**Portfolio General Information Generator**

**Developer Documentation**

# Overview

This module programmatically generates a Portfolio General Information table for 10 synthetic portfolios. It dynamically discovers Vanguard Target Retirement mutual funds from the public Vanguard site, derives portfolio base currency from Yahoo Finance metadata, and infers a realistic open date (and performance inception date) by taking the earliest inception date among a random sample of 4 target-date funds per portfolio. Outputs are aggregated into a single pandas DataFrame for downstream loading.

# Table of Contents

* Overview
* Table of Contents
* Dependencies
* Setup and Configuration
* Data Flow
* Module Components
* 4.1 Portfolio General Information Generation
* 4.2 Main Execution Function
* Key Design Decisions
* Data Structure
* Error Handling
* Testing Considerations
* Future Enhancements
* Developer Checklist

# Dependencies

The module relies on the following external libraries:

* pandas
* yfinance
* Babel (for currency names)
* Python standard library: random, datetime.date, io.StringIO
* (Optional) requests
* (Optional) tenacity
* (Optional) IPython.display

Ensure these dependencies are installed:

pip install pandas

pip install requests

pip install tenacity

pip install yfinance

pip install Babel

# Setup and Configuration

No specific runtime configuration is required. The module can be executed as a standalone Python script. (Optional) Ticker list and portfolio count may be changed in the function arguments.

# Data Flow

1. Seed RNG and set up constants (portfolio count, product codes, target-date fund tickers)
2. For each portfolio: sample K tickers from target-date funds

3) Fetch base currency (from Yahoo Finance) and currency name (via Babel with fallback)

4) Determine earliest inception date among sampled funds or fallback to today

5) Fill static attributes (category, style, flags, termination date)

6) Assign a unique product code to each portfolio

7) Aggregate rows into a single DataFrame

# 4. Module Components

## 4.1 Portfolio General Information Generation

**Function:** generate\_portfolio\_general\_information()

**Purpose:** Builds a complete “Portfolio General Information” table for synthetic target-date fund portfolios by dynamically pulling fund metadata and assembling consistent portfolio records.

**Process:**

* Build product code list (PRD001…PRD012), seed RNG, and define Vanguard Target Retirement ticker list
* For each portfolio:
* - Create PORTFOLIOCODE and NAME
* - Sample K tickers from target\_date\_funds
* - For first sampled ticker: fetch currency from Yahoo Finance, map to BASECURRENCYNAME via Babel (fallback to code)
* - For all sampled tickers: fetch fundInceptionDate, determine earliest date → set OPENDATE and PERFORMANCEINCEPTIONDATE (fallback: today)
* - Assign static attributes (category, style, flags, termination date)
* - Assign unique PRODUCTCODE
* Append each row to results list
* Return DataFrame (optional CSV export)

**Data Sources:**

* Vanguard Target Retirement ticker list (static)
* Yahoo Finance Ticker.info fields: currency, fundInceptionDate

**Output Structure:**

* One row per portfolio with identity, currency, date, classification, and flags

## 4.2 Main Execution Function

**Function:** main()

**Purpose:** Orchestrates the generation of the Portfolio General Information table and handles final assembly/output.

**Process:**

* Call generate\_portfolio\_general\_information()
* Validate DataFrame schema and row count
* Return DataFrame (optional CSV export)

# Key Design Decisions

* Dynamic metadata retrieval from Yahoo Finance for currency and inception dates
* Earliest inception date rule for realistic portfolio start dates
* Babel currency mapping with fallback to code
* Deterministic sampling via fixed RNG seed
* Simplified scope focused on Portfolio General Information only
* Static classification fields for style, category, and performance flag

# Data Structure

* PORTFOLIOCODE (str) — unique portfolio identifier (e.g., PORT001)
* PRODUCTCODE (str) — associated product identifier (e.g., PRD001)
* NAME (str) — portfolio display name
* BASECURRENCYCODE (str) — ISO currency code from Yahoo Finance
* BASECURRENCYNAME (str) — friendly name from Babel
* OPENDATE (date) — earliest sampled fund inception date or today
* PERFORMANCEINCEPTIONDATE (date) — mirrors OPENDATE
* TERMINATIONDATE (None) — open portfolios
* PORTFOLIOCATEGORY (str) — 'Individual Account'
* INVESTMENTSTYLE (str) — 'Growth'
* ISBEGINOFDAYPERFORMANCE (bool) — True

# Error Handling

* Missing Yahoo Finance fields → fallback to safe defaults
* Babel mapping failures → fallback to BASECURRENCYCODE
* Validate random sample size (k <= available tickers)
* Network/API errors propagate to caller for handling

# Testing Considerations

**Unit Testing**

* ✓ Verify DataFrame has expected row count and schema
* ✓ Confirm correct data types for each column
* ✓ Ensure OPENDATE and PERFORMANCEINCEPTIONDATE logic is correct
* ✓ Validate currency code and name mapping (with fallback)
* ✓ Check PRODUCTCODE uniqueness

**Integration Testing**

* ✓ Mock Yahoo Finance responses to test fallback paths
* ✓ Confirm deterministic output with fixed RNG seed
* ✓ Verify optional CSV export contains correct structure and values

# Future Enhancements

* Parameterize portfolio count and K sample size
* Dynamically fetch ticker list from Vanguard
* Cache/batch Yahoo Finance calls
* Validate currency consistency across sampled funds
* Extend to Holdings, Performance Factors, and Benchmark Associations
* Add export options for CSV, Parquet, and database

# 10.Developer Checklist

* ✓ All dependencies installed
* ✓ Ticker list validated and accessible
* ✓ Function calls produce expected DataFrame schema
* ✓ Optional CSV export tested
* ✓ Error handling paths tested with mock data
* ✓ Code reviewed for adherence to naming and style guidelines