**Assignment 3 : Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.**

**ACID Properties of a Transaction:**

ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure that database transactions are reliable, consistent, and recoverable.

**Atomicity:** This property ensures that a transaction is treated as a single unit of work. Either all operations within the transaction are successfully completed, or none of them are. If any part of the transaction fails, the entire transaction is rolled back, and the database is left unchanged.

**Consistency:** Consistency ensures that the database remains in a consistent state before and after the transaction. The integrity constraints, such as primary key constraints and foreign key constraints, are maintained throughout the transaction. It ensures that only valid data is written to the database.

**Isolation:** Isolation ensures that multiple transactions can operate independently of each other without interfering. Transactions should be isolated from each other until they are committed. This prevents concurrency-related issues such as dirty reads, non-repeatable reads, and phantom reads.

**Durability:** Durability guarantees that once a transaction is committed, its changes are permanently stored in the database and cannot be lost, even in the event of a system failure. The changes made by the committed transaction persist even if the system crashes or restarts.

**SQL Statements to Simulate a Transaction with Locking:**

Let's simulate a transaction using SQL statements with locking and demonstrate different isolation levels for concurrency control. We'll create a simple scenario where two transactions are trying to update the same row simultaneously.

Consider a table Employee with columns EmployeeID, Name, and Salary.

**Here's how we can simulate a transaction with locking using SQL statements:**

**-- Transaction 1**

**START TRANSACTION;**

**SELECT \* FROM Employee WHERE EmployeeID = 1 FOR UPDATE;**

**UPDATE Employee SET Salary = Salary + 1000 WHERE EmployeeID = 1;**

**COMMIT;**

**-- Transaction 2**

**START TRANSACTION;**

**SELECT \* FROM Employee WHERE EmployeeID = 1 FOR UPDATE;**

**UPDATE Employee SET Salary = Salary + 2000 WHERE EmployeeID = 1;**

**COMMIT;**

In this example, both transactions are trying to update the salary of the employee with EmployeeID 1. The FOR UPDATE clause is used to lock the selected row, ensuring that other transactions cannot update it until the current transaction is complete.