Data Engineer coding task

Convert level 3 to level 1 data.

You may write code with any language, library, or framework you are comfortable with. There should be clear instructions on how to execute the code. Note that this is a data engineering task and not a data science task, i.e. it's about engineering systems and not producing analytic insights.

The purpose of an exchange is to match market participants orders (buy against sell). The state of the market is represented in an order book. This shows for each price level and each side (buy/sell) the orders for that level.

Level 3 data are *updates* to market participants' orders. Because sending the full state of the order book for each modification to the book is slow and wastes bandwidth, the exchange publishes deltas/diffs instead, which describes *updates* to the state of the book.

Each order has a side, price and quantity. They are uniquely identified by an order id. An order update can be one of ADD, UPDATE, DELETE, or TRADE. ADD inserts a new order into the book. UPDATE modifies an existing order on the book. DELETE removes the order from the book. TRADE matches buy orders with sell orders at a specific price; the corresponding update to the book being a reduction of quantity outstanding in the respective orders according to the amount traded. (separate TRADE messages will be sent for buy vs. sell, so you only need to handle one side at a time; also the other side of the trade may never show up in the book, i.e. they are executed immediately)

Note that if there are any orders on both sides of a price level, then they would already be matched and traded. Hence at any given time all the outstanding sell orders are above a certain price level and all the outstanding buy orders would be below that price level. The lowest sell price level (ask price) with all the cumulative quantity at ask price level (ask size) and the highest buy level (bid price) with all the cumulative quantity at bid price level (bid size) constitute what we call the BBO (best bid and ask (offer)).

Level 1 data refers to the stream of updates to BBO as a consequence of updates to the book state. It is derived data, with the Level 3 market data being the raw data. Unlike with with Level 3 data the full state of the BBO should be published each time as it is not a voluminous amount of data.

You do not have to match orders: that is the exchange's decision, not ours. TRADE messages we receive are only for purposes of updating our view of the book.

Things to consider:

- How would you deal with data which arrive out of order?
- Bonus points for a streaming implementation.
- How might it be possible for a unified batch and streaming implementation to work?

Use the provided CSV files as L3 market data and expected L1 market data to test your program.

Explanation on L3 market data file format: (https://drive.google.com/open?id=1Kus049iLkqi6q-g3gkrypQIY65LUU3Ox)

Column Name	Notes
HOST	Timestamp in UTC. ("HOST" refers to timestamp of the host machine as opposed to at the exchange) Timestamp is not used to define an ordering on the data. See seq_num instead.
seq_num	Sequence number of L3 market data, which defines an ordering on L3 updates, i.e. packets with sequence number N should be applied to update the state of the book before doing the same with sequence number N+1. All the L3 market data entries carrying the same sequence number are received from exchange in a single packet. So you should output L1 market data in order of sequence number. You may assume there are no gaps and no duplicates in sequence numbers, except in cases when an image is received. See is_image. Note that the seq_num numbers are very large, and the program you are using to view the file may not distinguish between them. If in doubt, use a text viewer.
is_image	True or Null for a given seq_num Upon receiving such seq_num, you should clear existing book content you have built. Such image is received upon exchange open session.

add_orderid	OrderId is unique for side. You can have same orderId for buy and sell, you need to treat them as two different orders.
add_side	BUY or SELL
add_price	Price of this order For market orders, the value will be Integer.MAX for BUY and negative Integer.MAX for SELL
add_qty	Quantity of this order
add_position	Position of this order among all the orders queuing for the same price. As the position of an order does not affect the best bid and ask, you can ignore this column for this exercise.
update_orderid	
update_side	OrderId and Side are used together to identify the original Add order it’s trying to update
update_price	
update_qty	
update_position	See add_position
delete_orderid	OrderId and Side are used together to identify the order it’s trying to delete
delete_side	
trade_orderid	trade_orderid and trade_side are used in conjunction to identify the (resting) order on the book this trade applies to (the aggressing order (the one on the other side) is irrelevant to the purpose of updating the L3 book) trade_qty should be subtracted from the existing quantity of the (resting) order if there is no quantity remaining after subtraction, the (resting) order should be considered removed from the L3 book
trade_side	used in conjunction with trade_orderid to identify the (resting) order this trade applies to

trade_qty	determines quantity to subtract from the (resting) order
trade_price	this is irrelevant for the purposes of updating the L3 book

Expected L1 market data output:
(https://drive.google.com/open?id=1gt2BbVAehVo5_XKidSi2eyAqWMiJMBK2)

Column Name	Notes
time	Timestamp of L3 market data which triggers L1 market data update (in UTC)
bid_price	Best bid price
ask_price	Best ask price
bid_size	Total quantity for the best bid price
ask_size	Total quantity for the best ask price
seq_num	Sequence number of L3 market data which triggers L1 market data update

Common Questions:

1) How do I handle 1.7976931348623157e+308 in input L3 file?
This is Integer.MAX (java), you should treat any order carrying price equals Integer.MAX or Integer.MIN as market order, and exclude such **order** from book calculation.

2) Isn't 1.7976931348623157e+308 a floating point number?
No, check the exponent, it is an integer. In finance, we use integers so as to not lose precision from floating point arithmetic.

2) Why my L1 output has more records compared with expected output?
The expected L1 result was generated with consideration of other info, which was not exacted out in L3 market data. So your own output will have more L1 updates. This is expected.
You can use time column as the key to match your output to expected file. For any matched row, the rest of columns should carry the same values as the expected file.