# **Portfolio Performance Generator**

Developer Documentation

## **Overview**

This Python module generates portfolio performance for Target Date Funds. The generated dataset can be used to populate the AST\_MULTIASSET\_DB.DBO.PortfolioPerformance table. For real-world practicality and computational efficiency, performance is calculated at the product (fund ticker) level. The module derives daily gross and net returns for all fund tickers, assigns each portfolio a randomized inception date, and returns daily performance from that inception date through today.

## **Table of Contents**

1. Dependencies
2. Setup and Configuration
3. Data Flow
4. Module Components
5. 4.1 Performance Factor Generation
6. 4.2 Random Inception Date Utility
7. 4.3 Portfolio Performance Generation
8. 4.4 Main Execution Block
9. Key Algorithms and Design Decisions
10. Data Structure
11. Error Handling
12. Testing Considerations
13. Future Enhancements
14. Developer Checklist

## **1. Dependencies**

The module relies on the following libraries:  
- pandas: Data manipulation and DataFrame operations  
- numpy: Numerical operations (optional)  
- yfinance: Historical price data for fund tickers  
- yahooquery: Fund profile (expense ratio) and metadata  
- datetime / random: Date arithmetic and randomized inception dates

## **2. Setup and Configuration**

• The script maintains an internal list of Target Date Fund tickers and a global performance inception date.  
• Reproducibility is controlled by a random seed parameter in the main function.  
• The function `generate\_portfolio\_performance` requires access to holding data via `from HoldingDetails\_Table import get\_df\_merged`.

## **3. Data Flow**

1) Download daily price history for all fund tickers from a fixed inception date to today.  
2) Retrieve each fund’s expense ratio to compute net returns (gross less pro‑rated expense ratio).  
3) Build a performance reference table (product-level, daily, gross and net).  
4) Load merged holdings to enumerate portfolios and their associated fund tickers.  
5) Assign each portfolio a randomized inception date.  
6) Join portfolio positions to the performance reference table.  
7) Filter rows to dates on/after each portfolio’s inception date.  
8) Return the final Portfolio Performance table.

## **4. Module Components**

### **4.1 Performance Factor Generation**

Function: generate\_performance\_factors()  
Purpose: Computes product-level daily gross and net return series for all fund tickers.  
Logic:  
• Gross return: Close(t) / Close(t-1) − 1  
• Net return: Gross − (expense\_ratio × days/365)  
Output: DataFrame with columns FUND TICKER, PERFORMANCEINCEPTIONDATE, HISTORYDATE, PERFORMANCETYPE ('Portfolio Gross' / 'Portfolio Net'), PERFORMANCEFACTOR.

### **4.2 Random Inception Date Utility**

Function: random\_date(start, end)  
Purpose: Generates a realistic, randomized inception date per portfolio account within a specified window.  
Output: A `date` object in [start, end].

### **4.3 Portfolio Performance Generation**

Function: generate\_portfolio\_performance(start\_date, end\_date, seed)  
Purpose: Produces the final portfolio performance dataset.  
Inputs:  
• start\_date, end\_date: Window used to draw randomized inception dates.  
• seed: Random seed for reproducibility.  
Process:  
1) Load holdings via `get\_df\_merged()` and identify (PORTFOLIOCODE, FUND TICKER) pairs.  
2) Build a left table with currency, frequency ('D'), and a randomized `PORTFOLIOINCEPTIONDATE` per pair.  
3) Join to the performance factors table from `generate\_performance\_factors()` on FUND TICKER.  
4) Select standardized columns and filter to rows where PERFORMANCEINCEPTIONDATE ≥ PORTFOLIOINCEPTIONDATE.  
Output: DataFrame `df\_portfolio\_performance`.

### **4.4 Main Execution Block**

When the script is executed directly, it calls `generate\_portfolio\_performance()` and prints the resulting table. Export to CSV can be added if needed for downstream ingestion.

## **5. Key Algorithms and Design Decisions**

• Product-Level Aggregation: Performance is computed at the fund ticker level for operational simplicity and speed.  
• Net Return Calculation: Expense ratios are pro‑rated by the exact day count between observations.  
• Inception Logic: Each portfolio receives a randomized inception date to simulate real-world onboarding.  
• Daily Frequency: PERFORMANCEFREQUENCY is 'D' (daily) for compatibility with other tables.

## **6. Data Structure**

COLUMN TYPE DESCRIPTION  
PORTFOLIOCODE String Portfolio identifier  
HISTORYDATE Date Performance observation date  
CURRENCYCODE String Currency code (e.g., 'USD')  
CURRENCY String Currency name (e.g., 'US Dollar')  
PERFORMANCECATEGORY String Category level (e.g., 'Asset Class')  
PERFORMANCECATEGORYNAME String Category label (e.g., 'Total Portfolio')  
PERFORMANCETYPE String 'Portfolio Gross' or 'Portfolio Net'  
PERFORMANCEINCEPTIONDATE Date Previous observation date used for the return  
PORTFOLIOINCEPTIONDATE Date Randomized portfolio inception date  
PERFORMANCEFREQUENCY String Frequency code ('D')  
PERFORMANCEFACTOR Float Daily return

## **7. Error Handling**

• Missing Prices: Tickers without valid price history are skipped.  
• Empty Slices: Empty groups are ignored when computing returns.  
• Defaults: Expense ratio defaults to 0.0 when unavailable.

## **8. Testing Considerations**

• Validate that gross/net return math matches expectations for sample tickers.  
• Confirm reproducibility by fixing the random seed and comparing outputs.  
• Test varying inception date windows to ensure correct filtering.  
• Check column ordering and data types for downstream compatibility.

## **9. Future Enhancements**

• Add currency translation from holding currency to portfolio base currency.  
• Parameterize performance frequency (e.g., monthly, weekly).  
• Support benchmark-relative active returns.  
• Persist directly to Snowflake or a data lake with schema validation.

## **10. Developer Checklist**

✓ Install required dependencies  
✓ Verify Yahoo Finance access for both prices and fund profiles  
✓ Ensure `get\_df\_merged()` is available and returns expected columns  
✓ Set a random seed for reproducible tests  
✓ Validate output schema and date filtering logic