**1) Find the problems with ACID, and potential solutions without going to NoSQL**

ACID (Atomicity, Consistency, Isolation, Durability) is great for ensuring data integrity in databases, but it can be a bit strict at times. Here's a breakdown of the issues and how we can address them without switching to entirely different databases:

**Problems:**

* **Slowness:** ACID guarantees can make writing data slower, especially with complex transactions.
* **Scalability:** ACID databases can struggle to handle a massive amount of data or concurrent users.

**Solutions:**

* **Relaxing ACID:** Some systems can loosen certain ACID properties for specific situations. For instance, sacrificing some consistency for faster writes.
* **Optimistic Locking:** This approach lets multiple users edit data, but only the latest version gets saved, avoiding conflicts.
* **Eventual Consistency:** Data might not be instantly updated everywhere, but eventually becomes consistent across systems.

These are some ways to improve performance and scalability without abandoning the core strengths of ACID-based databases. It's about finding the right balance for your specific needs.

**2) Explain how ACID Properties are implemented in Distributed Database Transaction**

A distributed transaction is defined as a group of operations that are to be performed across more than one database or data repository. The operations are performed by multiple nodes that are connected to a single network. The distributed transaction ensures ACID (Atomicity, Consistency, Isolation, Durability) properties and data integrity.

Distributed databases spread information across different computers. ACID properties ensure this scattered data stays consistent even when updates fly around. Here's a breakdown of how they work:

* **Atomicity:** All or nothing - Imagine a bank transfer. Either the entire amount moves from one account to another, or nothing happens. In a distributed system, if one update fails at any location, the entire transaction is reversed everywhere.
* **Consistency:** Stays by the rules - Transactions follow predefined rules to keep the data in a valid state. Think of it like a recipe. You follow the steps (the transaction) and get a cake (consistent data) every time.
* **Isolation:** No peeking! Transactions are isolated from each other. Even if two transactions happen simultaneously, they don't see each other's intermediate changes. This prevents conflicts and ensures each transaction sees a consistent view of the data.
* **Durability:** Once written, stays written - Once a transaction is complete, the changes are permanently stored across all locations. Even if a system crashes, the data remains safe and sound. Imagine writing your grocery list on a whiteboard (distributed database). Even if the light flickers (system crash), your list is saved on a backup notepad (durable storage).

Implementing these properties in distributed databases can be tricky, but they are essential for ensuring data integrity and reliability.