***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.*

***explanation of ACID properties and SQL statements to demonstrate transactions, locking, and concurrency control:***

***ACID Properties:***

*ACID (Atomicity, Consistency, Isolation, Durability) ensures database transactions are processed reliably:*

*Atomicity: Ensures all operations in a transaction succeed or fail together.*

*Consistency: Maintains database consistency, enforcing constraints and rules.*

*Isolation: Controls concurrency, preventing transactions from interfering.*

*Durability: Guarantees transaction changes are permanent.*

***SQL Transaction Example:***

*-- Start transaction*

*BEGIN TRANSACTION;*

*-- Operation 1: Debit account A*

*UPDATE accounts SET balance = balance - 100 WHERE account\_id = 1;*

*-- Operation 2: Credit account B*

*UPDATE accounts SET balance = balance + 100 WHERE account\_id = 2;*

*-- Commit transaction*

*COMMIT;*

***Locking:***

***Locking ensures exclusive access to data:***

*-- Lock account A for update*

*SELECT \* FROM accounts WHERE account\_id = 1 FOR UPDATE;*

*-- Lock account B for update*

*SELECT \* FROM accounts WHERE account\_id = 2 FOR UPDATE;*

*-- Perform updates (as above)*

*-- Release locks*

*COMMIT;*

***Isolation Levels:***

***Isolation levels control concurrency:***

*READ UNCOMMITTED: Allows dirty reads.*

*READ COMMITTED: Prevents dirty reads.*

*REPEATABLE READ: Prevents non-repeatable reads.*

*SERIALIZABLE: Highest isolation, prevents all concurrency issues.*

***SQL Isolation Level Example (MySQL):***

*-- Set isolation level*

*SET TRANSACTION ISOLATION LEVEL READ COMMITTED;*

*-- Start transaction*

*BEGIN TRANSACTION;*

*-- Perform operations (as above)*

*-- Commit transaction*

*COMMIT;*

***Concurrency Control Demonstration:***

***Session 1 (Transaction 1):***

*SET TRANSACTION ISOLATION LEVEL READ COMMITTED;*

*BEGIN TRANSACTION;*

*SELECT \* FROM accounts WHERE account\_id = 1 FOR UPDATE;*

*-- Wait for Session 2 to update account 1*

*UPDATE accounts SET balance = balance - 100 WHERE account\_id = 1;*

*COMMIT;*

***Session 2 (Transaction 2):***

*SET TRANSACTION ISOLATION LEVEL READ COMMITTED;*

*BEGIN TRANSACTION;*

*UPDATE accounts SET balance = balance + 50 WHERE account\_id = 1;*

*-- Try to commit, will wait for Session 1 to release lock*

*COMMIT;*

***In this example:***

*Session 1 locks account 1.*

*Session 2 updates account 1, but waits for Session 1 to release the lock.*

*Session 1 updates account 1 and commits, releasing the lock.*

*Session 2 commits.*

*This demonstrates how isolation levels and locking control concurrency.*