**MODULE-3: Working with Databases**

1. Compare SQL and NoSQL Databases

comparison between **SQL** and **NoSQL** databases, based on several key factors:

| **Feature** | **SQL (Relational Databases)** | **NoSQL (Non-Relational Databases)** |
| --- | --- | --- |
| **Data Model** | Structured, tabular (rows and columns). Follows a predefined schema. | Flexible, can be document-based, key-value, graph, or wide-column. |
| **Schema** | Rigid – must define schema before inserting data. | Dynamic – schema can be modified easily; some are schema-less. |
| **Scalability** | Vertical scaling (scale-up – stronger hardware). | Horizontal scaling (scale-out – add more servers). |
| **Examples** | MySQL, PostgreSQL, Oracle, Microsoft SQL Server. | MongoDB, Cassandra, Redis, Couchbase, DynamoDB. |
| **Query Language** | Structured Query Language (SQL). | Varies by database (e.g., MongoDB uses BSON-style queries). |
| **Transactions** | Strong support for ACID (Atomicity, Consistency, Isolation, Durability) transactions. | Some support for ACID (depending on DB), but often prefers BASE (Basically Available, Soft state, Eventual consistency). |
| **Best For** | Complex queries, structured data, strict consistency needs (e.g., banking systems). | Large volumes of unstructured or semi-structured data, fast development, scalability (e.g., social media, real-time analytics). |
| **Flexibility** | Less flexible due to schema rigidity. | Highly flexible – accommodates changes easily. |
| **Joins** | Supports joins between tables. | Typically does not support joins (except graph DBs); data often denormalized. |