Code2Connect 2022
FICC Tech Problem Statement – Portfolio Tracker

Contents

1 Background and Introduction: 1
2 Portfolio and Bond Net Value Calculations: 2
2.1 Net Value and Cash Calculation as per events 3
2.2 Report Generator - Format 3
3 Requirement for the challenge 4
3.1 Essential Components: 4
3.1.1 Implementation boundaries: 5
3.1.2 Requirements: 6
3.1.3 Assumptions: 6
3.1.4 Marking Scheme: 6
3.2 Additional Component (Optional, for additional 10 Points) 7
3.2.1 Requirement and Marking Scheme. 7
3.3 Sample Input and Output 7
3.3.1 Inputs 7
3.3.2 Output 8
3.3.3 Presentation: 9
4 Words of Advice 10

1 Background and Introduction:

In a global financial institution, there are millions of trades processed on each day by various Trading Desk in different regions. Each of these desks will have multiple traders who will be making buy/sell decision on various products to maximize profit or hedge risks.

At a given point of time all these buy/sell of a particular product will result in change in positions and overall net-value of the portfolio for a trader as well as desk.

In order to track the current value desk/ trader will need an interface to look at following data points on real time as well as historical:

1. Current value of portfolio as per latest market and trade data.
2. Cash available in hand for a desk

Your task for this challenge is to develop a Portfolio Tracker system to show above data points at various intervals as well as real time.

For the purpose of this exercise, we are limiting traded product to Bond.

2 Portfolio and Bond Net Value Calculations:

Value of Bond can be calculated as per formula given below

Bond Net Value = Bond Quantity * Price / FXRate

Suppose we have a Bond B1 of quantity 100 on currency JPX with traded price 120892.Then as per initial FX Rate bond value will be:

*B1 Value =100*120892/136.14*

Let's take a simple example:

Initial FX Rates:

Currency	FxRate
JPX	136.14
USX	1

Table 1: Initial FX Rates

Initial Cash with Desks:

Desk	Cash
NY	1000000

Table 2: Initial Cash with Desks

Bond details:

BondID	Currency
B34678	JPX

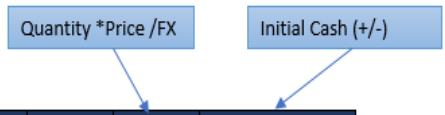
Table 3: Bond details

Events:

{'EventID': 1, 'EventType': 'PriceEvent', 'BondID': 'B34678', 'MarketPrice': 10000}
{'EventID': 2, 'EventType': 'TradeEvent', 'Desk': 'NY', 'Trader': 'T6899554', 'Book': 'NY00', 'BuySell': 'buy', 'Quantity': 533, 'BondID': 'B34678'}
{'EventID': 3, 'EventType': 'PriceEvent', 'BondID': 'B34678', 'MarketPrice': 10090}
{'EventID': 4, 'EventType': 'TradeEvent', 'Desk': 'NY', 'Trader': 'T6899554', 'Book': 'NY00', 'BuySell': 'sell', 'Quantity': 33, 'BondID': 'B34678'}
{'EventID': 5, 'EventType': 'FXEvent', 'ccy': 'JPX', 'rate': 135}
{'EventID': 6, 'EventType': 'PriceEvent', 'BondID': 'B34678', 'MarketPrice': 10100}

Table 3: Sequential event data

2.1 Net Value and Cash Calculation as per events



After Event	Desk	Trader	Book	Buy/Sell	Quantity	BondID	Position	Price	FX	Value	Cash
2	NY	T6899554	NY00	B	533	B34678	533	10000	136.14	39150.87	960849.13
4	NY	T6899554	NY00	S	33	B34678	500	10090	136.14	2445.791	963294.93
6	NY	T6899554	NY00				500	10100	135	37407.41	No change in cash

2.2 Report Generator - Format

Cash level portfolio* after event 6 -Shows cash in USX for each desk*

Desk	Cash
NY	963294.92

Table 4: cash_level_portfolio_6.csv

Position_level_portfolio* after event 6 -Shows trader and Book level Position and NetValue for each desk*

Desk	Trader	Book	Positions	NV
NY	T6899554	NY00	500	37407.41

Table 5: position_level_portfolio_6.csv

***bond_level_portfolio* after event 6-** *Shows trader, Bond and Book level Position and NetValue for each desk*

Desk	Trader	Book	BondID	Positions	NV
NY	T6899554	NY00	B34678	500	37407.41

Table 6: bond_level_portfolio_6.csv

Currency level portfolio after event 6 - *Shows currency level Position and NetValue for each desk*

Desk	Currency	Positions	NV
NY	JPX	500	37404.41

Table 7: currency_level_portfolio_6.csv

Terms	Explanation
Desk	A trading desk is a designated area of a bank, investment firm, or trading floor dedicated to the sale and purchase of specific types of products. One desk can have multiple traders.
Book	The book in which the trade is recorded; one book can have multiple trades; each trade can only belong to one book. A trader can own multiple books, but a book belongs to single trader. Think of book as an account.
Instrument	An asset which can be traded. For our use case it is Bond with unique Ids
Trader	Unique Id given to a person in a trading desk who decided when to buy and when to sell an instrument. Each trader belongs to a particular desk.
Positions	Total quantity group by instrument and books
Cash	Actual cash in hand as per desk denoted in USX for this problem statement
Bond	A fixed income financial instrument. Bonds are units of corporate debt issued by companies and securitized as tradeable assets
Currency	Currency in which trade gets represented. In real world it is USD, SGD etc.
Portfolio	Representation of positions at different level, e.g., Book. Trader, Desk
FXRate	Rate to convert one currency to another
NV	Net Value = Quantity *Market Price

Table 8:Terms Explanation

3 Requirement for the challenge

For the purpose of this challenge, you must work individually on following components:

3.1 Essential Components:

1. An embedded database to:
    a. Get the bond details from an embedded database:
        i. Design a schema to store Bond definition from **Bond.csv** file.
    b. Get initial cash from an embedded database
        i. Design a schema to store initial cash from **Initial_cash.csv** file,
2. Read data from json file **events.json** and create components /interface listed below. You can add/remove more components as per your design:
    a. **Market Data Producer**: Implement a market data producer that publishes FX and Bond prices on regular interval. You can induce delay of 1-5 seconds in publishing to simulate real world. This will help to create real-time dashboard to tick when FX event comes in .
    b. **Trade Event Data Producer:** Implement a trade event data producer that publishes trade event on regular interval. You can induce delay of 1-5 seconds in publishing.
    c. **Portfolio Engine** -Subscribes to market data and trade event to show real time value of a portfolio. Implements a real time Bond price calculator to calculate bond value with underlying price. Instruction to calculate Bond price is given in section 2.
    d. **Cash Adjuster** -Adjust the state of cash for each desk based on trading events received and the latest FX rates for the respective denomination currencies received.
    e. **Report Generator** - Implement a reporting interface to query the Cash and Positions after a certain number of event message have been processed. These reports are slicing

and dicing of same data at different levels. Report generator outputs a file in a given format as per query criteria. Different query criteria to be implemented are given below:

i. ***cash_level_portfolio :***

Input
event_id=876
category= ['Desk', 'Trader', 'Book']
measures=['Cash']

ii. ***position_level_portfolio :***

Input
event_id= 876
categories= ['Desk', 'Trader', 'Book']
measures= ['Positions', 'NV']

iii. ***bond_level_portfolio:***

Input
event_id=876
categories= ['Desk', 'Trader', 'Book', 'BondID']
measures= ['Positions', 'NV']

iv. ***currency_level_portfolio :***

Input
event_id=876
categories= ['Desk', 'Currency']
measures= ['Positions','NV']

f. **Portfolio dashboard -**User Interface to show ticking portfolio as per trader/Desk.

3. A Microsoft PowerPoint presentation deck that does not exceed 10 pages to display and explain your solutions. A suggested format is in Presentation:

3.1.1 Implementation boundaries:

1. At no point Cash balance for a particular desk can go below zero. All such buy events should be mark as an exception.
2. A trader cannot sell unless it owns a particular Bond of that quantity in that book. All such sell events should be mark as an exception.
3. Though market data and event data ticks separately they should follow sequential order as per event_id.
4. Report generator queries are inclusive of given event_id
5. Number of currencies are not limited to initial data set.

6. Number of desks, traders are not limited to initial data set.
7. All final output data should be round up to two decimal places. Your final value could differ few points based on rounding up at various level. Try to get as close as possible to sample output file provided.
8. You are free to use any language and libraries.

3.1.2 Requirements:

1. Provide clear documentation with instructions to run the application and other implementation details.
2. Your program should be well documented in-line and should explain itself.
3. Your program should produce relevant logs as and when needed.
4. We will provide one set of validation input and output files in the format defined in section Inputs and section output This is only for your reference.
5. We will be giving several sets of testing input files in the format defined in input section on day before submission. We will also mention set of eventIds you need to process. You are required to **process all of them and send back your results** in given output format This is used to evaluate your work.

3.1.3 Assumptions:

- We will be using some hypothetical currencies for our data sets.
- All the results which we report from our system will be reported in "USX".
- All **TradeEvent** Buy/Sell will be a Market Order. This means price will be MarketPrice from last published **PriceEvent** for that BondID.
- CSV file could have additional whitespaces. Validation and correction should be done by the code.

3.1.4 Marking Scheme:

Item	Points
Overall design, architecture, algorithm, performance, documentation, coding practice, test cases and edge cases.	20
Set up a functioning essential component - **Market Data Producer, Trade Event Data Producer, Portfolio Engine and CashAdjuster**	20
Set up a functioning essential component-**Report Generator** to generate proper results as per given query and format	20
Presentation decks on thinking process, design considerations, algorithm complexity, solutions and challenges faced.	20
Portfolio dashboard -User Interface to show ticking portfolio as per trader/Desk. Be creative and implement a simple user-friendly design. Can be a CLI as well.	20
Subtotal	100

Table 8: Objectives & Milestones for Essential Component

3.2 Additional Component (Optional, for additional 10 Points)

1. Develop report generator where any form of slicing and dicing of given data is allowed based on different combination of category and measures.
2. Any other visual representation of data you can think of – Graph /chart /tables

3.2.1 Requirement and Marking Scheme.

3. Select the above additional components based on your interest, strength and schedule. You don't have to complete all of them.
4. Indicate if you have attempted additional component. For additional component, you can add up to two pages in the presentation to display your work.
5. Additional component has a full mark of **10 additional points** and is marked independently.
6. The rest remains the same as 3.1.4

3.3 Sample Input and Output

3.3.1 Inputs

1. initial_fx.csv
    a. Contains initial FX rate for set of hypothetical currencies.
    b. Definition of terms are the same as Table 8

Currency	FxRate
JPX	136.14
USX	1

Table 9: initial_fx.csv format

2. bond_details.csv
    a. Contains bond Id to uniquely identify each bond and underlying currency for the bond.
    b. Definition of terms are the same as Table 8

BondID	Currency
B34678	JPX

Table 10: bond_details.csv_format

3. initial_cash.csv
    a. Contains Desk and Cash in hand as per each desk.
    b. Cash is always in USX
    c. Definition of terms are the same as Table 8

Desk	Cash
NY	1000000
TOK	1000000

Table 11: initial_cash.csv formats

4. events.json
    a. Contains three type of events:

i. **TradeEvent** – Denotes buy/sell event of a particular bond.

```
{
'EventID': 4,
'EventType': 'TradeEvent',
'Desk': 'NY',
'Trader': 'T6899554',
'Book': 'NY00',
'BuySell': 'sell',
'Quantity': 33,
'BondID': 'B34678'
}
```

ii. **FXEvent** – Denotes change in current FX for a particular currency

```
{
'EventID': 1000,
'EventType': 'FXEvent',
'ccy': 'SGX',
'rate': 1.35
}
```

iii. **PriceEvent** - Denotes change in price for a particular bond. Price is always in underlying bond currency.

```
{
'EventID': 6,
'EventType': 'PriceEvent',
'BondID': 'B34678',
'MarketPrice': 10100
}
```

3.3.2 Output

We will provide you set of EventIds before final day of submission. You should run you code against those EventIds and provide use output as per instruction given below:

Create different folder for each EventId. Folder should be name as output_<< EventId>>. Each folder should contain all csv file mention below:

1. *exclusions_<<event_id>>.csvs*
   - Excluded **TradeEvent** which cannot be processed because:
     - QUANTITY_OVERLIMIT -There is sell **TradeEvent** though there is no relevant quantity available for that trader in that books.
     - CASH_OVERLIMIT -There is buy **TradeEvent**, but existing cash for that desk is all exhausted.
     - NO_MARKET_PRICE – If there is buy/sell **TradeEvent** without any **PriceEvent** for that Bond Id earlier mark those trade as excluded.
   - Format:

EventID	Desk	Trader	Book	BuySell	Quantity	BondID	Price	ExclusionType
								CASH_OVERLIMIT
								QUANTITY_OVERLIMIT

Table 13: exclusion.csv formats

2. *cash_level_portfolio_<<event_id>>.csv* -Contains cash in USX for each desk

Desk	Cash
NY	555857.68
TOK	1000000

Table 14: cash_level_portfolio_4.csv

3. *position_level_portfolio_<<event_id>>.csv* - Contains trader and Book level Position and NetValue for each desk

Desk	Trader	Book	Positions	NV
NY	T6899554	NY00	500	444142.32

Table 15: position_level_portfolio_4.csv

4. *bond_level_portfolio_<<event_id>>.csv* - Contains trader, Bond and Book level Position and NetValue for each desk

Desk	Trader	Book	BondID	Positions	NV
NY	T6899554	NY00	B34678	500	444142.32

Table 16: bond_level_portfolio_4.csv

5. *currency_level_portfolio_<<event_id>>.csv* - Contains currency level Position and NetValue for each desk

Desk	Currency	Positions	NV
NY	JPX	500	444142.32

Table 17: currency_level_portfolio_4.csv

6. **Portfolio dashboard** -There is no specific format. You can use command line to show ticking data as per desk or Trader. Your Dashboard can tick at time intervals or once an event comes in. Be creative and come up with nice dashboard to display
   a. Total positions group by either Trader /Desk/Book
   b. Total portfolio Net value

3.3.3 Presentation:

The following is a suggested format of your presentation. This is for your reference only.

1. Cover Page: include your name and University

2. Overview: describe the high-level approach taken for the solution, and which additional components you have attempted.
3. Data Analysis: describe the input/output data
4. Design and architecture of your solution.
5. UI interface demo
6. Optionally, any additional commentary on your solution
7. Results from Essential Component -1
8. Results from Essential Component -2
9. Solutions of Additional Components
10. Takeaways and conclusions

4 Words of Advice

This project is intended to give you an exposure into how engineers work at the bank i.e., you get to work on everything which you like, where you feel challenged and where there is immense learning. Start early. The UI should be functional and intuitive rather than beautiful. Don't complicate the algorithm i.e., don't add too many conditions and make it as generic as much as possible. Moreover, an important part of working at the Bank is making it easy for the users and fellow engineers, so write very well documented code along with any supportive documentation. All the very best and make sure you enjoy the project!