**Q2 (a)**

(i) One of the 5 Vs of Big Data is *Velocity*. Explain what it means and

what this has to do with NoSQL (3 marks)

**Answer**:

* **what it means**: the speed of data processing/how fast the data is coming in.

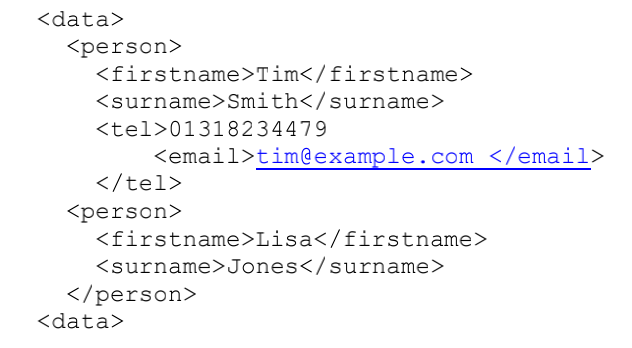
[reference] <https://www.bbva.com/en/five-vs-big-data/>

<https://whatis.techtarget.com/definition/3Vs>

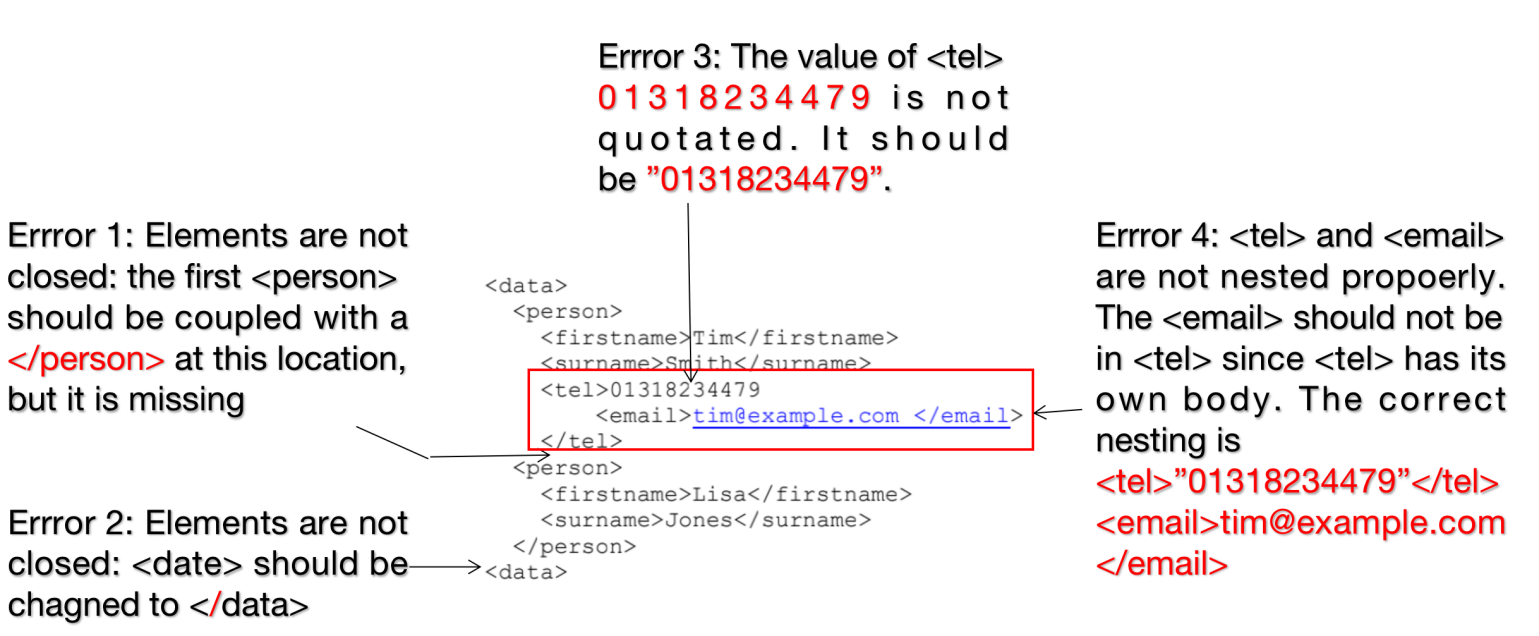
* **what this has to do with NoSQL**: The NoSQL is more suitable than SQL database to fit the Velocity. The reason is the the NoSQL database stores the data in a form of flat collections. The single piece of data is usually not partitioned and stored in a form of entity. Read and writing a single entity is faster and thus much more easier than SQL. This nature makes NoSQL a better choice to handle the read-time processing of the fast-coming data.

[reference] <https://www.dezyre.com/article/nosql-vs-sql-4-reasons-why-nosql-is-better-for-big-data-applications/86>

(ii) Identify 3 things that are incorrect in the following XML (3 marks)



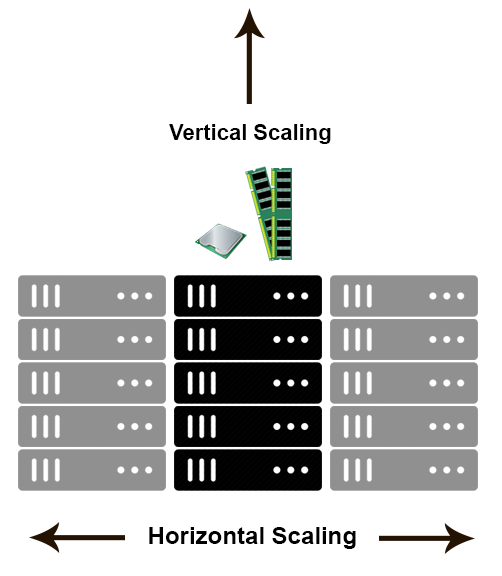
**Answer**:



(iii) Explain the difference between *vertical* and *horizontal scaling* (2 marks)

**Answer:**

* **Horizontal scaling** means that you scale by adding more **machines** into your pool of resources whereas
* **Vertical scaling** means that you scale by adding more **power (CPU, RAM)** to an existing machine.



[reference] <https://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases>

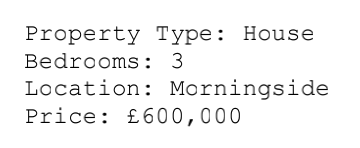
**Detailed Differences**:

|  |  |  |
| --- | --- | --- |
|  | Horizontal Scaling  (scaling out) | Vertical Scaling  (scaling up) |
| Databases | In a database world, horizontal scaling is usually based on the partitioning of data (each node only contains part of the data). | In vertical scaling, the data lives on a single node and scaling is done through multi-core, e.g. spreading the load between the CPU and RAM resources of the machine. |
| Downtime | In theory, adding more machines to the existing pool means you are not limited to the capacity of a single unit, making it possible to **scale with less downtime**. | Vertical scaling is limited to the capacity of one machine, scaling beyond that capacity can involve downtime and has an upper hard limit, i.e. the scale of the hardware on which you are currently running. |
| Concurrency | Also described as distributed programming, as it involves distributing jobs across machines over the network. Several patterns associated with this model: Master/Worker\*, Tuple Spaces, Blackboard, MapReduce. | Actor model: concurrent programming on multi-core machines is often performed via multi-threading and in-process message passing. |
| Message passing | In distributed computing, the lack of a shared address space makes data sharing more complex. It also makes the process of sharing, passing or updating data more costly since you have to pass copies of the data. | In a multi-threaded scenario, you can assume the existence of a shared address space, so data sharing and message passing can be done by passing a reference. |
| Examples | Cassandra, **MongoDB**, Google Cloud Spanner | MySQL, Amazon RDS |

[reference] <https://www.section.io/blog/scaling-horizontally-vs-vertically/>

(iv) Consider the following property information (i.e. building) and

write in JSON format (2 marks)



**Answer**:

