# **Benchmark Performance to Snowflake Loader**

Developer Documentation

## **Overview**

This Python module loads benchmark performance data, defined using the table structure in Benchmark\_Performance\_table.py, into Snowflake. The generated dataset aligns with the AST\_MULTIASSET\_DB.DBO.BenchmarkPerformance table schema. Before running this script, configure the environment (.env) file with Snowflake credentials and import `get\_benchmark\_performance()` from Benchmark\_Performance\_table.py.

## **Table of Contents**

1. Dependencies
2. Setup and Configuration
3. Data Flow
4. Module Components
5. 4.1 Snowflake Connection Setup
6. 4.2 Retrieve Last Date for Benchmark Code
7. 4.3 Orchestrate Benchmark Load
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**

This script requires:  
- pandas: Data manipulation  
- snowflake.connector: Database connectivity  
- dotenv: Environment variable loading  
- argparse: CLI argument parsing  
- datetime, pathlib, os: File paths and date handling  
It also relies on `get\_benchmark\_performance()` from Benchmark\_Performance\_table.py.

## **2. Setup and Configuration**

• Ensure .env file contains Snowflake credentials: SNOWFLAKE\_USER, SNOWFLAKE\_PASSWORD, SNOWFLAKE\_ACCOUNT, SNOWFLAKE\_ROLE, SNOWFLAKE\_WAREHOUSE, SNOWFLAKE\_DATABASE, SNOWFLAKE\_SCHEMA.  
• Place .env path in ENV\_PATH in the script.  
• Import `get\_benchmark\_performance()` before running.  
• Configure tickers, start/end dates, and frequency through CLI arguments.

## **3. Data Flow**

1) Connect to Snowflake using environment variables.  
2) For each benchmark ticker, query the last available HISTORYDATE1.  
3) Determine start date (last date + 1 or full start date if no data).  
4) Fetch new benchmark data via `get\_benchmark\_performance()`.  
5) Concatenate fetched DataFrames.  
6) Load into a temporary table in Snowflake.  
7) Merge into the target BenchmarkPerformance table to avoid duplicates.

## **4. Module Components**

### **4.1 Snowflake Connection Setup**

Function: get\_snowflake\_connection()  
Purpose: Returns a Snowflake connection object using environment variables.

### **4.2 Retrieve Last Date for Benchmark Code**

Function: get\_last\_date\_for\_code(cs, code)  
Purpose: Queries Snowflake to find the latest HISTORYDATE1 for a given BENCHMARKCODE.  
Ensures incremental loading by avoiding already loaded dates.

### **4.3 Orchestrate Benchmark Load**

Function: orchestrate\_benchmark\_load(tickers, full\_start\_date, end\_date, frequency='D')  
Purpose: Main loader logic combining connection, last date retrieval, data fetch, and load.  
Workflow:  
• Loop through tickers and determine load start date.  
• Fetch data and append to a list.  
• Create temporary table matching target schema.  
• Bulk insert fetched data into temp table.  
• Merge temp table into target table to insert only new rows.

### **4.4 Main Execution Block**

Function: main()  
Purpose: CLI entry point for the script.  
Parses arguments for tickers, full start date, end date, and frequency, then calls orchestrate\_benchmark\_load().

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

• Incremental Load: Fetches only data newer than the last Snowflake record.  
• MERGE Operation: Ensures duplicates are not inserted.  
• Temporary Table: Used for staging before merging into the target.

## **6. Data Structure**

Matches AST\_MULTIASSET\_DB.DBO.BenchmarkPerformance schema:  
BENCHMARKCODE, PERFORMANCEDATATYPE, CURRENCYCODE, CURRENCY, PERFORMANCEFREQUENCY, VALUE, HISTORYDATE1, HISTORYDATE

## **7. Error Handling**

• Skips loading if no new data is found.  
• Handles case when last date is None.  
• Connection closed in finally block to ensure cleanup.

## **8. Testing Considerations**

• Test with tickers having existing data and new data.  
• Verify no duplicates after multiple runs.  
• Confirm schema alignment between fetched DataFrame and Snowflake table.

## **9. Future Enhancements**

• Parallel loading for multiple tickers.  
• Logging integration.  
• Configurable schema and table names.

## **10. Developer Checklist**

✓ Configure .env with Snowflake credentials.  
✓ Import get\_benchmark\_performance() from Benchmark\_Performance\_table.py.  
✓ Run script with correct CLI arguments.  
✓ Validate load results in Snowflake.