Coding Test – Java Developer

Objective

The objective of this coding test is to assist Lonsec in finding a Java developer with proven Java skills. A person that passionate about coding, who will enjoy coding the solution to the below problem. We are looking for a candidate with good software design skills which are demonstrated in their solution. In addition, we expect the candidate to have good written communication skills, assessed by how easily the instructions provided by the candidate can be followed.

Submitting your solution

Create and load your solution to a repository in your GitHub account
Send the link of your GitHub repository to colin.ellerton@lonsec.com.au

Expectations

The solution must be coded in Java 7 or Java 8
Please provide unit tests and instructions on how to run your unit tests and instructions on how to build and run your application.
Include documentation of any assumptions that you made in your solution

Application context

Lonsec has a fund research business and as part of the fund research process there is analysis on the performance data of a fund and how this performance compares to a benchmark.

Application Inputs

Fund.csv

A file containing information about funds. Each row of the file represents a fund.

FundCode: A unique identifier for the fund

FundName: The name of the fund

BenchmarkCode: The code of the benchmark that this fund uses to compare against

funds.csv

FundCode	FundName	BenchmarkCode
fund1	fund 1	bm1
fund2	fund 2	bm1
fund3	fund 3	bm1

Benchmark.csv

A file containing information about benchmarks. Each row of the file represents a benchmark.

BenchmarkCode: A unique identifier for the benchmark

BenchmarkName: The name of the benchmark

benchmark.csv

BenchmarkCode	BenchmarkName
bm1	bench 1
bm2	bench 2

ReturnSeries

A file containing the return series of funds or benchmarks. Each row of the file represents a return for a fund or benchmark on a given date.

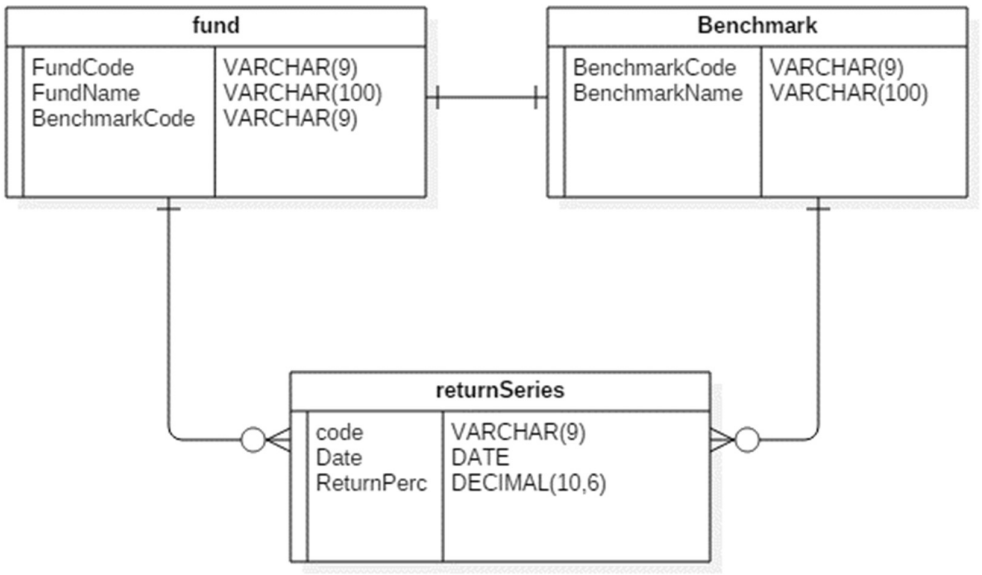
fundReturnSeries.csv

Code	Date	return(%)
fund1	30/06/2016	1.107845
fund1	31/07/2016	0.707635
fund1	31/08/2016	0.420467
fund1	30/09/2016	-0.232219
fund1	31/10/2016	-1.317753
fund1	30/11/2016	-1.470981

benchReturnSeries.csv

Code	Date	return(%)
bm1	31/05/2016	-0.04215
bm1	30/06/2016	1.309476
bm1	31/07/2016	0.730010
bm1	31/08/2016	0.420060
bm1	30/09/2016	-0.225320
bm1	31/10/2016	-1.292895
bm1	30/11/2016	-1.460981

Entity relationship diagram



Application Algorithm

Here is the basic logic required by the application, further detail and examples are provided in the application output sections.

For each fund return in the fundReturnSerices.csv file

- Use the fundCode to lookup the fundName from the fund.csv file
- Use the date to lookup the benchmark return from the benchmarkReturnSeries.csv file
- Subtract the benchmark return from the fund return to get the Excess return.
- Take the Excess return result and lookup the outperformance text

For each date in fundReturnSeries.csv

- take the group of funds for that date and rank them by their return value

Application output file

The application will produce one output file called monthlyOutperformance.csv

The output is to be sorted by [Date] descending, then sub-sorted by [Rank] ascending

Here is an example of the output, only showing data for 30/06/2016. Your solution should contain data for multiple months, as described further down in this document.

monthlyOutperformance.csv

FundName	Date	Excess	OutPerformance	Return	Rank
Fund 1	30/06/2016	1.15	out performed	1.11	1
Fund 5	30/06/2016	1.21	out performed	0.71	2
Fund 2	30/06/2016	0.58		0.42	3
Fund 4	30/06/2016	0.76		0.23	4
Fund 3	30/06/2016	-1.98	under performed	-1.32	5
Fund 7	30/06/2016	-1.90	under performed	-1.47	6
Fund 6	30/06/2016	-1.35	under performed	-2.10	7

Application Output Calculations

This section describes the columns in the output and how to derive the values.

Excess

For each fund (fundReturnSeries.csv) month end, lookup the benchmark (benchReturnSeries.csv) data row for the corresponding date, Subtract the benchmark return from the fund return. Round Half-Down to 2 decimal places

Example for FundCode fund1, on 30/06/2016

fundReturnSeries.csv

Code	date	return(%)
fund1	30/06/2016	1.107845

benchReturnSeries.csv

Code	date	return(%)
bm1	2016-06-30	-0.04215

Excess = [fund return (%)] - [benchmark return (%)]
= 1.107845 – (–0.04215)
=1.149995
=1.15%

Outperformance

For the Excess calculated for the fund on the given month, use the following table to lookup the outperformance text.

Excess	OutPerformance
< -1%	underPerformed
> 1%	outPerformed
Otherwise	<<BLANK>>

e.g. If the excess for Fund1 on the 30/06/2016 is 1.15.

1.15 is greater than 1, this matches the second condition, so the outperformance text is:-

Out performed

Rank

For each month end within the fundReturnSeries.csv file, lookup all funds which have a row for that month end. Then for that month end rank the funds by their return(%) value. The fund with the highest return(%) is rank 1

Example for 30/06/2016

fundReturnSeries.csv

FundCode	date	return(%)	Rank
fund1	30/06/2016	1.107845	1
fund5	30/06/2016	0.707635587	2
fund2	30/06/2016	0.420467635	3
fund4	30/06/2016	-0.232219351	4
fund3	30/06/2016	-1.317753134	5
fund7	30/06/2016	-1.470981688	6
fund6	30/06/2016	-2.095689	7

Design considerations

The files provided are samples only, we will test your code on larger input files.
Input parameters may change, e.g. file name, file path, excess lookup, outperformance text
There could be bad data in the input files. We might want to add more calculations