Database Management Assignment:-

Section A: Introduction to SQL/NoSQL

1. You are working on a project where you need to store large amounts of structured and semi-structured data. Which type of database (SQL or NoSQL) would you choose and why? Explain with a practical example.

    - For storing large amounts of structured and semi-structured data, NoSQL databases (like MongoDB) are preferred due to their flexibility, scalability, and high performance. SQL databases (like MySQL) are best for structured data requiring strict relationships.
    - Example: An IoT system storing real-time sensor data (semi-structured JSON format) would use MongoDB, whereas a banking system (structured transactions) would prefer MySQL.

2. A company wants to migrate from a relational database to a NoSQL database for better scalability. What challenges might they face? Discuss with an example.

    - Migrating from SQL to NoSQL presents challenges like data modeling differences, lack of ACID compliance, and query complexity.
    - Example: An e-commerce company moving from MySQL to MongoDB might struggle with normalizing relational data, leading to duplication. Ensuring data consistency and transaction support in NoSQL requires careful planning.

Section B: Advantages and Disadvantages of SQL/NoSQL
3. You are designing an e-commerce website's database. Explain the advantages and disadvantages of using SQL vs. NoSQL in this scenario.

Database Type	**Advantages**	**Disadvantages**	**Example**
SQL	Ensures data consistency and relational integrity Supports complex queries (JOINs, transactions)	Harder to scale with large traffic Performance issues in high-demand scenarios	Amazon uses MySQL for handling secure payment transactions
NoSQL	Faster read/write speeds for large datasets Schema flexibility allows frequent updates	Lacks strong ACID compliance for critical transactions Complex queries due to lack of structured relationships	Amazon uses DynamoDB for product catalogs to handle millions of items dynamically

4. A banking system requires high consistency and ACID compliance. Which database system (SQL or NoSQL) would you recommend? Justify your answer with a real-world use case.

- A banking system requires high consistency and ACID compliance, making SQL databases (e.g., PostgreSQL, MySQL) the best choice.
- Example:A transaction system in a bank (like Citibank) ensures atomicity (no partial transactions) and consistency (correct balances after transfers), which is not guaranteed in NoSQL.

Section C: Managing Databases
5. You are a database administrator and need to perform routine maintenance on a production database. Describe at least three essential database management tasks you would perform.

As a database administrator, essential tasks include:

1. Backup & Recovery: Ensuring disaster recovery by scheduling backups.
2. Index Optimization: Improving query performance by maintaining indexes.
3. Monitoring & Security: Checking logs, preventing unauthorized access.

Example: A hospital database needs daily backups to prevent patient data loss.
6. An online streaming service needs to optimize its database performance. What strategies can be used for effective database management in this case?

To optimize a streaming service database (e.g., Netflix), strategies include:

1. Partitioning & Indexing for faster queries.
2. Replication & Caching (e.g., using Redis) for quick access.
3. Load Balancing to distribute traffic across servers.

Example: Netflix uses Apache Cassandra (NoSQL) for scalable, fast data retrieval.

Section D: Identifying System Databases in SQL Server
7. List and describe the system databases in SQL Server. Provide one practical use case for each system database.

1. **master** – Stores system-wide configuration (e.g., user logins).
2. **model** – Acts as a template for new databases.
3. **msdb** – Manages jobs, backups, and scheduling tasks.
4. **tempdb** – Stores temporary tables and session-specific data.

Example:

msdb helps in automating backups for disaster recovery.

8. You have accidentally deleted a user database in SQL Server. Which system database would you use to recover it, and how?

If a user database is accidentally deleted, the msdb system database is used to restore it.

Recovery Steps:

1. Check backup history in msdb.dbo.backupset.
2. Restore using:

QUERY: RESTORE DATABASE MyDatabase FROM DISK = 'backup_location.bak'

Section E: Normalization Forms (1NF, 2NF, 3NF, BCNF)
9. Given the following unnormalized table:

OrderID	**CustomerName**	**Product**	**Quantity**	**SupplierName**	**SupplierContact**
101	John Doe	Laptop	1	ABC Ltd.	1234567890
102	Jane Smith	Phone	2	XYZ Inc.	9876543210

Convert it to 1NF, 2NF, and 3NF with proper explanations.

1NF

OrderID	CustomerName	Product	Quantity	SupplierName	SupplierContact
101	John Doe	Laptop	1	ABC Ltd.	1234567890
102	Jane Smith	Phone	1	XYZ Inc.	9876543210
102	Jane Smith	Phone	1	XYZ Inc.	9876543210

2NF

OrderID	CustomerName	Product	Quantity
101	John Doe	Laptop	1
102	Jane Smith	Phone	1
102	Jane Smith	Phone	1

Product	SupplierName	SupplierContact
Laptop	ABC Ltd.	1234567890
Phone	XYZ Inc.	9876543210

3NF

OrderID	CustomerName	Product	Quantity
101	John Doe	Laptop	1
102	Jane Smith	Phone	1
102	Jane Smith	Phone	1

Product	SupplierID
Laptop	1
Phone	2

SupplierID	SupplierName	SupplierContact
1	ABC Ltd.	1234567890
2	XYZ Inc.	9876543210

10. A company is facing redundancy issues in their database. How would applying BCNF help reduce redundancy? Explain with an example.

    BCNF is an advanced version of 3NF that eliminates functional dependencies that cause redundancy. A table is in BCNF if:

1. It is already in 3NF (No Partial or Transitive Dependencies).
2. Every determinant is a candidate key (i.e., no non-trivial functional dependencies exist where a non-key attribute determines another attribute).

    Example: University Database

    Original Table (3NF but Not BCNF)

StudentID	StudentName	Course	Instructor
1	Alice	DBMS	Prof. X
2	Bob	DBMS	Prof. X
3	Charlie	Networks	Prof. Y

Issue:

- Each Course is always taught by one Instructor → Instructor depends on Course, not StudentID.

- Course → Instructor is a functional dependency, but Course is not a candidate key.

  This violates BCNF because Course is not a Primary Key, but it still determines another attribute (Instructor).

StudentID	StudentName	Course
1	Alice	DBMS
2	Bob	DBMS
3	Charlie	Networks

Course	Instructor
DBMS	Prof. X
Networks	Prof. Y