Tab 1

Comprehensive Code Descriptions

Real Estate Team - BU MSBA Capstone Project - Summer 2025

BA888 - Shiv Nag, Courtney Vincent, Zicheng Wang

This document provides an overview of the Python scripts that make up the automated Real Estate fund data pipeline built for Assette. Each script plays a distinct role in sourcing, transforming, and populating structured data into Snowflake tables used for downstream analytics and reporting. The design is modular and scalable, with each component handling a discrete task in the data flow.

Benchmark Characteristics

This table stores calculated and descriptive characteristics for REIT benchmarks, supporting analytics, fact sheets, and investor reporting. Each record links a benchmark, identified by BENCHMARKCODE, to a specific characteristic on a given HISTORYDATE. The dataset captures both quantitative metrics—such as volatility, rolling returns, and drawdowns—and qualitative descriptors, such as category names and display labels, through fields like CHARACTERISTICNAME and CHARACTERISTICDISPLAYNAME. CURRENCYCODE and CURRENCY ensure all values are clearly tied to a monetary standard, while LANGUAGECODE supports localization for client deliverables. CATEGORY and CATEGORYNAME group characteristics into logical sets for easier reporting and filtering in downstream tools. STATISTICTYPE defines how the value should be interpreted (e.g., point-in-time value, average), while CHARACTERISTICVALUE stores the metric result as a standardized string for compatibility across systems. ABBREVIATEDTEXT provides an optional shortened form for tight layouts in dashboards and printed materials. The HISTORYDATE field ensures temporal accuracy, allowing time-series analysis of benchmark attributes. This dataset is populated via scripts that pull benchmark data from Yahoo Finance and other sources, calculate key performance metrics, and format them for Snowflake ingestion. Validation routines check for missing values, extreme outliers, and proper category-to-characteristic mappings before upload. Benchmark characteristics are essential for providing context to raw performance data, enabling portfolio managers, analysts, and clients to interpret results in terms of risk, return consistency, and market positioning. Without this table, performance reporting would lack the supplemental metrics needed for robust analysis and decision-making.

| Column | Data Type | Description |
| --- | --- | --- |
| BENCHMARK CODE | String | Unique code identifying the benchmark |
| CURRENCY CODE | String | Currency code used |
| CURRENCY | String | Full currency name |
| LANGUAGE CODE | String | Langauge used (eg EN) |
| CATEGORY | String | Category code |
| CATEGORY NAME | String | Full name of the category |
| CHARACTERISTIC NAME | String | Name of the characteristic |
| CHARACTERISTIC DISPLAY NAME | String | Display name for the characteristic |
| STATISTIC TYPE | String | Statistic type |
| CHARACTERISTIC VALUE | String | Value |
| ABBREVIATED TEXT | String | N/A |
| HISTORY DATE | Date | Date of the characteristic Record |

Benchmark General Information

This script manages the creation and population of the **Benchmark General Information (BGI)** table, which contains essential metadata about benchmark ETFs used to evaluate Real Estate portfolio performance. Benchmarks like VNQ, IYR, and SCHH are defined in a structured pandas DataFrame, with attributes such as symbol, name, and a flag indicating if performance is calculated at the beginning of the day. The data can be manually defined or loaded from a YAML configuration file, offering flexibility for both static and scalable ingestion patterns.

The script includes a validation function to ensure required fields are present, symbols are valid, and the data types match expectations. Once validated, the data is uploaded into a staging table in Snowflake using the write\_pandas utility, then merged into the permanent BenchmarkGeneralInformation table.

This automated process supports data accuracy, maintainability, and repeatability. It ensures downstream analytics tools have a clean, consistent set of benchmarks for performance attribution, comparisons, and reporting. It also lays the groundwork for dynamic benchmark updates via integration with APIs like Yahoo Finance.

The modular architecture supports future enhancements, such as adding inception dates, validating live ticker status, or expanding to support non-Real Estate benchmark categories. Overall, this file serves as a critical foundation for aligning portfolios with appropriate market benchmarks in Assette’s Real Estate analytics pipeline.

| BENCHMARKCODE | VARCHAR | Unique internal identifier for the benchmark (e.g., REITBENCH01). **Primary Key** |
| --- | --- | --- |
| NAME | VARCHAR | Full name or title of the benchmark fund or index. |
| SYMBOL | VARCHAR | Market ticker symbol used for the benchmark (e.g., VNQ, IYR, SCHH). |
| ISBEGINOFDAYPERFORMANCE | BOOLEAN | Indicates if benchmark performance is measured from beginning-of-day prices. |

Benchmark Performance

This script powers the ingestion and normalization of the **Benchmark Performance** (BP) table data for Real Estate funds using Yahoo Finance as a data source. It begins by retrieving Real Estate benchmark codes and symbols from the BenchmarkGeneralInformation table in Snowflake, then uses the yfinance library to fetch historical daily price data. Each benchmark's price history is normalized to start at a value of 100, creating a standardized total return index over time. Where available, inception dates are extracted from metadata; otherwise, the earliest available price date is used as a proxy. The resulting performance data is enriched with metadata such as currency, frequency, and performance data type.

Validation logic ensures key fields are present, flags outliers, and checks for proper performance type classification before upload. The script then uploads data to a Snowflake staging table and uses a deduplicated MERGE operation to populate the BenchmarkPerformance table reliably. Timestamp and date fields are carefully handled to preserve accuracy and compatibility with downstream systems. Logging is used to trace errors, missing values, or anomalies during validation.

This file is critical for ensuring Assette’s Real Estate benchmarks have accurate, complete, and normalized historical performance data. The dataset it produces underpins performance reporting, analytics, and visualizations in client dashboards and investor materials.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| BENCHMARKCODE | VARCHAR | Unique identifier for the benchmark |
| PERFORMANCEDATATYPE | VARCHAR | Type of performance data, e.g., "Total Return" |
| CURRENCY | VARCHAR | The currency name in which performance values are denominated |
| CURRENCYCODE | VARCHAR | The ISO currency code, e.g., "USD" |
| PERFORMANCEFREQUENCY | VARCHAR | Frequency of performance data, e.g., "Daily" |
| VALUE | FLOAT | Normalized price value indexed to 100 at the start date |
| HISTORYDATE | TIMESTAMP | The inception date of the benchmark (date corresponding to data start) |
| HISTORYDATE1 | DATE | HISTORYDATE as a plain date (without time component) |

Holdings Details

This file generates a fully synthetic REIT (Real Estate Investment Trust) holdings dataset designed for ingestion into Snowflake’s data warehouse. It defines a strict column structure to replicate real-world portfolio holdings, including financial, geographic, and classification metadata. Each row represents a security within a portfolio, including attributes like ticker, market value, book value, risk country, and sector classifications. The script randomly assigns values while maintaining realistic ranges and types — for example, prices and quantities are generated and multiplied to compute market value. Ticker symbols are cross-referenced against Yahoo Finance to fetch actual prices, ensuring realism. It also includes fallback logic for unavailable tickers and logs any skipped entries (e.g., those missing historical prices).

This dataset is structured to match the column expectations of Snowflake tables, particularly Holdings\_Details, and is used to test downstream integration with other Snowflake-linked tables such as Security\_Master, Portfolio\_Map, or performance metrics. All column names are uppercased and normalized for compatibility with Snowflake's ingestion processes. The file ensures referential integrity by including synthetic yet consistent keys like PORTFOLIOCODE, ISINCODE, and TICKER. It plays a foundational role in simulating the asset-level structure of a REIT portfolio, allowing for broader system testing, performance validation, and analytics pipeline development.

Ultimately, this notebook enables rapid prototyping and automation of realistic financial holdings data, supporting use cases like fact sheet generation, client reporting, or dashboard visualization within the Snowflake-based Assette ecosystem.

Portfolio Benchmark Association

This script automates the mapping of Real Estate portfolios to their appropriate benchmarks using portfolio and benchmark metadata from Snowflake. It begins by retrieving active Real Estate portfolio codes and names from the PortfolioGeneralInformation table, then pulling benchmark codes and names from the BenchmarkGeneralInformation table.

The script compares portfolio names to benchmark names using both exact and keyword-based matching to identify the most relevant associations. Each match is assigned a RANK to indicate primary, secondary, or additional benchmark relevance for that portfolio. The resulting association data is validated to ensure all required fields are present, ranks are valid, and no duplicate portfolio–benchmark combinations exist.

Validated results are uploaded to a Snowflake staging table using write\_pandas() and merged into the PortfolioBenchmarkAssociation table via a deduplicated MERGE operation.

This process programmatically maintains accurate portfolio–benchmark linkages, eliminating the need for manual mapping and reducing the risk of human error. By standardizing benchmark assignments, it ensures consistency across performance analytics, reporting pipelines, and comparative analysis workflows.

The resulting data enables accurate benchmarking in Assette’s dashboards, fact sheets, and internal analytics tools. Careful handling of validation and merging ensures data integrity and compatibility with downstream systems. This file is critical to maintaining scalable, reliable, and repeatable benchmark association logic that supports Assette’s Real Estate fund performance measurement and reporting capabilities.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| PORTFOLIOCODE | VARCHAR | Unique identifier for the portfolio (from PortfolioGeneralInformation). |
| BENCHMARKCODE | VARCHAR | Unique identifier for the benchmark (from BenchmarkGeneralInformation). |
| RANK | NUMBER | Ranking of the benchmark for the given portfolio (1 = primary, 2 = secondary, etc.). |
| RECIPIENTCODE | VARCHAR | Optional field for external recipient logic; currently always NULL. |

Portfolio General Information

This file builds and loads the **Portfolio General Information (PGI)** table, which stores structured metadata for synthetic REIT portfolios. It begins by classifying REIT tickers into investment styles using external sources like SEC, REITNotes, and Nareit. The script merges this classification data into a standardized format, resolving discrepancies to assign final styles like “Equity,” “Mortgage,” or “Hybrid.” It then defines five synthetic portfolios, associating each with a name, style, product code, open date, and base currency. These records are assembled into a DataFrame (pgi\_df) formatted for ingestion into Snowflake.

To ensure data quality, the code performs **validation checks** on investment styles, logs any anomalies, and enforces rules to maintain clean and usable data. Once validated, the script creates a temporary staging table (STG\_PGI) and uses a SQL MERGE operation to update or insert rows into the target PortfolioGeneralInformation table. The pipeline also supports **modular scripts** (classify\_styles.py, generate\_pgi.py, validate.py, main.py) for better scalability and separation of concerns.

This PGI pipeline is critical because it acts as the foundation for linking performance, benchmark, and product-level data within Assette’s platform. By programmatically populating and maintaining portfolio metadata, it enables automated reporting, consistent analytics, and robust data lineage. The modular structure allows for quick adaptation to new asset types or business needs, while detailed logging makes it easy to troubleshoot or audit. Overall, it ensures that the platform has a reliable and extensible source of truth for portfolio-level information.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| PORTFOLIOCODE | VARCHAR | Unique code identifying the portfolio |
| NAME | VARCHAR | Human-readable portfolio name, typically describing investment focus |
| INVESTMENTSTYLE | VARCHAR | Describes the investment style/category (e.g., Equity, Hybrid) |
| PORTFOLIOCATEGORY | VARCHAR | High-level category classification of the portfolio (e.g., REIT) |
| OPENDATE | DATE | The date when the portfolio was officially opened |
| PERFORMANCEINCEPTIONDATE | DATE | The date from which performance measurement begins |
| BASECURRENCYCODE | VARCHAR | ISO currency code representing the portfolio's base currency |
| BASECURRENCYNAME | VARCHAR | Full name of the portfolio’s base currency |
| ISBEGINOFDAYPERFORMANCE | BOOLEAN | Flag indicating if performance is measured at the beginning of day |
| PRODUCTCODE | VARCHAR | Code representing the product grouping the portfolio belongs to |
| TERMINATIONDATE | DATE | Date the portfolio was terminated or null if active |

Portfolio Performance

This table captures the calculated historical performance of Assette’s simulated REIT portfolios. Each record represents a portfolio’s performance on a specific date, storing key attributes such as portfolio code, currency, performance category, performance type, and inception dates. The data is generated by portfolio performance scripts that aggregate ticker‑level returns, apply market‑cap weighting, and simulate different return types (gross, net, and model net of fees). HISTORYDATE is recorded in daily frequency to allow for granular performance tracking, trend analysis, and benchmark comparisons. PERFORMANCECATEGORY and PERFORMANCECATEGORYNAME fields classify the portfolio at the asset class level, enabling consistent grouping in dashboards and reports. PERFORMANCETYPE distinguishes between calculation methods, which is essential for accurate fee modeling and investor reporting. PERFORMANCEINCEPTIONDATE and PORTFOLIOINCEPTIONDATE ensure downstream processes can filter results based on portfolio history and relevance. The PERFORMANCEFACTOR stores the normalized return value, which is validated to reject extreme or erroneous results before upload. All values are currency‑coded for consistency, with CURRENCYCODE and CURRENCY aligning with standardized reference data. The dataset is designed for Snowflake ingestion and serves as a core input to Assette’s client‑facing performance dashboards, fact sheets, and attribution reports. This table is critical because it provides a consistent, validated, and complete view of portfolio performance, ensuring accuracy in analytics, client communications, and compliance reporting.

| Column | Data Type | Description |
| --- | --- | --- |
| PORTFOLIO CODE | String | Unique code identifying the portfolio (1-5) |
| HISTORY DATE | Date | Date of the performance record |
| CURRENCY CODE | String | Currency Code (eg USD) |
| CURRENCY | String | Currency Name |
| PERFORMANCE CATEGORY | String | Performance Category (asset class) |
| PERFORMANCE CATEGORY NAME | String | Full name of performance category |
| PERFORMANCE TYPE | String | Type of perfromance (e.g., gross, net) |
| PERFORMANCE INCEPTION DATE | Date | Date when performance began tracking |
| PORTFOLIO INCEPTION DATE | Date | Date when the portfolio was established |
| PERFORMANCE FREQUENCY | String | D- Daily |
| PERFORMANCE FACTOR | Float | Value of performance factor |

Product Master

This script powers the ingestion, classification, and enrichment of the **Product Master** table metadata for Real Estate investment strategies. It begins by defining product records with basic attributes such as product code, name, and vehicle category, either from internal sources or Snowflake staging tables. Classification logic is applied to infer share classes (e.g., Institutional, Retail, Preferred, Class A/B/C) and vehicle types/categories (e.g., Mutual Fund, Separate Account) based on naming conventions and business rules. Products identified as ETFs are recast into modeled strategies—renaming them, assigning them a commingled fund vehicle type, and defaulting to an institutional share class. Additional static fields such as asset class, marketing flag, strategy, and performance account codes are populated to meet ProductMaster table requirements.

Validation routines check for missing or invalid share class assignments, log issues, and enforce thresholds to ensure data quality before loading. Once processed, the enriched dataset is written to a Snowflake staging table using write\_pandas() and merged into the production ProductMaster table with a deduplicated MERGE operation to insert only new products. All column names are normalized for Snowflake compatibility to avoid ingestion conflicts. Logging provides traceability for errors, warnings, and validation results throughout execution.

By automating this workflow, the script eliminates manual classification, improves consistency in product metadata, and ensures alignment with Assette’s business definitions. The resulting standardized dataset underpins fact sheet generation, portfolio analytics, compliance checks, and downstream reporting. This process is critical for maintaining reliable, up-to-date product reference data that supports both internal operations and client-facing deliverables.

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| PRODUCTCODE | String | Unique identifier code for the product |
| PRODUCTNAME | String | Full name of the product |
| ASSETCLASS | String | Asset class designation |
| ISMARKETED | Integer | Indicator if the product is actively marketed (1 = yes, 0 = no) |
| PARENTPRODUCTCODE | String | Code of the parent product grouping |
| PERFORMANCEACCOUNT | String | Identifier for the performance tracking account |
| REPRESENTATIVEACCOUNT | String | Identifier for the representative account |
| SHARECLASS | String | Share class category of the product |
| STRATEGY | String | Investment strategy category |
| VEHICLECATEGORY | String | Classification of the product vehicle type (e.g., "Mutual Fund", "ETF") |
| VEHICLETYPE | String | More specific vehicle type designation (e.g., "Separate Account", "Commingled Fund") |

Security Master

This table serves as the central repository of all security identifiers used in Assette’s REIT data pipeline. It contains the authoritative list of tickers along with associated metadata such as security name, geographic region, sector classification, and industry. The data is primarily sourced from Yahoo Finance and other reference files, ensuring that tickers are standardized and enriched with consistent descriptive attributes. By centralizing this metadata, the Security Master provides a single point of truth for all downstream processes, reducing the risk of mismatched or incomplete security information. The table’s schema is intentionally minimal but covers the core attributes necessary for portfolio assignment, benchmarking, and product mapping. TICKER acts as a unique key and is frequently used as a foreign key in related tables, including holdings, benchmarks, and performance datasets. REGION, SECTOR, and INDUSTRY fields allow for segmentation, filtering, and analytical grouping in reporting. Data in the Security Master is validated for uniqueness of ticker values, non-null descriptive fields, and consistent sector/industry classification. Updates to this table are performed through ingestion scripts that pull fresh data from APIs and CSV inputs, followed by normalization routines. The Security Master also enables interoperability with synthetic data generation scripts, ensuring that simulated securities maintain realistic classifications. This table is foundational to the integrity of Assette’s REIT data warehouse, as every other data domain—performance, holdings, benchmarks, products—relies on accurate and complete security metadata to function properly.

| Column | Data Type | Description |
| --- | --- | --- |
| TICKER | String | Ticker symbol |
| NAME |  | Name of security |
| REGION |  | Geographic Region |
| SECTOR |  | Sector Classification |
| INDUSTRY |  | Industry Name (real estate) |