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.**

   ANS: If your data is well-structured and has relationships, SQL is a better choice, but if you need flexibility and speed, NoSQL works best. NoSQL is useful for large, fast-changing data, like social media posts or user activity logs. An example is an online store using SQL for orders and NoSQL for customer reviews. SQL keeps data accurate, while NoSQL handles large traffic smoothly. The choice depends on how your data is used.

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.**

   ANS: Moving from SQL to NoSQL can be hard because they store and manage data differently. SQL uses tables, while NoSQL uses flexible documents or key-value pairs. Ensuring data accuracy in NoSQL is tricky because it focuses more on speed than strict rules. For example, a retail company switching to MongoDB must change its database design and queries. Planning carefully can make migration smooth and prevent data loss.

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.

ANS: SQL is great for an e-commerce site because it ensures that transactions and product data stay accurate. However, it can slow down when traffic increases. NoSQL is better for things like customer reviews and recommendations because it handles large amounts of data quickly. The downside of NoSQL is that it may sometimes show outdated stock levels. Many websites use both—SQL for orders and NoSQL for user interactions.

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.

ANS: A banking system needs a strong, reliable database, so SQL is the best choice. It ensures transactions like deposits and withdrawals are processed correctly. NoSQL is faster but doesn’t always guarantee perfect accuracy, making it risky for banks. For example, a bank using PostgreSQL ensures that every money transfer is safely recorded. Without SQL’s strict rules, financial data could become inconsistent.

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.

ANS: Database management includes taking regular backups to avoid data loss, optimizing queries to make searches faster, and keeping security up to date. Monitoring performance helps prevent slowdowns. Checking data for errors ensures accuracy. Deleting unnecessary data saves storage space and keeps the database running smoothly.

6. An online streaming service needs to optimize its database performance. What strategies can be used for effective database management in this case?

ANS: A streaming service can make its database faster by storing frequently used data in a cache. Splitting data across multiple servers improves speed. Load balancing helps handle high traffic so the system doesn't crash. Optimizing queries ensures videos load quickly. These methods make streaming smoother with less buffering.

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.

ANS: SQL Server has important system databases: Master (stores main settings), Model (helps create new databases), MSDB (handles backups and schedules tasks), and TempDB (manages temporary storage). The Master database keeps track of server configurations. The Model database provides a template for new databases. MSDB helps with automated backups and job scheduling. TempDB speeds up temporary operations.

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

ANS: If a database is deleted by mistake, it can be recovered using the MSDB database, which stores backup records. The administrator can restore it from the latest saved backup. Transaction logs help recover recent changes. Regular backups are important to prevent losing important data. A disaster recovery plan ensures quick restoration and minimal downtime.

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**

OrderID	**CustomerName**	**Product**	**Quantity**	**SupplierName**	**SupplierContact**
102	**Jane Smith**	**Phone**	**2**	**XYZ Inc.**	**9876543210**

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

ANS: First Normal Form (1NF) - Removing Repeating Groups

To achieve 1NF, we must ensure atomicity, meaning each field should have a single value, and there should be no duplicate columns or multivalued attributes. Since the table does not have repeating product columns, it is already in 1NF.

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

Second Normal Form (2NF) - Removing Partial Dependencies

To achieve 2NF, we must ensure that all non-key attributes depend on the entire primary key and not just a part of it. Since a composite key (OrderID, Product) is required to uniquely identify each row, we must separate order details from customer information.

Orders Table (OrderID is the Primary Key)

OrderID	CustomerName
101	John Doe
102	Jane Smith

OrderDetails Table (OrderID + Product is the Primary Key)

OrderID	Product	Quantity	SupplierName	SupplierContact
101	Laptop	1	ABC Ltd.	1234567890
102	Phone	2	XYZ Inc.	9876543210

Now, each non-key attribute fully depends on the whole primary key, eliminating partial dependencies.

Third Normal Form (3NF) - Removing Transitive Dependencies

To achieve 3NF, we must remove transitive dependencies, meaning non-key attributes should not depend on other non-key attributes. Here, SupplierContact depends on SupplierName, so we must move supplier information to a separate table.

Orders Table

OrderID	CustomerName
101	John Doe
102	Jane Smith

OrderDetails Table

OrderID	Product	Quantity	SupplierID
101	Laptop	1	S001
102	Phone	2	S002

Suppliers Table (SupplierID is the Primary Key)

SupplierID	SupplierName	SupplierContact
S001	ABC Ltd.	1234567890
S002	XYZ Inc.	9876543210

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

ANS: BCNF helps reduce repeated data by organizing information into separate tables. If supplier details are stored in every order, BCNF moves them to a different table to avoid duplication. This makes the database cleaner and more efficient. Queries run faster, and storage space is used better. A properly organized database is easier to update and prevents errors.