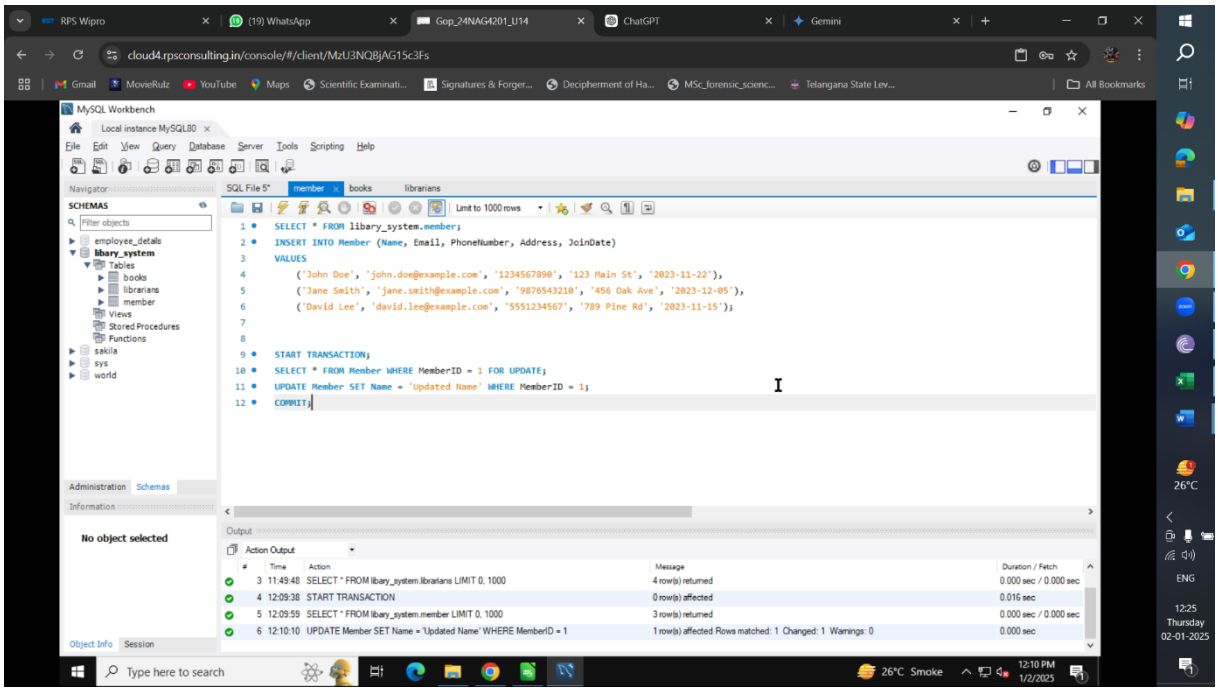
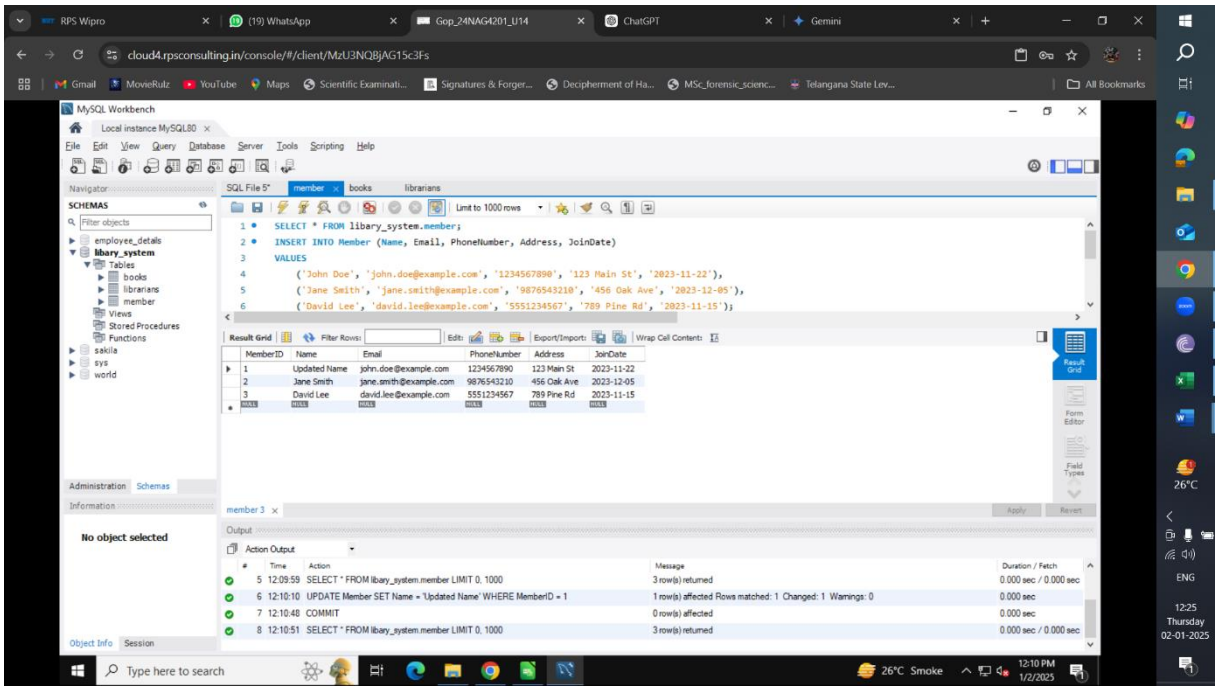
Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

ACID Properties of a Transaction

- Atomicity:
  - Meaning: A transaction is an indivisible unit. Either all operations within the transaction are successfully executed, or none of them are.
  - Example: If you're transferring money between accounts, either both the debit and credit operations succeed, or neither happens.
- Consistency:
  - Meaning: A transaction must leave the database in a valid state. It should not violate any constraints or rules defined for the data.
  - Example: After a successful transaction, the total balance across all related accounts should remain consistent.
- Isolation:
  - Meaning: Concurrent transactions should not interfere with each other. The effects of one transaction should not be visible to other transactions until it is fully committed.
  - Example: If two users are trying to update the same inventory record simultaneously, the database should ensure that their actions don't lead to incorrect stock levels.
- Durability:
  - Meaning: Once a transaction is committed, its changes are permanent. Even in case of system failures (like power outages), the committed changes should be preserved.
  - Example: If a payment transaction is successful, the funds should be deducted from the customer's account permanently, even if the system crashes during the process.





START TRANSACTION;: Begins a transaction block.
SELECT * FROM Member WHERE MemberID = 1 FOR UPDATE;: This statement retrieves the row with MemberID = 1 and acquires an exclusive lock on it. Other transactions cannot modify this row until the current transaction is committed or rolled back.
UPDATE Member SET Name = 'Updated Name' WHERE MemberID = 1;: Updates the Name of the member with MemberID = 1.
COMMIT;: Commits the transaction, making the changes permanent and releasing the lock.
Isolation Levels

Isolation levels control how transactions interact with each other. Common isolation levels include:

Read Uncommitted (lowest isolation): Allows reading data that has not yet been committed by other transactions (dirty reads).
Read Committed: Prevents dirty reads by only allowing reading data that has been committed by other transactions.
Repeatable Read: Prevents dirty reads and non-repeatable reads (where a repeated read of the same data returns different values).
Serializable (highest isolation): Prevents dirty reads, non-repeatable reads, and phantom reads (where new rows appear or existing rows disappear between reads).
Demonstrating Isolation Levels (Conceptual)

Read Uncommitted: Imagine two transactions:

Transaction A: Reads the Name of a member.
Transaction B: Updates the Name of the same member and then rolls back the changes.
In Read Uncommitted, Transaction A might see the uncommitted changes made by Transaction B, leading to a dirty read.
Read Committed: In this case, Transaction A would not see the changes made by Transaction B until Transaction B commits them.

Repeatable Read: Ensures that if Transaction A reads the Name of a member multiple times within the same transaction, it will always see the same value, even if other transactions modify the Name and commit their changes.

Serializable: Provides the highest level of isolation, preventing all types of anomalies. However, it can significantly impact performance due to increased locking.