# MongoDB

**MongoDB** is a document database. It stores data in a type of JSON format

A record in is document and we have key values pairs

Example Document

```
{ title: "Post Title 1",
 tags: ["news", "events"],
 date: Date()}
```

**SQL Databases:**
Relational, stores related data in separate tables.
Data is queried from multiple tables using joins.

**MongoDB:**
Document database (non-tabular).
Stores data in flexible documents, keeping related data together.
Fast data reading due to consolidated documents.
Collections are used instead of tables for grouping data.

You can use the **MongoDB Query API** to perform:
. Adhoc queries with mongosh, Compass, VS Code, or a MongoDB driver for the programming language you use.
. Data transformations using aggregation pipelines.
. Document join support to combine data from different collections.
. Graph and geospatial queries.
. Full-text search.
. Indexing to improve MongoDB query performance.
Time series analysis.

**Database using mongosh**

*you can see which database you are using by typing db in your terminal.

*To see all available databases, type show dbs

change or create new database

```
use database_name
```

**Create Collection**

1,You can create a collection using the createCollection()

```
db.createCollection
("CollectionName")
```

2,You can also create a collection during the insert process.object is a valid JavaScript object containing post data

```
db.CollectionName.insertOne
(object)
```

**Insert Documents**

1,insert a single document

```
db.CollectionName.insertOne({
title: "Post Title 1",
date: Date() })
```

2,insert multiple documents at once

```
db.CollectionName.insertMany([
 {
  title: "Post Title 2",
  category: "Event",
 },
 {
  title: "Post Title 4",
  category: "Event",
 } ])
```

**Find Data**

```
db.CollectionName.find(
{category: "News",active:1} )
```

second parameter is an optional object that describes which fields to include in the results.1 to include a field and 0 to exclude a field.

```
db.Collection.find({category:
"News"}, {title: 1, date: 1})
```

**Update Document**

1,update the first document that is found matching the provided query.(find post with title=title1 and update likes to 2)

```
db.posts.updateOne( { title: "Post
Title 1" }, { $set: { likes: 2 } } )
```

2,Update the document, but if not found insert it

```
db.posts.updateOne(
 { title: "Post Title 5" },
 {$set:
   { title: "Post Title 5",
     body: "Body of post.",
     tags: ["news", "events"],
     date: Date()  }
 }, { upsert: true })
```

3.update all documents that match the provided query.(Update likes on all documents by 1)

```
db.posts.updateMany({}, { $inc:
{ likes: 1 } })
```

**Delete Documents**

1,delete the first match

```
db.posts.deleteOne({ title: "Post
Title 5" })
```

2,delete all matches

```
db.posts.deleteMany({ category:
"Technology" })
```

**Query Operators**

**Comparison**

$eq: Values are equal
$ne: Values are not equal
$gt: Value is greater than another value
$gte: Value is greater than or equal to another value
$lt: Value is less than another value
$lte: Value is less than or equal to another value
$in: Value is matched within an array

# MongoDB

**Logical**

$and: Returns documents where both queries match

$or: Returns documents where either query matches

$nor: Returns documents where both queries fail to match

$not: Returns documents where the query does not match

**Evaluation**

$regex: Allows the use of regular expressions when evaluating field values

$text: Performs a text search

$where: Uses a JavaScript expression to match d

## Update Operators

**Fields**

The following operators can be used to update fields:

$currentDate: Sets the field value to the current date

$inc: Increments the field value

$rename: Renames the field

$set: Sets the value of a field

$unset: Removes the field from the document

**Array**

The following operators assist with updating arrays.

$addToSet: Adds distinct elements to an array

$pop: Removes the first or last element of an array

$pull: Removes all elements from an array that match the query

$push: Adds an element to an array

Set the lastModified field to the current date.

```
db.students.updateOne(
  { _id: 1 },{ $currentDate: { lastModified: true } });
```

Add "Art" to the courses array

```
db.students.updateOne({ _id: 1 },{ $push: { courses: "Art" } });
```

## Aggregation Pipelines

$match: Filters documents based on a condition, similar to a WHERE clause in SQL.

Example: Filter students who are enrolled.

`{ $match: { enrolled: true } }`

$group: Groups documents by a specified field and can apply aggregate functions like sum, average, count, etc.

Example: Group students by their major and count the number of students in each major.

`{ $group: { _id: "$major", studentCount: { $sum: 1 } } }`

$sort: Sorts the documents by a specified field.

Example: Sort the grouped results by student count in descending order.

`{ $sort: { studentCount: -1 } }`

$project: Shapes the documents by including, excluding, or adding fields.

Example: Include only the major and student count in the output.

`{ $project: { major: "$_id", studentCount: 1, _id: 0 } }`

$limit: Limits the number of documents passed to the next stage.

Example: Limit the output to the top 3 majors.

`{ $limit: 3 }`

$lookup: Performs a left outer join to another collection.

Example: Lookup additional information about each student's courses from a courses collection.

```
{ $lookup: {
    from: "courses",
    localField: "courses",
    foreignField: "courseId",
    as: "courseDetails"
}}
```

$addFields: Adds new fields to documents.

Example: Add a new field for full name by concatenating first and last names.

`{ $addFields: { fullName: { $concat: ["$firstName", " ", "$lastName"] } } }`

```
db.students.aggregate([
   // Match only enrolled students { $match: { enrolled: true } },
   // Group by major and count students   { $group: { _id: "$major", studentCount: { $sum: 1 } } },
   // Sort by student count { $sort: { studentCount: -1 } },
  // Project the result { $project: { major: "$_id", studentCount: 1, _id: 0 } },
 // Limit to top 3 majors{ $limit: 3 }
]);
```