1) Explain ACID properties of a transaction with suitable example.

A database transaction must satisfy four key properties — ACID:

1Atomicity
A transaction is “all or nothing”.
If any step fails, the whole transaction is rolled back.

Example:
Money transfer from A to B

- Debit A
- Credit B
  If debit succeeds but credit fails → rollback debit.

---

2 Consistency
Transaction must convert database from one valid state to another valid state.

Example:
Balance of A + B must remain same before and after transfer.

3Isolation
Concurrent transactions should not interfere with each other.

Example:
While T1 is updating A,
T2 should not read intermediate value.

---

4 Durability
Once committed, transaction results must persist even after system failure.

Example:
If system crashes after commit → balances remain updated.

---

2) Define Transaction. Explain various states of a transaction.

A **transaction** is a sequence of database operations forming a single logical unit of work.

Example operations:
Read, Write, Update, Insert, Delete

Transaction States

1 **Active**
Transaction execution in progress.

2 **Partially Committed**
Final statement executed.

3**Committed**
Transaction completed successfully.

4 **Failed**
Error occurred during execution.

5 **Aborted**
Transaction rolled back and database restored.

6**Terminated / End**
Transaction finished after commit or abort.

---

3) Explain Two-Phase Commit Protocol (2PC).

Used in **distributed databases** to ensure atomicity across multiple sites.

Phase-1: Prepare / Voting Phase

Coordinator sends **Prepare(T)** to all participants.

Each site replies:

- **Vote-Commit** → ready to commit

- **Vote-Abort** → cannot commit

Phase-2: Commit Phase

If *all* vote-commit → Coordinator sends **Global-Commit**

Else → Coordinator sends **Global-Abort**

Failure Handling

- If coordinator fails → participants wait/timeout
- Recovery is done using logs

2PC ensures:

Atomicity
 Consistency
 Slow due to blocking behaviour

---

4) Differentiate between Conflict Serializability & View Serializability.

Basis	Conflict Serializability	View Serializability
Definition	Schedules obtained by swapping **non-conflicting operations**	Schedules that produce **same read & write results**
Condition	Conflicting operations must preserve order	Final read, write & initial read must match
Test method	Precedence Graph	Difficult to test
Guarantee	Conflict serializable ⇒ View serializable	View serializable ⇏ Conflict serializable

Practical usage	Used in concurrency control	Theoretical concept

---

---

5) Explain Deadlock with suitable example.

Deadlock occurs when:

> Two or more transactions wait for each other's locked resources
> and none can proceed.

Example

T1 locks A → needs B
T2 locks B → needs A

Result → **circular wait**

Deadlock Conditions

1. Mutual exclusion
2. Hold and wait
3. No pre-emption
4. Circular wait

Deadlock Handling

- Deadlock prevention
- Deadlock avoidance (Wait-Die / Wound-Wait)
- Deadlock detection & recovery

6) Write short note on Log-Based Recovery.

DBMS maintains a **log file** to support recovery.

Log Record Format

`<TransactionID, DataItem, Old Value, New Value>`

Types of Logging

1 **Undo Logging**
Restores old values during rollback.

2 **Redo Logging**
Reapplies committed updates after crash.

3 **Undo-Redo Logging**
Supports both rollback & recovery.

Commit Logging Rules

- `<T start>` — transaction begins
- `<T, X, V1, V2>` — before & after values logged
- `<T commit>` — commit completed

---

7) Explain Query Processing Steps (or with diagram).

Query processing converts **high-level SQL** into **efficient execution plan**.

Steps

1 **Query Parsing & Syntax Checking**

- verifies query syntax
- produces parse tree

2 **Query Optimization**

- chooses lowest-cost execution plan
- uses statistics

3 **Execution Plan Generation**

4 **Query Execution Engine**

- executes operations
- interacts with storage

---

8) Explain Heuristics in Query Optimization.

Heuristics are **rule-based transformations** to reduce query cost.

Common Heuristic Rules

1. Perform **selection early**
2. Perform **projection early**
3. Replace Cartesian product with **joins**
4. Perform operations on **smaller relations first**
5. Combine selections & projections

Example

Before optimization
σ city="PUNE" (Customer × Account)

After heuristic optimization
σ city="PUNE" (Customer ⋈ Account)

Result → fewer tuples processed