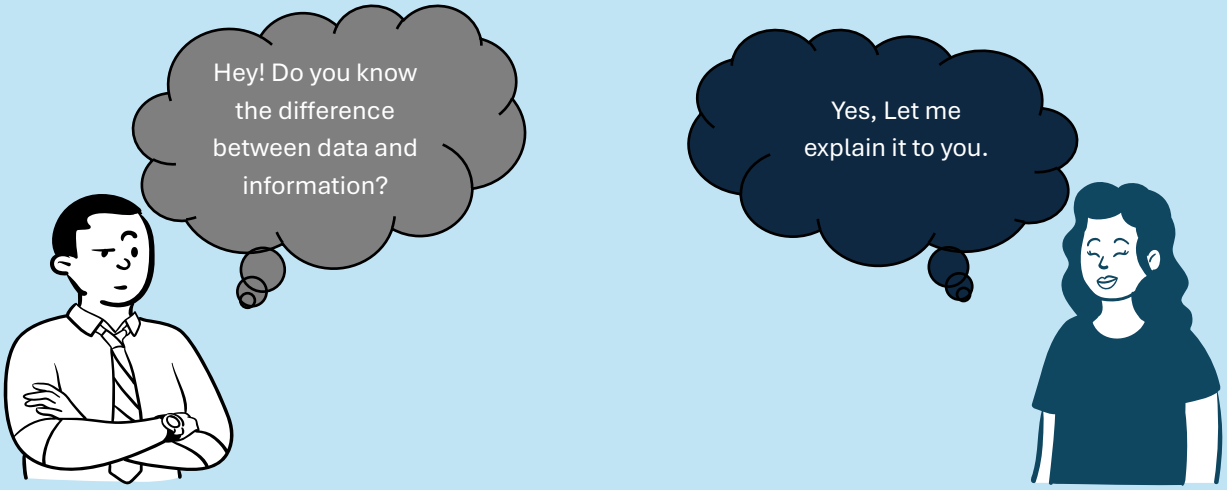
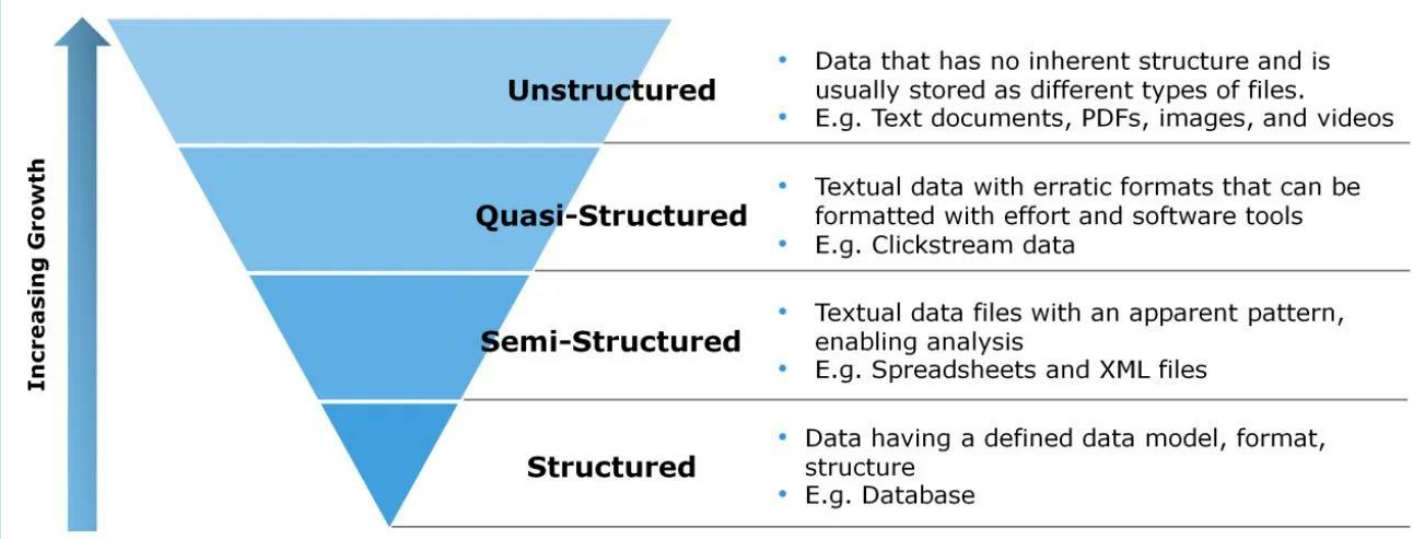


Data represents raw elements or unprocessed facts, including numbers and symbols to text and images. When these pieces are analyzed and contextualized, they transform into something more meaningful is known as **information.**

Here are some more Benefits to review why do we need data:

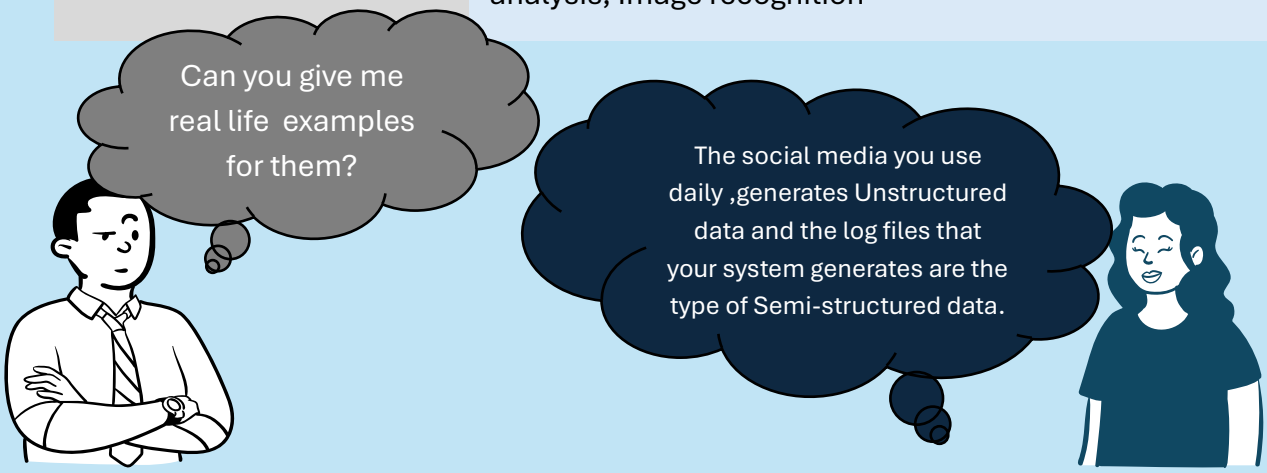
Data can be Classified into 4 major categories:



Ques 1: Now lets talk about the difference between Unstructured and Semi-structured type of data and their examples:

FEATURES	UNSTRUCTURED DATA	SEMI-STRUCTURED DATA
Definition	Data Lacking a predefined format and organisation	Data with some integral structure but lacking a rigid schema
Characteristics	Diverse formats, Challenging to process and analyse, rich and diverse information	Flexible format, Adaptable to evolving data needs, requires sspecialized tools
Advantages	Captures real world context, Valuable to sentiment analysis and trend identification	Flexible and scalable, suitable for real time applications

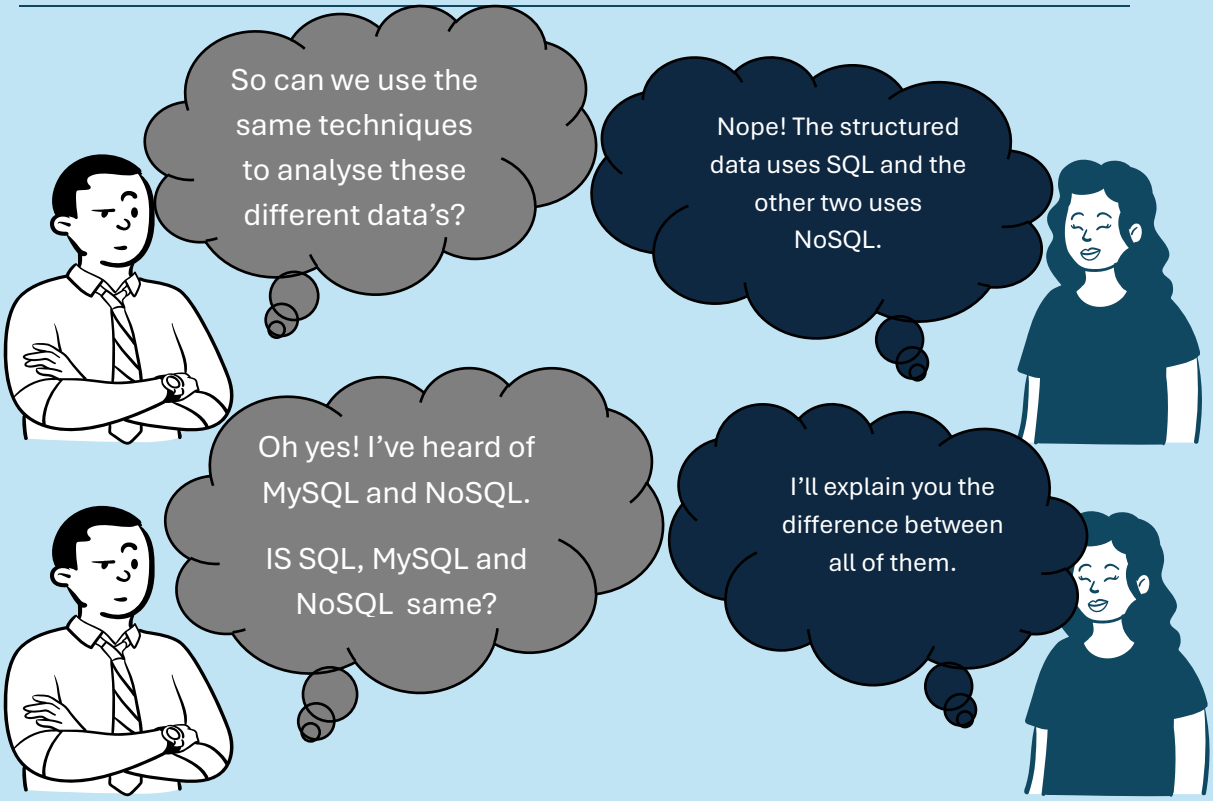
Disadvantages	Difficult to process and analyse, Data quality concerns	Limited data integration potential, lack of standarized formats
Tools and Techniques	Natural language processing, Machine learning, sentiment analysis, Image recognition	JSON and XML parsers, stream processing tools, data pipelines



Here are some more examples and Use Cases of them:

	Unstructured data	**Semi-Structured data**
Examples	Email body text,Social media posts,Blog entries,Photographs,Audio recordings,Videos,Customer reviews,Chat conversations,Literature texts,News articles,Personal diaries,Handwritten notes, **VR/AR Experiences, Live Event Transcripts, Memes, Historical Documents**	JSON files,XML files,HTML documents,CSV files, Email metadata (headers),Web server logs,Configuration files,NoSQL database records,Markup languages (e.g., Markdown),Sensor data with labels,E-commerce transaction records (with structured fields and free text), **YAML Files, BibTeX Entries, Rich Text Format (RTF) Documents, iCalendar Files (.ics)**
Use Cases	Customer feedback analysis,Social media monitoring, Content analysis, Multimedia processing	Real time analysis, Sensor data analysis, Web scrapping, Scientific experiments

Ques 2: Now lets talk about the difference between SQL, NoSQL and MySQL.

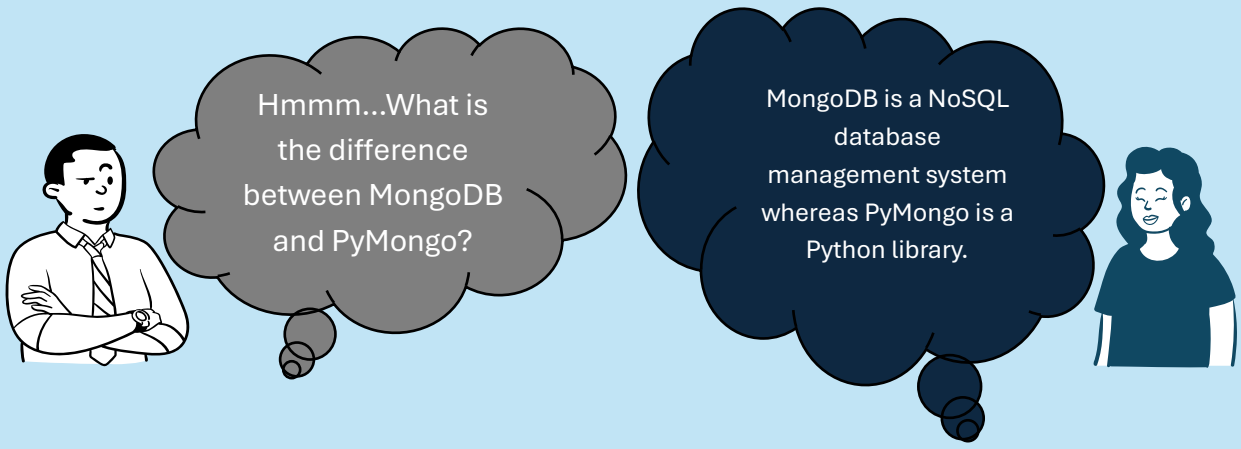


SQL	MySQL
SQL is a programming language that you can use to query and process information in a relational database	MySQL is a relational database management system
You use SQL to interact with, add to, manipulate, and change databases.	As a relational database management system, you use MySQL to create tables for storing related data.
You can use SQL with other relational databases.	You could use a different relational database management system instead.

NoSQL	MySQL
Non-relational, can be document-based, key-value pairs, wide-column stores, or graph databases.	Relational, structured in tables with rows and columns.
Schema-less, allowing for dynamic and flexible data structures.	Fixed schema that requires predefined structures, making it less flexible in handling changing data structures.

Below given is the difference between all three:

	SQL	NoSQL	MySQL
Full form	Structured query language	Not-only-SQL	MySQL (Its name is a combination of "My", the name of co-founder Michael Widenius's daughter My, and "SQL")
Nature	A language used for managing and manipulating relational databases.	A broad category of database management systems that do not necessarily use SQL.	A popular open-source relational database management system (RDBMS) that uses SQL.
Structure	Data is stored in tables (rows and columns).	Can be document-based, key-value pairs, wide-column stores, or graph databases.	Data is organized in tables with rows and columns.
Schema	Schema-based with a predefined structure.	Schema-less or dynamic schema	Schema-based with a predefined structure
Property	Ensures Atomicity, Consistency, Isolation, and Durability (ACID)	Basically available, soft state and eventually consistent(BASE)	Ensures ACID properties
Use cases	Suitable for applications requiring multi-row transactions, such as financial systems.	Ideal for handling large volumes of unstructured or semi-structured data	Suitable for web applications, data warehousing, e-commerce platforms.
Examples	SQL queries	MongoDB (document-based), Redis (key-value), Cassandra (wide-column), Neo4j (graph).	MySQL, PostgreSQL, SQLite



Ques 3: List out all syntax of py-mongo and mongoshell. Include example of mongoshell with syntax.

Py-Mongo syntax with example	Mongo Shell syntax with examples
1. **Connecting to MongoDB** • from pymongo import MongoClient • client = MongoClient('localhost', 27017) • db = client['database_name'] 2. **Inserting Documents** • collection.insert_one({"name": "John", "age": 30}) • collection.insert_many([{"name": "Jane", "age": 25}, {"name": "Doe", "age": 35}]) 3. **Querying Documents** • collection.find_one({"name": "John"}) • collection.find({"age": {"$gt": 25}}) 4. **Updating Documents** • collection.update_one({"name": "John"}, {"$set": {"age": 31}}) • collection.update_many({"age": {"$lt": 30}}, {"$set": {"status": "young"}}) 5. **Deleting Documents** • collection.delete_one({"name": "John"}) • collection.delete_many({"age": {"$lt": 30}}) 6. **Counting Documents** • collection.count_documents({}) • collection.count_documents({"age": {"$gt": 25}}) 7. **Creating Indexes** • collection.create_index([("name", pymongo.ASCENDING)]) 8. **Aggregation** • collection.aggregate([{"$match": {"age": {"$gt": 25}}}, {"$group": {"_id": "$status", "count": {"$sum": 1}}}])	1. **Connecting to MongoDB** • mongo **Show Databases** • show dbs **Use Database** • use database_name **Show Collections** • show collections 2. **Inserting Documents** • db.collection_name.insertOne({"name": "John", "age": 30}) • db.collection_name.insertMany([{"name": "Jane", "age": 25}, {"name": "Doe", "age": 35}]) 3. **Querying Documents** • db.collection_name.findOne({"name": "John"}) • db.collection_name.find({"age": {"$gt": 25}}) 4. **Updating Documents** • db.collection_name.updateOne({"name": "John"}, {"$set": {"age": 31}}) • db.collection_name.updateMany({"age": {"$lt": 30}}, {"$set": {"status": "young"}}) 5. **Deleting Documents** • db.collection_name.deleteOne({"name": "John"}) • db.collection_name.deleteMany({"age": {"$lt": 30}}) 6. **Counting Documents** • db.collection_name.countDocuments({}) • db.collection_name.countDocuments({"age": {"$gt": 25}})

9. Bulk Operations

- bulk = collection.initialize_ordered_bulk_op()
- bulk.insert({"name": "John", "age": 30})
- bulk.find({"name": "Jane"}).update({"$set": {"age": 26}})
- bulk.execute()

10. **Listing Collections**
    - db.list_collection_names()

11. **Dropping a Collection**
    - collection.drop()

12. **Finding Distinct Values**
    - collection.distinct("name")

7. **Creating Indexes**
    - db.collection_name.createIndex({"name": 1})

8. **Aggregation**
    - db.collection_name.aggregate([{"$match": {"age": {"$gt": 25}}}, {"$group": {"_id": "$status", "count": {"$sum": 1}}}])

9. **Bulk Operations**
    - var bulk = db.collection_name.initializeOrderedBulkOp()
    - bulk.insert({"name": "John", "age": 30})
    - bulk.find({"name": "Jane"}).update({"$set": {"age": 26}})
    - bulk.execute()

10. **Listing Collections**
    - db.getCollectionNames()

11. **Dropping a Collection**
    - db.collection_name.drop()

12. **Finding Distinct Values**
    - db.collection_name.distinct("name")

13. **Find All Documents in a Collection**
    - db.collection_name.find()

14. **Find Documents with a Query**
    - db.collection_name.find({"city": "New York"})

Some Extra MongoShell syntax with examples:

1. **Backup Database**
    - mongodump --db database_name --out /path/to/backup
2. **Restore Database**
    - mongorestore /path/to/backup

3. Find Documents with Regex

- db.collection_name.find({"name": {"$regex": "^A", "$options": "i"}})

4. **Update Documents with Increment Operator**
    - db.collection_name.updateOne({"name": "Bob"}, {"$inc": {"age": 1}})
5. **Find Documents with Array Field**
    - db.collection_name.find({"tags": "mongodb"})
6. **Find Documents with Embedded Document**

- db.collection_name.find({"address.city": "New York"})

7. **Push to Array Field**
   - db.collection_name.updateOne({"name": "Alice"}, {"$push": {"hobbies": "cycling"}})
8. **Add Unique Values to Array Field**
   - db.collection_name.updateOne({"name": "Alice"}, {"$addToSet": {"hobbies": "reading"}})
9. **Remove Element from Array Field**
   - db.collection_name.updateOne({"name": "Alice"}, {"$pull": {"hobbies": "cycling"}})
10. **Project Specific Fields**
    - db.collection_name.find({}, {"name": 1, "age": 1, "_id": 0})
11. **Sort Query Results**
    - db.collection_name.find().sort({"age": -1})
12. **Limit Query Results**
    - db.collection_name.find().limit(5)
13. **Skip Query Results**
    - db.collection_name.find().skip(10)
14. **Use $text for Text Search**
    - db.collection_name.createIndex({"description": "text"})
    - db.collection_name.find({"$text": {"$search": "mongodb"}})
15. **Use $expr for Field Comparison**
    - db.collection_name.find({"$expr": {"$gt": ["$spent", "$budget"]}})
16. **Rename a Field**
    - db.collection_name.updateMany({}, {"$rename": {"oldFieldName": "newFieldName"}})