BorderGuru – Full Stack Developer

Imagine you want to assist some US-based merchants to ship their goods to resellers in Hamburg. Lets assume that every order can be represented by a tuple ***(orderId, companyName, customerAddress, orderedItem, price, currency)*** in order to allow finance team to invoice.
Thus, some sample data might look like this:

```
001, SuperTrader, Steindamm 80, Macbook, 1700, EUR
002, Cheapskates, Reeperbahn 153, Macbook, 1700, EUR
003, MegaCorp, Steindamm 80, Book "Guide to Hamburg", 20, EUR
004, SuperTrader, Sternstrasse  125, Book "Cooking  101", 10, EUR
005, SuperTrader, Ottenser Hauptstrasse 24, Inline Skates, 75, EUR
006, MegaCorp, Reeperbahn 153, Playstation 4, 270, EUR
007, Cheapskates, Lagerstrasse  11, Flux compensator, 2000, USD
008, SuperTrader, Reeperbahn 153, Inline Skates, 75, EUR
```

Using **node.js**, please implement a working solution to perform the following kind of operations on the data:

1. Add new order
2. Show all orders from a particular company
3. Show all orders to a particular address
4. Delete a particular order given an OrderId
5. Display how often each item has been ordered, in descending order (ie in the above example, 2x for Macbook and Inline skates, 1x for the rest)

Now imagine that after deploying and running the system described before successfully for some months, your stakeholders ask you to register companies info collection in order to store final customer's info. Assuming that ***you cannot edit the order structure you created in previous exercise***, as it would break retro compatibility, add all the data you think is necessary to represent a company entity and provide the following functionalities:

I. Get company info
II. Update company info
III. Delete company
IV. Get all order bought by one company
V. Get the amount of money paid by a company
VI. Get all companies that bought a certain orderItem

Please optimize your code and do not convolute it with handling exceptions/edge cases – we are more interested in readability for this solution.

Finally, could you answer these questions:

a) Why did you pick your first particular your design? What assumptions did you make, and what tradeoffs did you consider?

b) What did you consider when you had to design the second solution? And which assumptions and tradeoffs you made?

c) What do you like (and dislike) about Node/Javascript compared to other programming languages?

Please send us **an archive** with your solution together with build instructions.
If you have any further questions please let us know. Otherwise, we look forward to seeing your solution!