**Benchmark Characteristics Generator**

**Developer Documentation**

# Overview

This module generates a Benchmark Characteristics Table by retrieving key financial metrics for given benchmarks using the Yahoo Finance API (yfinance). It calculates statistics such as P/E ratios, dividend yields, returns, market cap, and more for each benchmark, storing them in a structured DataFrame format suitable for database ingestion.

# Table of Contents

* Overview
* Table of Contents
* Dependencies
* Setup and Configuration
* Data Flow
* Module Components
  + 4.1 Benchmark Characteristics Generation Function
  + 4.2 Main Usage Pattern
* Key Design Decisions
* Data Structure
* Error Handling
* Testing Considerations
* Future Enhancements
* Developer Checklist

# 1. Dependencies

**External libraries:**- pandas  
- yfinance  
- python-dateutil  
  
**Standard library:**  
- time  
- logging  
- datetime  
- typing  
  
**Install with:**  
pip install pandas

pip install time

pip install logging

pip install yfinance

pip install datetime

pip install typing

# Setup and Configuration

Logging is configured at the INFO level for tracking execution steps. Display formatting disables scientific notation in Pandas for float values. Constants define available characteristics, ETF proxies for S&P 500, delays between requests, and currency name mapping. An optional benchmark map dictionary is required to link benchmark codes to their Yahoo Finance tickers.

# Data Flow

1) Accept a dictionary mapping BENCHMARKCODE → Yahoo Finance ticker.

2) For each benchmark:

a. Combine the primary ticker with S&P 500 ETF proxies (SPY, IVV, VOO).

b. Fetch general benchmark info from Yahoo Finance.

c. Determine currency code and name.

d. Compute characteristics using helper functions.

e. If characteristic is "# of Securities" and benchmark is S&P 500 (gspc), fetch constituent count.

3) Append each characteristic as a separate row in the output DataFrame.

4) Delay execution between benchmarks to avoid API rate limits.

5) Return the DataFrame for downstream use.

## 4.1 Benchmark Characteristics Generation Function

Function:  
build\_benchmark\_characteristics\_table(benchmark\_map: dict[str, str]) -> pd.DataFrame  
  
**Purpose:**  
Generates a characteristics table for provided benchmarks by fetching live market data and computing a predefined list of metrics.  
  
**Process:**  
- Loop through each benchmark in the input dictionary.  
- Append ETF proxies to the ticker list.  
- Retrieve currency info and resolve currency name.  
- Compute metrics including P/E (TTM & forward), P/B, P/S, dividend yield, EPS metrics, ROE, returns, and market cap.  
- Format results into a row per characteristic with metadata fields.  
  
**Data Sources:**  
- Yahoo Finance API via yfinance library.  
  
**Output Structure:**  
- DataFrame with columns: BENCHMARKCODE, CURRENCYCODE, CURRENCY, LANGUAGECODE, CATEGORY, CATEGORYNAME, CHARACTERISTICNAME, CHARACTERISTICDISPLAYNAME, STATISTICTYPE, CHARACTERISTICVALUE, ABBREVIATEDTEXT, HISTORYDATE.

## 4.2 Main Usage Pattern

No main() wrapper — module is typically called directly:  
from benchmark\_characteristics import build\_benchmark\_characteristics\_table  
df\_char = build\_benchmark\_characteristics\_table({"gspc": "^GSPC"})  
print(df\_char)

# 5. Key Design Decisions

* ETF proxies are used as fallback for missing S&P 500 benchmark data.
* Delay between API calls prevents Yahoo Finance throttling.
* Caching of yfinance.Ticker.info reduces repeated API calls.
* Flexible characteristic computation — each metric has a dedicated helper function.
* Return format is normalized for database insertion.

# 6. Data Structure

- BENCHMARKCODE (str) – Internal code (e.g., "gspc").

- CURRENCYCODE (str) – ISO currency code (e.g., "USD").

- CURRENCY (str) – Full currency name.

- CHARACTERISTICNAME (str) – Metric name (e.g., "Price/Earnings (TTM)").

- CHARACTERISTICVALUE (float/int) – Metric value.

- HISTORYDATE (date) – Date of calculation.

# 7. Error Handling

- API call failures return None for that metric.

- Missing values are skipped from final output.

- ETF proxy fallback is attempted before giving up on a metric.

# 8. Testing Considerations

Unit Testing

✓ Validate column schema of returned DataFrame.

✓ Confirm each helper function computes metrics correctly.

✓ Mock Yahoo Finance calls to avoid network dependency.

Integration Testing

✓ Run against multiple benchmarks to ensure consistent formatting.

✓ Verify "# of Securities" is correct for S&P 500 benchmarks.

# 9. Future Enhancements

- Add support for non-equity benchmarks.

- Parallelize API calls for faster execution.

- Add configurable characteristic list.

- Store raw API responses for auditability.

# 10. Developer Checklist

✓ Dependencies installed.

✓ API reachable.

✓ Outputs validated.

✓ Metric computations tested.