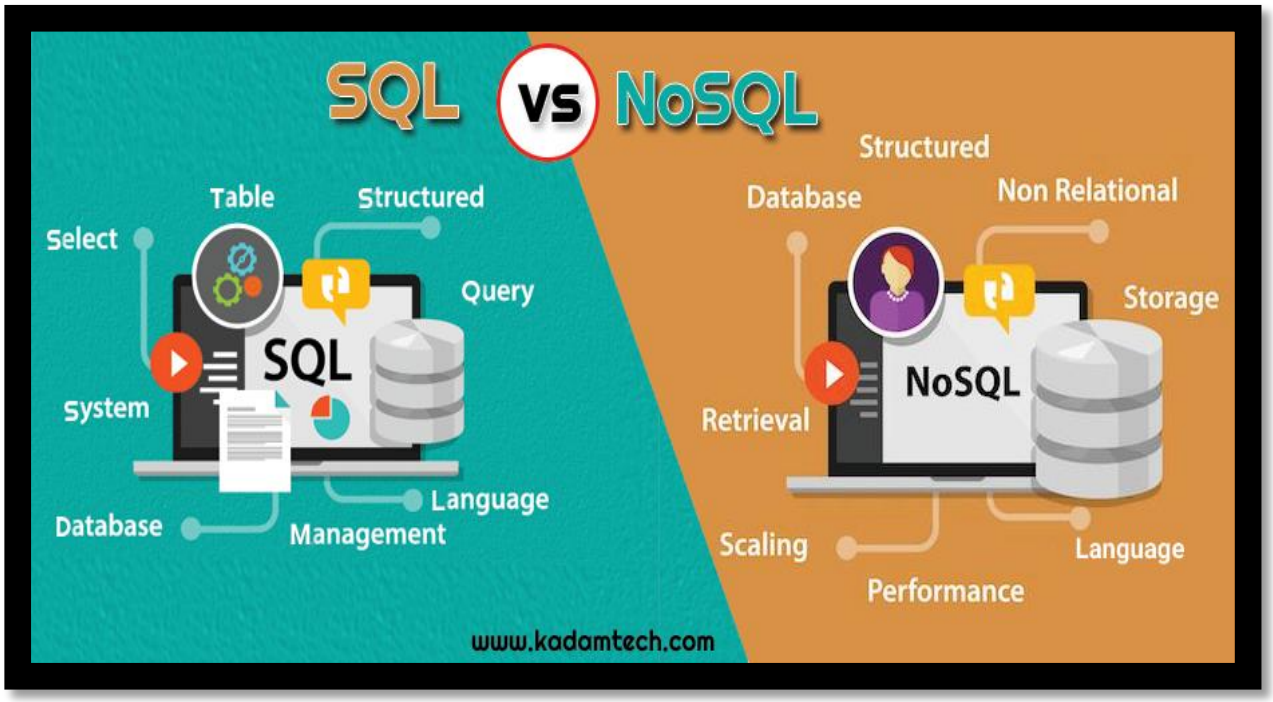
SQL vs NOSQL

What is SQL and what is NOSQL: -



SQL is a language used to manage relational databases — databases that store data in tables with rows and columns.

- **Examples of SQL Databases:**
  - **MySQL**
  - **PostgreSQL**
  - **Microsoft SQL Server**
  - **Oracle Database**

🔹 **Key Features:**

- **Structured data** with a fixed schema (table format)
- **Relationships** between tables using foreign keys
- Supports **ACID transactions** (ensures data integrity)
- Uses **SQL queries** to insert, retrieve, update, and delete data

🔹 **Example SQL Query:**

```
SELECT * FROM users WHERE age > 18;
```

✅ **NoSQL (Not Only SQL)**

NoSQL is a broad term for databases that don't use the traditional relational model. It's used for **unstructured, semi-structured, or rapidly changing data.**

🔹 **Types of NoSQL Databases:**

1. **Document-based** (e.g., MongoDB)
2. **Key-value stores** (e.g., Redis)
3. **Wide-column stores** (e.g., Cassandra)
4. **Graph databases** (e.g., Neo4j)

🔹 **Key Features:**

- **Flexible schema** (can store different types of data in the same place)
- **Horizontal scaling** (easier to scale across multiple servers)

- High performance for large, distributed systems
- Often used in real-time web apps, big data, IoT, etc.

🔹 **Example NoSQL (MongoDB) Query:**

```
db.users.find({ age: { $gt: 18 } })
```

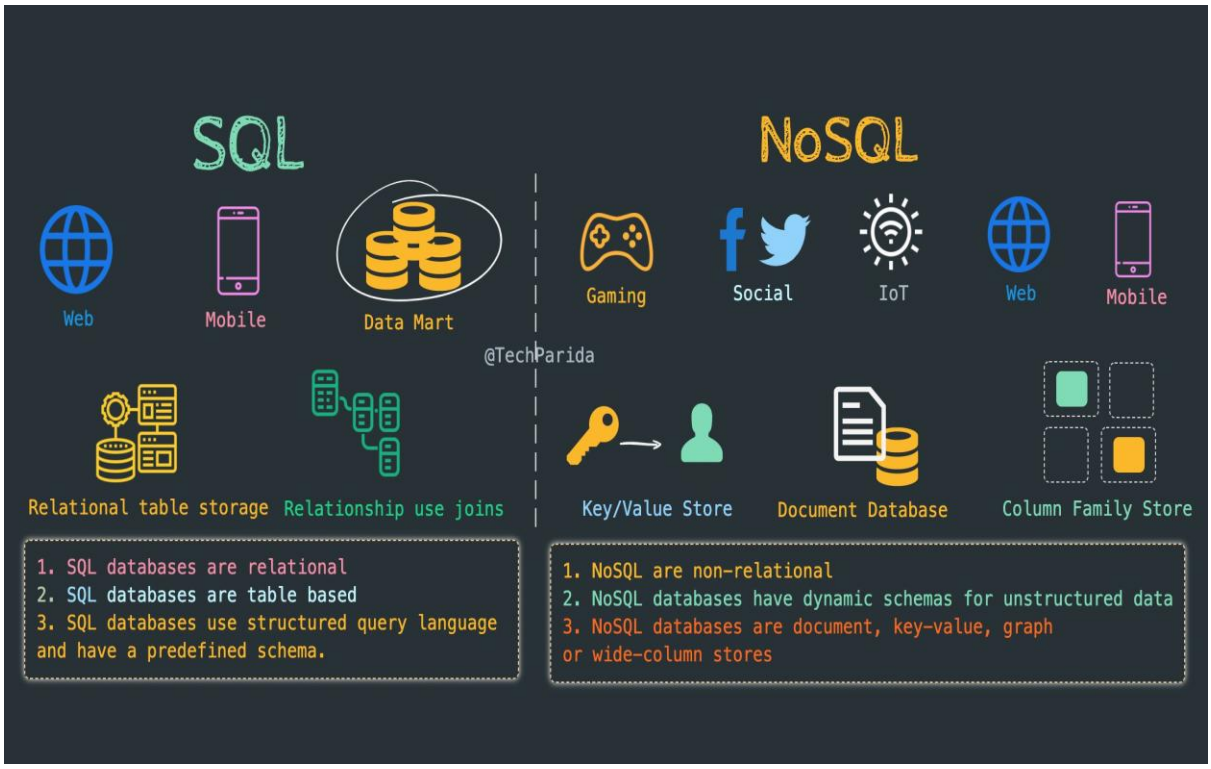
🆚 **SQL vs NoSQL Comparison:**

Feature	SQL	NoSQL
Data Structure	Tables (Rows & Columns)	Documents, Key-Value, etc.
Schema	Fixed	Flexible / Dynamic
Query Language	SQL	Various (Depends on DB)
Relationships	Strong (joins supported)	Weak or manual
Scalability	Vertical (scale up)	Horizontal (scale out)
Use Case	Traditional applications	Big Data, Real-time apps

✅ **Summary:**

- Use **SQL** when your data is structured and relationships are important.
- Use **NoSQL** when you need speed, flexibility, scalability, or are dealing with unstructured data.

What are the scenarios in which we have to choose one over the another



🔁 SQL vs NoSQL: When to Use Which

Requirement	Use SQL	Use NoSQL
Structured data with fixed schema	✅	
Unstructured or dynamic data		✅
Complex joins and relationships	✅	
High-speed reads/writes		✅
Strong data integrity / ACID transactions	✅	
Big Data / horizontal scalability		✅
Flexible and fast development		✅

✅ **Use SQL When:**

- Your data is highly **structured**
- You require **relationships** between entities (e.g., foreign keys)
- You need **transactional accuracy** (e.g., banking, inventory)
- **Schema** changes are rare
- Complex **queries and joins** are common
-

✅ **Use NoSQL When:**

- Your data is **semi-structured** or **unstructured**
- The schema needs to be **flexible** or change frequently
- You're handling **large-scale, distributed,** or **real-time** applications
- You prioritize **speed and performance** over strict consistency
- You need to store data like **JSON, key-value pairs, graphs, or documents**

Advantages of SQL and NOSQL

✅ **Advantages of SQL Databases**

Advantage	Description
🔒 **ACID Compliance**	Ensures **data integrity** with transactions – atomic, consistent, isolated, durable.
📊 **Structured Data with Schema**	Data is organized in **tables with strict schema**, which enforces data quality.
🔗 **Strong Relationships**	Built-in support for **foreign keys and joins** – ideal for relational data.
📈 **Powerful Query Language (SQL)**	Standardized, well-known language for complex queries, aggregations, and filtering.
🛠️ **Mature Ecosystem & Tools**	Tools for reporting, analytics, and data management are widely available.
🧪 **Data Validation & Constraints**	Enforces rules (e.g., NOT NULL, UNIQUE, CHECK) directly at the database level.

✅ Advantages of NoSQL Databases

Advantage	Description
🔄 **Flexible Schema**	No fixed structure – allows storing different formats (e.g., JSON, key-value).
⚡ **High Performance**	Optimized for **fast reads/writes**, especially under heavy loads.

Advantage	Description
🌍 **Horizontal Scalability**	Designed to scale **across many servers** (distributed architecture).
🛠️ **Handles Unstructured Data**	Ideal for storing logs, social media data, IoT data, multimedia, etc.
🚀 **Rapid Development**	Developers can move quickly without worrying about altering rigid schemas.
🔗 **Specialized Models**	Supports document (MongoDB), key-value (Redis), graph (Neo4j), column (Cassandra), etc.
💡 **Eventual Consistency** (optional)	Useful when availability and partition tolerance are more important than strict consistency (CAP theorem).