Introduction to Databases

A **database** is an organized collection of data that can be accessed, managed, and updated efficiently. It allows storing data in a structured manner, making it easy to retrieve and manipulate.

Types of Databases

1. **Relational Databases (SQL)**
   - Uses structured schema (tables with rows and columns).
   - Data is stored in a tabular format.
   - Ensures ACID (Atomicity, Consistency, Isolation, Durability) properties.
   - Examples: MySQL, PostgreSQL, SQLite, SQL Server.
2. **Non-Relational Databases (NoSQL)**
   - Schema-less or dynamic schema.
   - Can store data in different formats like documents, key-value pairs, graphs, or wide-columns.
   - Provides high scalability and flexibility.
   - Examples: MongoDB, Cassandra, Redis, Firebase.

SQL vs NoSQL

Feature	SQL (Relational)	NoSQL (Non-Relational)
Structure	Tables (rows & columns)	Documents, Key-Value, Graphs
Schema	Fixed schema	Dynamic schema
Scalability	Vertical Scaling	Horizontal Scaling
Transactions	ACID Compliance	BASE (Basically Available, Soft state, Eventual consistency)
Query Language	SQL	Query languages like MongoDB Query Language (MQL)
Use Case	Structured Data, Banking, ERP	Big Data, Real-time apps, JSON-based

---

MongoDB and PyMongo CRUD Operations

MongoDB is a **NoSQL document-based database**, and **PyMongo** is a Python library to interact with MongoDB.

1. Install PyMongo

```
pip install pymongo
```

2. Connect to MongoDB

```
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["students_db"]
collection = db["students"]
```

CRUD Operations in PyMongo

Create (Insert Data)

```
# Insert One
student = {"name": "Ali", "age": 22, "course": "Database"}
collection.insert_one(student)

# Insert Multiple
students = [
    {"name": "Sara", "age": 21, "course": "AI"},
    {"name": "Ahmed", "age": 23, "course": "Web Dev"}
]
collection.insert_many(students)
```

Read (Retrieve Data)

```
# Find One
student = collection.find_one({"name": "Ali"})
print(student)

# Find All
for student in collection.find():
    print(student)

# Find with Condition
for student in collection.find({"course": "AI"}):
    print(student)
```

Update (Modify Data)

```
# Update One
collection.update_one({"name": "Ali"}, {"$set": {"age": 23}})

# Update Many
collection.update_many({"course": "AI"}, {"$set": {"course": "Machine Learning"}})
```

Delete (Remove Data)

```
# Delete One
collection.delete_one({"name": "Ali"})

# Delete Many
collection.delete_many({"course": "Machine Learning"})
```

Bonus: Advanced MongoDB Queries

- **Sorting**: `collection.find().sort("age", -1)` (Descending)
- **Limiting Results**: `collection.find().limit(5)`
- **Projection (Selecting Fields)**: `collection.find({}, {"_id": 0, "name": 1})`