Convert level 3 to level 1 data.

You may write code in any language 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

level and all the outstanding buy orders would be below that level. The lowest sell (ask) level

and the highest buy (bid) level 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-

g3gkrypQlY65LUU3Ox)

|  |  |
| --- | --- |
| Column Name | Notes |
| HOST | Timestamp in UTC |
| 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 per sequence number. |
| 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.  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 |  |
| 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, you should treat any order carrying price equals Integer.MAX or

Integer.MIN as market order, and exclude such order from book calculation.

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.