# **Benchmark Performance Generator**

Developer Documentation

## **Overview**

This Python module generates benchmark performance for Target Date Funds using real-world market data. The generated dataset can be used to populate the AST\_MULTIASSET\_DB.DBO.BenchmarkPerformance table. We select GSPC (Equity) and AGG (Bond) as benchmark codes. Unlike other tables, BenchmarkPerformance is the only one where data is pulled directly from live market sources into the dataset, resulting in a slightly different and more streamlined workflow.

## **Table of Contents**

1. Dependencies
2. Setup and Configuration
3. Data Flow
4. Module Components
5. 4.1 Benchmark Performance Retrieval
6. 4.2 Build Benchmark Performance
7. Key Algorithms and Design Decisions
8. Data Structure
9. Error Handling
10. Testing Considerations
11. Future Enhancements
12. Developer Checklist

## **1. Dependencies**

The module relies on the following libraries:  
- pandas: Data manipulation and DataFrame operations  
- yfinance: Historical market price data retrieval  
- datetime: Date parsing and formatting

## **2. Setup and Configuration**

• Requires internet access to retrieve benchmark prices from Yahoo Finance.  
• Benchmark tickers are specified in the script (e.g., 'GSPC', 'AGG').  
• Frequency parameter defaults to daily ('D'), but can be adjusted.  
• Returned DataFrame matches Snowflake table schema for direct ingestion.

## **3. Data Flow**

1) Define benchmark ticker(s), start date, end date, and frequency.  
2) Fetch adjusted close prices via yfinance.  
3) Resample prices if non-daily frequency is requested.  
4) Format dates as both date strings and timestamp strings to match database schema.  
5) Append metadata columns (benchmark code, performance data type, currency, frequency).  
6) Return DataFrame for downstream processing or direct database upload.

## **4. Module Components**

### **4.1 Benchmark Performance Retrieval**

Function: get\_benchmark\_performance(benchmark\_ticker, start\_date, end\_date, frequency='D')  
Purpose: Fetches benchmark price data from Yahoo Finance and shapes it to match the Snowflake BenchmarkPerformance table schema.  
Output: DataFrame with columns:  
BENCHMARKCODE, PERFORMANCEDATATYPE, CURRENCYCODE, CURRENCY, PERFORMANCEFREQUENCY, VALUE, HISTORYDATE1, HISTORYDATE.  
Notes:  
• Automatically adjusts prices for splits/dividends.  
• Handles empty datasets by returning an empty DataFrame with correct column headers.

### **4.2 Build Benchmark Performance**

Function: build\_benchmark\_performance()  
Purpose: Calls get\_benchmark\_performance() within this script to produce the BenchmarkPerformance table.  
Usage: Enables users to preview the table structure directly from this script before uploading to Snowflake.

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

• Direct Data Pull: This is the only performance table sourced directly from real-time market data.  
• Schema Alignment: Output DataFrame is structured exactly as required by the Snowflake table.  
• Flexible Frequency: Supports daily or aggregated performance intervals.  
• Simplicity: Focused solely on benchmark price history without synthetic data generation.

## **6. Data Structure**

COLUMN TYPE DESCRIPTION  
BENCHMARKCODE String Benchmark identifier (e.g., 'GSPC', 'AGG')  
PERFORMANCEDATATYPE String Type of performance data (always 'Prices')  
CURRENCYCODE String Currency code (e.g., 'USD')  
CURRENCY String Currency name (e.g., 'US Dollar')  
PERFORMANCEFREQUENCY String Performance interval code ('D' for daily)  
VALUE Float Benchmark price value  
HISTORYDATE1 String Date string (YYYY-MM-DD)  
HISTORYDATE String Timestamp string (YYYY-MM-DD HH:MM:SS)

## **7. Error Handling**

• Empty DataFrames: If yfinance returns no data for a ticker/date range, an empty DataFrame is returned with proper columns.  
• Date Formatting: Ensures both date and timestamp fields are present and formatted.  
• Graceful Fallback: Avoids runtime errors when data is unavailable.

## **8. Testing Considerations**

• Verify data retrieval for known benchmark tickers over a given date range.  
• Test different frequency settings and confirm correct resampling.  
• Confirm schema matches Snowflake table exactly.  
• Validate currency and metadata fields are correctly populated.

## **9. Future Enhancements**

• Add support for multiple benchmarks in a single call.  
• Integrate retry logic for transient network failures.  
• Enable configurable currency codes and automatic FX conversion.  
• Allow alternate market data providers for redundancy.

## **10. Developer Checklist**

✓ Install required dependencies (pandas, yfinance)  
✓ Verify internet connectivity for market data retrieval  
✓ Confirm benchmark tickers and date ranges before running  
✓ Validate output schema matches Snowflake table  
✓ Test different frequency values if needed