1 Theory

1. SSD
2. asd
3. sad
4. asdasd
5. The numbers start at 1 with each use of the `enumerate` environment.
6. Another entry in the list
7. When it comes to fault tolerance, a hardware error could be a disk, power, CPU, or network failure. A software error could be a bug in the system causing errors. It could also be system crashes due to errors caused by the system having been turned on for too long. Human errors are usually due to accidental actions such as deleting or overwriting a file by mistake, or inserting wrong data.
8. To achieve fault tolerance, one could restart computers frequently to avoid crashes due to the system having been running for too long. One could also store the data across multiple computers such that in the case one computer fails, only a small portion of the data is affected, though this is hard to do on stateful systems. Another way to achieve fault tolerance is by replication, i.e. having the same data stored on multiple computers. This way, if one computer fails, the data is still available on the other computers, though this method could be expensive.
9. SQL is a relational database where relationships between different data in the database can be easily defined using joins. In a document database, however, this is not the case. This is due to the fact that the data is stored in documents, and the documents are self-contained, i.e. not split into multiple tables. This structure also makes it bad for supporting many-to-many relationships, something SQL is good at. In the case where we wish to model a paper that has many sections and words, and additionally many authors, where each author with name and address have many written papers, we would have to store each entity in its own self-contained document. So we would have one document representing each paper with sections and words, in addition to the authors of the paper. We would have another document representing all authors with their name and address. The relationship defining which authors have written which papers would be stored in the paper document, though to extract their name and address we would have to define many-to-many relationshops not contained in any of the documents, presumably using application code or some other method. Using only self-contained documents, we would have to store a lot of redundant, duplicate data where each paper has all authors, with their names and adresses as well, listed in the document. This is however not a good solution due to the fact that in different papers, the same author could have their name spelt differently or have some name changes. Trying to get all papers written by a specific author could prove difficult due to this.
10. If the data comes in the form of self-contained documents where there are few relationshops between them, document databases are a good fit. However, in the case where anything could be related to everything, a graph database would be a better fit. For example, when we want to store data containing people and how they are related to each other, a graph database would be preferable over a document database, as the data is highly relational.
11. asdasd
12. asdasd
13. sad