
| Discount Impact | Higher discounts increase units\_sold using a weighted multiplier |
| Seasonality | Summer (Jun–Aug): 20% sales boost; Holidays (Nov–Dec): 50% sales boost |
| Date Handling | Weekly data aligned to Sundays (W-SUN frequency) |
| Economic Data | Downloaded from FRED only for necessary periods and merged weekly |
| Reprocessing | Avoids regenerating months already processed; only appends new |
| Logging | All updates and warnings logged in data\_check\_log.txt and printed to console |

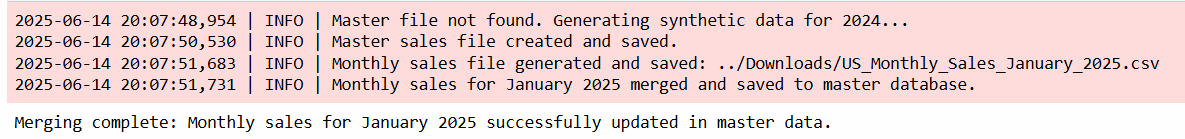
# Code Comments & Logic Highlights

Code is modular with function-level separation:

* generate\_synthetic\_sales(): Creates weekly sales with discount and seasonality
* fetch\_latest\_economic\_indicators(): Pulls CPI and Gas data from FRED
* merge\_sales\_with\_economics(): Merges economic data into sales DataFrame
* main(): Manages initialization, monthly logic, file updates, and logging

All critical steps are commented to explain logic and purpose.

* In the first run, the historical data file for the year 2024 is generated, along with the data for January 2025, including avg\_gas\_price and cpi values.



* On the second run, data for February 2025 is generated.



* On the third run, data for March 2025 is generated.





* On the fourth run, data for April 2025 is generated.



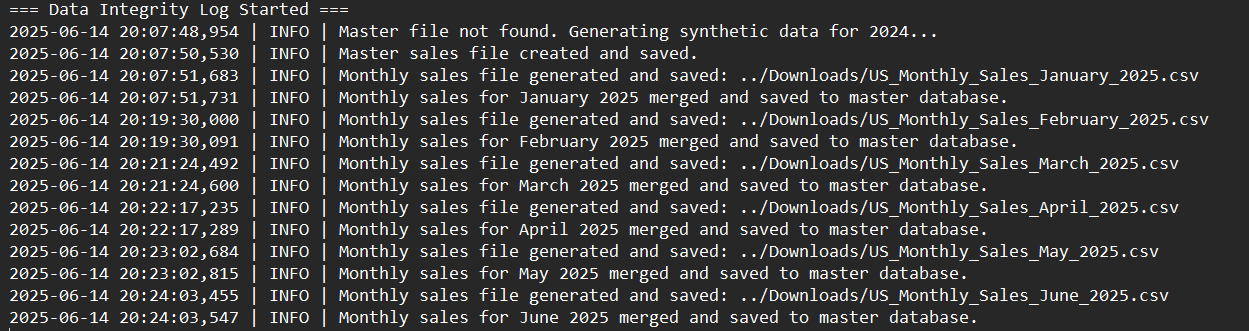


* On the fifth run, data for May 2025 is generated.



* On the sixth run, data for June 2025 is generated.



* Running the code again after the current month will not generate new data.
* Log file captures all processing details.

# Final Note on Execution Logic

There are two primary scenarios handled in the code:

Scenario 1: No Master File Exists

* Generates synthetic data for Jan–Dec 2024.
* Adds economic context and saves as master dataset.

Scenario 2: Master File Exists

* Identifies last generated month.
* If monthly data is current, exits without processing.
* Otherwise, generates, merges, and updates files.

This ensures automated monthly updates without duplication or reprocessing, making it production-ready for scheduled pipelines.