SECTION 1: DBMS Architecture

Q1. Explain 3-Level DBMS Architecture in your own words.

There are three levels in DBMS:

I. External Level
II. Conceptual Level
III. Internal Level

1. External Level

Purpose:
The external level is meant to show only the necessary data to each user. It controls what user is allowed to see and keeps sensitive data hidden.

Real-life example:
In a college database, a student can see only their marks and attendance, while a teacher can see the marks of all students in the class.

If this level is removed:
All users will see the full database which can lead to data misuse and security problems.

2. Conceptual Level

Purpose:
The conceptual level defines the overall structure of the database. It shows what data is stored and how different data items are related.

Real-life example:
This level defines tables like Student, Course, Faculty, and Exams and explains how students are linked to courses and exams.

If this level is removed:

The database will not have a clear structure, making it difficult to manage relationships between students, courses and results.

3. Internal Level

Purpose:

The internal level focuses on how data is physically stored in the system. It deals with storage methods, files, and indexing.

Real-life example :

Student records, exam results, and attendance data are stored in files on the server with indexes to allow faster searching.

If this level is removed:

Data storage and retrieval will be inefficient, causing slow performance and possible data loss.

Q2. Banking System Views

1. What data each role should see?

Customer View:

Customers can see their account balance, recent transactions and personal details. This keeps banking simple for customers and protects their private information.

Bank Teller View:

A bank teller can access customer account details related to deposits, withdrawals and basic account handling. This helps the teller complete daily banking tasks without unnecessary access.

Branch Manager View:

The branch manager can view overall branch performance, total transactions and staff activity. This information helps in supervising operations and making decisions.

Why this separation is important?

Separating views improves security, allows access based on job roles and prevents misuse of confidential banking data.

SECTION 2: CAP Theorem

Q3. Definitions with Examples

Consistency:

Consistency means the same data is shown everywhere.
Example : Bank balance is the same in all branches.

Availability:

Availability means the system always responds.
Example : Instagram loads likes anytime.

Partition Tolerance:

Partition tolerance means the system works despite network failure.
Example : WhatsApp works during server issues.

Q4. Classify the following systems:

System	CP / AP /CA	Explain Reason
Bank transaction system	CP	Correct balance is more important than fast response.
Instagram likes counter	AP	Likes can update late, but the app should keep running.

Netflix streaming	AP	Video should not stop even if some data updates are slow.
Airline ticket booking	CP	One seat must be booked by only one person.

Q5. Uber Data Center Failure Scenario

1. Risks that may occur:

If Uber loses connection between two data centers, the same ride request may be assigned more than once. Prices may be shown incorrectly, and drivers may get matched with the wrong passengers.

2. CAP property Uber should prioritize:

Uber should prioritize **Availability and Partition Tolerance (AP)**.

3. Why:

Uber is a real-time application, so the app must continue working even during network issues. Small differences in data, like temporary pricing or matching errors, can be corrected later, but stopping the service is not acceptable.

SECTION 3: ACID Properties

Q6. ACID with banking logic

Atomicity

- All or nothing
- Failure → partial transfer

Consistency

- Rules maintained
- Failure → wrong balances

Isolation

- Transactions don’t interfere
- Failure → dirty reads

Durability

- Data saved permanently
- Failure → lost transactions

Q7. Analyze this scenario: User transfers ₹5000 from Account A to Account B. System crashes after debit but before credit.

1. Failed property: **Atomicity**
2. DBMS action:
    - Rollback debit
    - Or complete credit using logs
3. Business impact:
    - Customer trust loss
    - Financial mismatch

SECTION 4: Transactions & Concurrency

Q8. Transaction Boundaries

Must be one transaction

- Payment
- Inventory update
- Invoice generation

Why

- Prevent payment without product
- Ensure consistency

Q9. Last Movie Ticket Problem

Data problems

- Double booking
- Lost update

Business impact

- Customer complaints
- Refund loss

Prevention

- Locks
- Transactions
- Isolation level

Q10. Locking Mechanisms (Tabular Form)

Type of Lock	**Purpose**	**Operations Allowed**	**Example**
Shared Lock (S-Lock)	Used when data is only **read**	Read only (no update allowed)	Checking bank account balance
Exclusive Lock (X-Lock)	Used when data is **modified**	Read and Write	Updating account balance after withdrawal

Q11. Deadlock

Scenario

- T1 locks Account A → waits for B
- T2 locks Account B → waits for A

Why

- Circular wait

Solution

- Timeout
- Deadlock detection
- Resource ordering

SECTION 6: Indexing

Q12. Indexing Examples

Library

- Book index helps find page quickly

Amazon

- Product index speeds up search

Q13. Index Types

Query Requirement	Suitable Index Type	Reason
Search by **Customer_ID**	Primary Index / B-Tree Index	Fast lookup using unique key
Retrieve orders sorted by **Order Date**	B-Tree Index	Supports range queries and sorting
Search product by **Product Name**	Hash Index (exact match) / B-Tree (range search)	Hash for quick exact match, B-Tree for alphabetical order

SECTION 7: Analyst Thinking Challenge

Q14. Performance Issue

Questions to DBA

- Table size?
- Query execution plan?
- Existing indexes?

Possible causes

- Missing indexes
- Large joins
- Full table scans

How indexing helps

- Faster lookup
- Reduced scan time
- Better dashboard performance