📘 NoSQL Notes (Not Only SQL)

From Basics to Advanced with Definitions & Examples

🔹 What is NoSQL?

NoSQL refers to a type of **non-relational database** that stores and retrieves data in formats other than traditional **rows and columns** used by relational databases like MySQL or PostgreSQL.

- It stands for **"Not Only SQL"**.
- Designed for **large-scale data**, **real-time web apps**, **unstructured/semi-structured data**.

🔑 Key Features of NoSQL

Feature	Description
Schema-less	No predefined structure
Horizontal Scaling	Easily scalable by adding more machines
High Performance	Faster read/write operations
Supports Big Data	Handles large volumes of unstructured/semi-structured data
Flexible Data Models	JSON, XML, Key-Value, etc.

🧩 4 Types of NoSQL Databases

1. Document-based (e.g., MongoDB)

- Stores data in **JSON** or **BSON** format
- Each document is like a record (object)

Example:

json
CopyEdit

```
{
  "_id": 1,
  "name": "Dhanush",
  "course": "BCA",
  "skills": ["HTML", "CSS", "JavaScript"]
}
```

MongoDB Query:

js
CopyEdit

```
db.students.find({ name: "Dhanush" });
```

---

2. Key-Value Stores (e.g., Redis, DynamoDB)

- Stores data as **key-value pairs**
- Very fast lookup and caching

Example:

txt
CopyEdit

```
"user:1001" → "Dhanush"
```

Redis Commands:

bash
CopyEdit

```
SET user:1001 "Dhanush"
```

```
GET user:1001
```

---

3. Column-based (e.g., Apache Cassandra, HBase)

- Data is stored in **columns** rather than rows.
- Ideal for **analytical queries** over huge datasets.

Example:

ID	Name	Email
1	Dhanush	dhanush@email.com

Cassandra Query:

sql
CopyEdit
```
SELECT name FROM students WHERE id=1;
```

---

4. Graph-based (e.g., Neo4j)

- Designed to handle **networked data** with relationships.
- Uses nodes (entities) and edges (relationships).

Example:

scss
CopyEdit
```
(Dhanush) – [FRIENDS_WITH] –> (Arjun)
```

Neo4j Cypher Query:

cypher
CopyEdit
```
MATCH (a:Person)-[:FRIENDS_WITH]->(b:Person)
```

```
RETURN a.name, b.name;
```

🧰 When to Use NoSQL

Scenario	NoSQL is Suitable?
Big data / real-time apps	✅ Yes
Schema flexibility is needed	✅ Yes
Complex relationships between data	❌ Use Graph DB
Fixed schema & strong integrity required	❌ Use SQL
Fast read/write for large traffic	✅ Yes

🧪 Basic MongoDB Commands

js
CopyEdit

```
// Create database and collection
use school;
db.students.insertOne({ name: "Dhanush", age: 21 });

// Read
db.students.find({ name: "Dhanush" });

// Update
db.students.updateOne({ name: "Dhanush" }, { $set: { age: 22 } });

// Delete
db.students.deleteOne({ name: "Dhanush" });
```

🧾 Advantages of NoSQL

- Handles **huge volumes of data**
- **Faster development** cycles
- Ideal for **cloud-based** and **real-time** applications
- **No complex joins** or normalization needed

🧱 Popular NoSQL Databases

DB Name	Type	Use Case
MongoDB	Document	Web apps, real-time dashboards
Redis	Key-Value	Caching, session management
Cassandra	Column	Big data, time-series storage
Neo4j	Graph	Social networks, recommendations
CouchDB	Document	Offline-first mobile apps
Firebase	Realtime DB	Android/Web app backend

🔁 SQL vs NoSQL – Quick Comparison

Feature	SQL (MySQL/PostgreSQL)	NoSQL (MongoDB, etc.)
Structure	Tables, rows, columns	Documents, key-values
Schema	Fixed	Flexible / dynamic
Relationships	Strong (JOINs)	Weak or none
Scaling	Vertical	Horizontal (easier)
Language	SQL	Varies (Mongo, Cypher)
Best For	Structured, relational data	Big, unstructured data

📦 Mini Project Idea (NoSQL)

Student Record in MongoDB

json
CopyEdit

```
{
  "_id": 101,
  "name": "Dhanush",
  "department": "Computer Science",
  "attendance": {
    "2025-07-25": "Present",
    "2025-07-26": "Absent"
  }
}
```

Query to find absent students:

js
CopyEdit

```
db.students.find({ "attendance.2025-07-26": "Absent" });
```

📘 NoSQL Notes – Part 2 (Advanced + Real-World Concepts)

🔷 1. Advanced MongoDB Queries

🔸 Find with Conditions

js

CopyEdit

```
db.students.find({ age: { $gt: 20, $lte: 25 } });
```

🔸 Projection (only specific fields)

js
CopyEdit

```
db.students.find({}, { name: 1, age: 1, _id: 0 });
```

🔸 Logical Operators

js
CopyEdit

```
db.students.find({ $or: [{ age: 20 }, { name: "Dhanush" }] });
```

🔸 Sorting & Limiting

js
CopyEdit

```
db.students.find().sort({ age: -1 }).limit(5);
```

🔷 2. Aggregation Framework in MongoDB

Powerful for transforming and analyzing data.

Example: Grouping and Average

js
CopyEdit

```
db.students.aggregate([
  { $group: { _id: "$course", avgAge: { $avg: "$age" } } }
]);
```

Example: Filtering + Counting

js
CopyEdit

```
db.students.aggregate([
  { $match: { age: { $gt: 18 } } },
```

```
  { $count: "TotalAbove18" }
]);
```

🔷 3. MongoDB Indexing

Indexing boosts query performance.

js
CopyEdit
```
db.students.createIndex({ name: 1 });
```

Types of Indexes:

- Single Field
- Compound
- Text
- Geospatial

🔷 4. Redis Advanced Concepts

🔸 Data Structures in Redis

- `String` – basic value
- `List` – ordered list
- `Set` – unique unordered values
- `Sorted Set` – ranked set
- `Hash` – key-value inside key

Example: Hash for user profile

bash
CopyEdit

```
HMSET user:101 name "Dhanush" age "21"
HGETALL user:101
```

🔸 TTL – Time to Live

Set expiry for keys (e.g., caching):

bash
CopyEdit

```
SET session:101 "active"
EXPIRE session:101 60
```

🔷 5. CAP Theorem (Important for NoSQL)

CAP = Consistency, Availability, Partition Tolerance
You can **only achieve 2 at a time** in a distributed system.

DB Type	Favored Properties
MongoDB	CP
Cassandra	AP
Redis	AP

🔷 6. Real-time Sync with Firebase

Firebase is a real-time NoSQL database by Google.

json
CopyEdit

```
{
  "users": {
    "101": {
```

```
      "name": "Dhanush",
      "online": true
    }
  }
}
```

Updates reflect **instantly across devices**.

🔷 7. Data Modeling in NoSQL

Unlike SQL (normalized), NoSQL favors **embedding** or **referencing**.

🔸 Embedding (One-to-Many inside One document)

json
CopyEdit

```
{
  "name": "Dhanush",
  "courses": [
    { "title": "HTML" },
    { "title": "CSS" }
  ]
}
```

🔸 Referencing (Linked via IDs)

json
CopyEdit

```
{ "_id": 1, "name": "Dhanush", "course_ids": [1, 2] }

{ "_id": 1, "title": "HTML" }
{ "_id": 2, "title": "CSS" }
```

🔷 8. NoSQL Security Basics

- Use authentication (`SCRAM-SHA-1` in MongoDB)
- Enable **IP whitelisting** and **SSL**
- Use **Role-Based Access Control (RBAC)**

🔷 9. Backup & Restore in NoSQL

MongoDB Backup

bash
CopyEdit

```
mongodump --db mydatabase
```

MongoDB Restore

bash
CopyEdit

```
mongorestore --db mydatabase dump/mydatabase
```

🔷 10. NoSQL Use Cases

Use Case	NoSQL Type
Real-time chat apps	Firebase / Redis
Social networks	Neo4j (Graph)
Product catalogs	MongoDB
IoT sensor data	Cassandra / InfluxDB
Caching and sessions	Redis

📊 Example Project: Real-Time Student Portal (MongoDB + Firebase)

1. **Student data** stored in MongoDB
2. **Attendance status** updated in Firebase and reflected live
3. **Login tokens** managed using Redis cache
4. **Graph relationships** of courses using Neo4j