DATA ANALYTICS

Assignment-2

SECTION-1 (DBMS Architecture)

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

External Level

Purpose: Shows only required data to each user.

Real-life example: In a banking app, a customer sees only their account details.

If removed: Users may access sensitive data, causing security problems.

Conceptual Level

Purpose: Describes overall database structure and relationships.

Real-life example: Defines how customer and account tables are connected.

If removed: Database becomes unorganized and difficult to manage.

Internal Level

Purpose: Handles physical storage of data.

Real-life example: Indexing and disk storage management.

If removed: Data access becomes slow.

Q2. Consider a Banking System — Design views.

Customer — What data they see:

Balance, transactions, personal details

Bank Teller — What data they see:

Customer account info and update access

Branch Manager — What data they see:

Reports and branch performance data

Why separation is important:

It protects sensitive data and ensures proper access control.

SECTION 2 – (CAP Theorem)

Q3. Define Consistency, Availability, Partition Tolerance.

Consistency:

All users see the same updated data at the same time.

Availability:

The system always responds to user requests.

Partition Tolerance:

The system continues working even during network failure.

Q4. Classify the following systems.

C=Consistency

P=Partition tolerance

A=Availability

- Bank transaction system — CP

  Reason: Correct and consistent data is critical.

- Instagram likes counter — AP

  Reason: System must stay available even if counts vary slightly.

- Netflix movie streaming — AP

  Reason: Availability is more important than consistency.

- Airline ticket booking — CP

  Reason: Consistency prevents double booking.

Q5. Scenario Question (If Uber loses connection between two data centers)

1. Risks:

Ride mismatch, duplicate assignment, delay in updates

2. CAP property to prioritize:

Availability and Partition Tolerance

3. Why:

Users expect fast service even during network issues.

SECTION 3 – (ACID Properties)

Q6. Explain each ACID property.

Atomicity

Transaction happens completely or not at all

Example: Bank transfer must debit and credit

If fails: Money may be lost

Consistency

Database remains valid before and after transaction

Example: Correct balance maintained

If fails: Incorrect data stored

Isolation

Transactions do not affect each other

Example: Two transfers processed independently

If fails: Wrong temporary data visible

Durability

Saved data remains after crash

Example: Transfer record saved permanently

If fails: Data lost

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

1. Which ACID property failed?

Atomicity

2. What should DBMS do?

Rollback transaction and restore original balance

3. Business impact if not fixed?

Financial errors and loss of customer trust

SECTION 4 – (Transactions & Concurrency)

Q8. Identify transaction boundaries.

For Online Shopping:

Steps:

1. **Add to cart**
2. **Make payment**
3. **Update inventory**
4. **Generate invoice**

1. Steps that must be one transaction:

Make payment, update inventory, generate invoice

2. Why?

They depend on each other and must succeed together.

Q9. Concurrency Problem (Two users attempt to book last movie ticket simultaneously)

1. Possible data problems:

Double booking, inconsistent seat data

2. Business impact:

Refunds and customer dissatisfaction

3. How control prevents it:

Using locks and transaction management

SECTION 5 – (Locking Mechanism)

Q10. Differentiate Shared Lock & Exclusive Lock.

Shared Lock Purpose: Allows only reading data

Real Example: Checking balance

Exclusive Lock Purpose: Allows updating data

Real Example: Updating balance

Q11. Deadlock Scenario

Design a simple situation where:

Transaction A holds Resource 1

Transaction B holds Resource 2

Both wait for each other

Explain:

Why deadlock occurs

One solution to prevent it

Why deadlock occurs:

Two transactions hold resources and wait for each other.

One solution to prevent it:

Use timeout or deadlock detection.

SECTION 6 – (Indexing)

Q12. Explain Indexing using:

Library example

Indexing is like the catalog in a library. Instead of checking every book one by one, you look at the catalog to find the shelf location quickly. In databases, indexing helps find records faster without scanning the whole table.

Amazon search example

When we search a product on Amazon, results appear instantly because product names and details are indexed. Without indexing, the system would check every product and take more time.

Q13. Decide Index Type (Finging index and justify for given scenario)

Searching customer by Customer_ID

Index Type: Hash Index

Reason: Customer_ID is unique and searched directly, so hashing gives fast lookup.

Sorting orders by Order Date

Index Type: B-Tree Index

Reason: B-Tree supports sorting and range queries efficiently.

Searching products by Product Name

Index Type: B-Tree Index

Reason: Text searching and ordered matching works well with B-Tree structure.

SECTION 7 – (Analyst Thinking Challenge)

Q14. Performance Issue Scenario

Company reports slow query performance for sales dashboard.

As data analyst:

1. What questions will you ask DBA?

Are indexes created on frequently used columns?

How large is the database/table size?

Are there slow or complex queries running?

Is server memory or CPU overloaded?

2. What database elements might cause slowness?

Missing indexes

Very large tables

Complex joins

Poor query design

Locking or concurrency issues

3. How indexing helps solve the problem?

Indexing reduces the need to scan the entire table.

It allows the database to locate required data quickly, improving query speed and dashboard performance.